Оставьте отзыв о скачивании PDF-файла.

# Использование WinDbg и команд отладчика

Статья • 28.10.2023

WinDbg — это отладчик, который можно использовать для анализа аварийных дампов, отладки динамического кода в пользовательском режиме и режиме ядра, а также проверки регистров ЦП и памяти.

##  Начало работы с WinDbg

Эта последняя версия предоставляет более современный пользовательский интерфейс с обновленным интерфейсом, полноценными возможностями создания скриптов, расширяемой моделью данных отладки, встроенной поддержкой отладки по времени (TTD) и множеством дополнительных функций.

Дополнительные сведения см. в статье Общие сведения о WinDbg.

Сведения об установке отладчика см. в статье Установка отладчика Windows.

Чтобы приступить к работе с WinDbg, см. статью начало работы с отладкой Windows.

## Команды отладчика

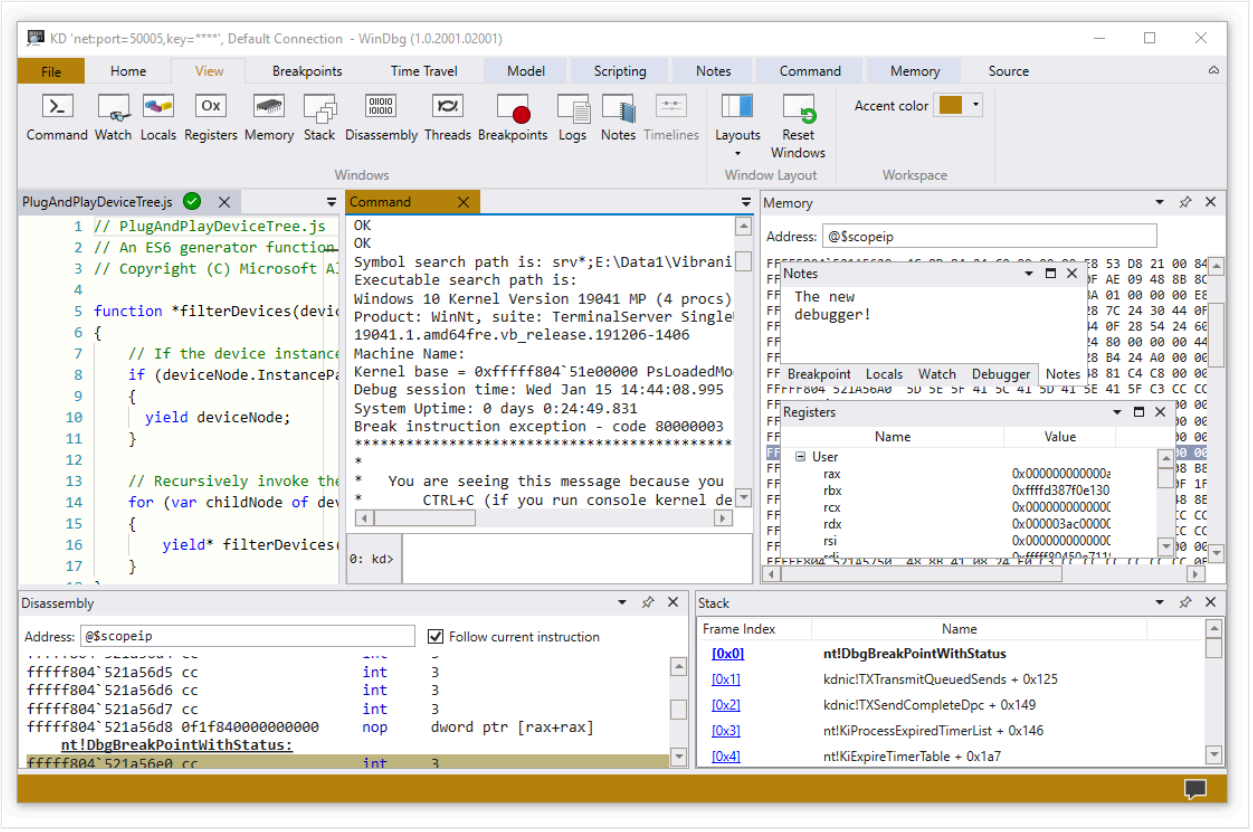
Дополнительные сведения о командах отладчика см. в статье Использование команд отладчика.

# Что такое WinDbg?

Статья • 03.04.2025

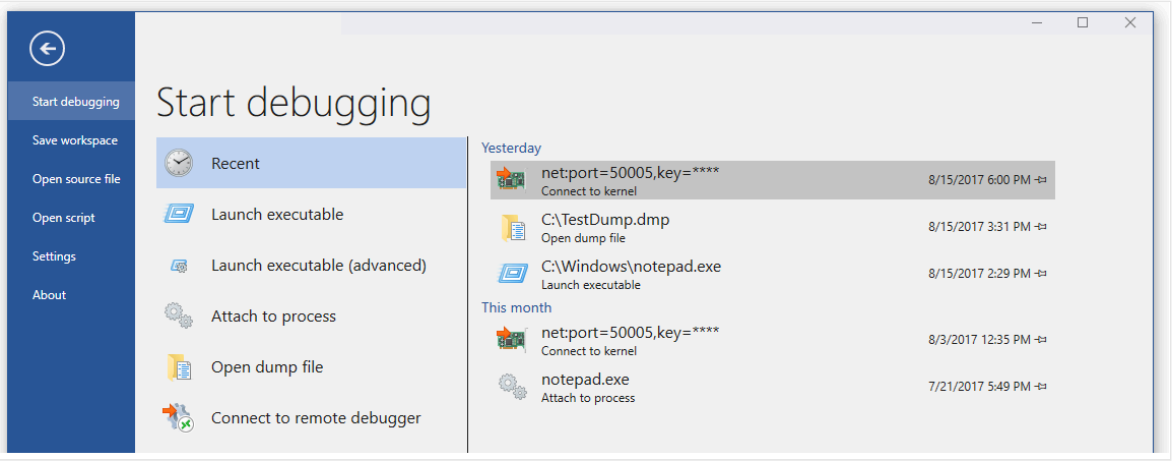
WinDbg — это последняя версия WinDbg с более современными визуальными элементами, более быстрыми окнами, полноценной возможностью для скриптов, созданная с акцентом на расширяемую модель данных отладчика.

> **Примечание**
>
> Ранее выпущено как *WinDbg Preview* в Microsoft Store , *WinDbg* использует тот же базовый механизм, что и *WinDbg (классическая версия)* и поддерживает все те же команды, расширения и рабочие процессы.

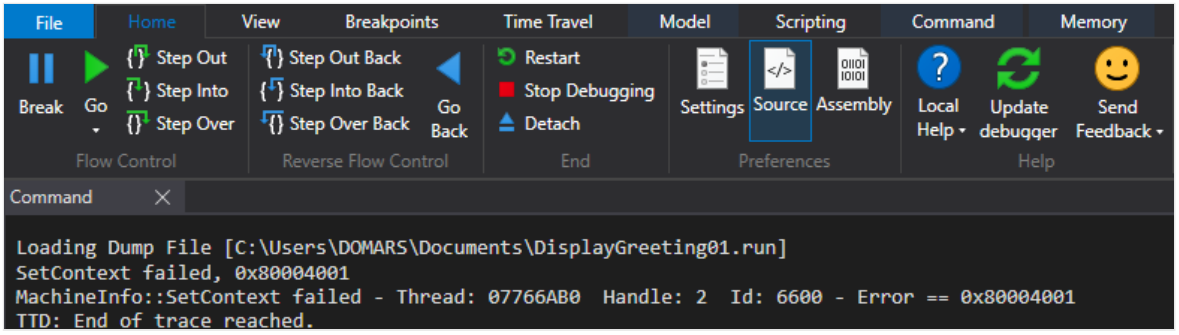



# Общие функции

- **Настройка подключения и восстановление** — сохраняются последние цели и конфигурации сеанса. Их можно быстро перезапустить из меню файла.




- **Темная тема** — перейдите к параметрам файла > , чтобы включить темную тему.




- **Навигация по клавиатуре** — используйте клавиши CTRL+TAB, чтобы легко перемещаться между окнами с помощью клавиатуры.



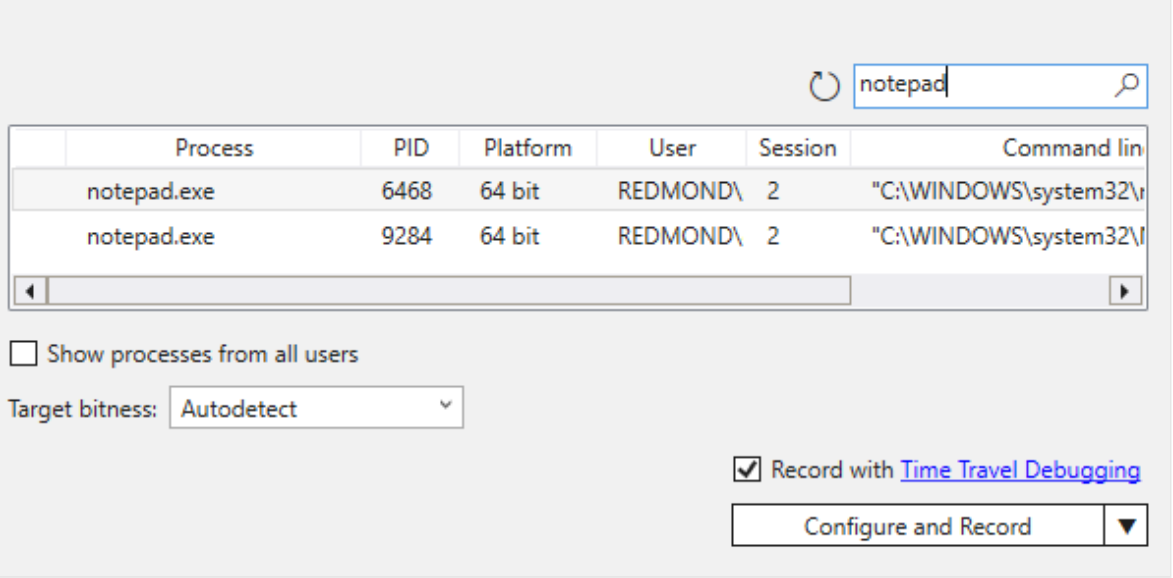

- **Обнаружение архитектуры процессора для файлов дампа** — автоматическое определение архитектуры процессора для упрощения управляемой отладки.

- **Улучшения производительности** — окна инструментов загружаются асинхронно и могут быть отменены. При выполнении команды WinDbg может остановить загрузку локальных языков, часов или других окон.

# Запуск режима отладки

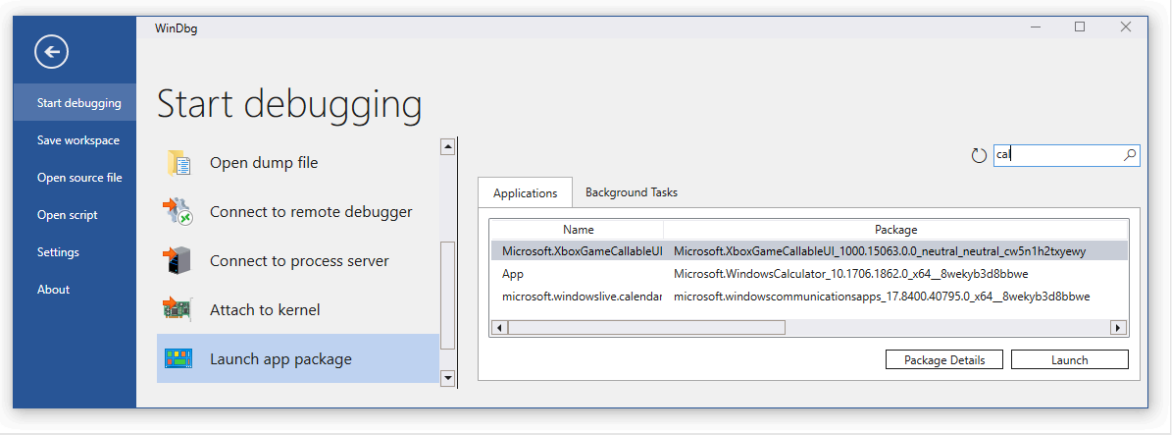
- **Встроенная отладка перемещения по времени (TTD)** — установите флажок "Запись с отладкой перемещения по времени" при запуске или присоединении к процессу. WinDbg настроит TTD, начнет запись и откроет трассировку после этого.

  Дополнительные сведения см. в статье "Отладка временных путешествий" — обзор.

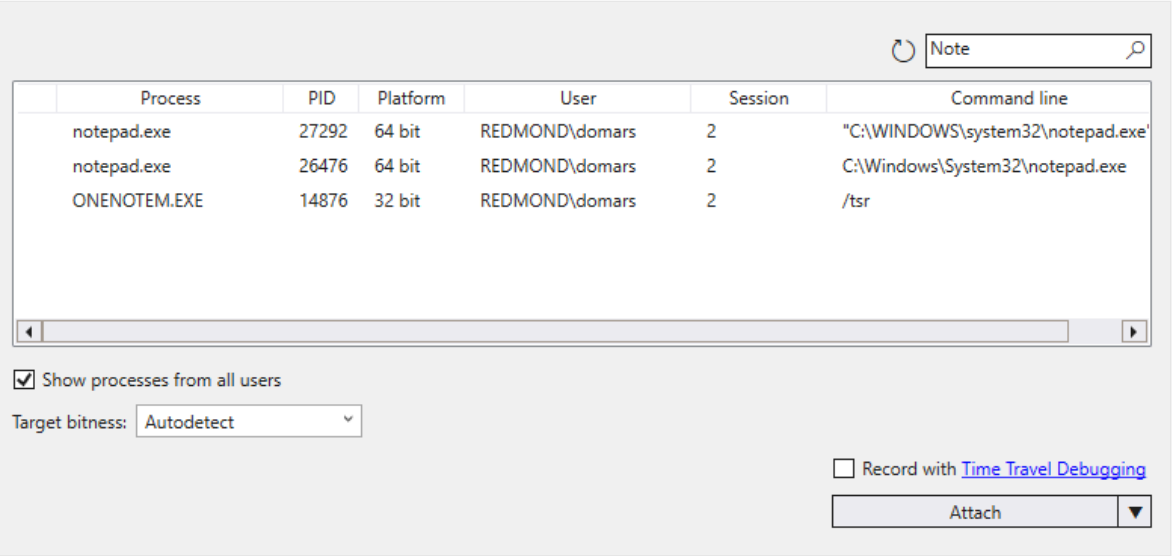



- **Запуск пакетов приложений** — отладка универсального приложения или фоновой задачи в одном щелчке мыши.

  Дополнительные сведения см. в разделе "Запуск пакета приложения".

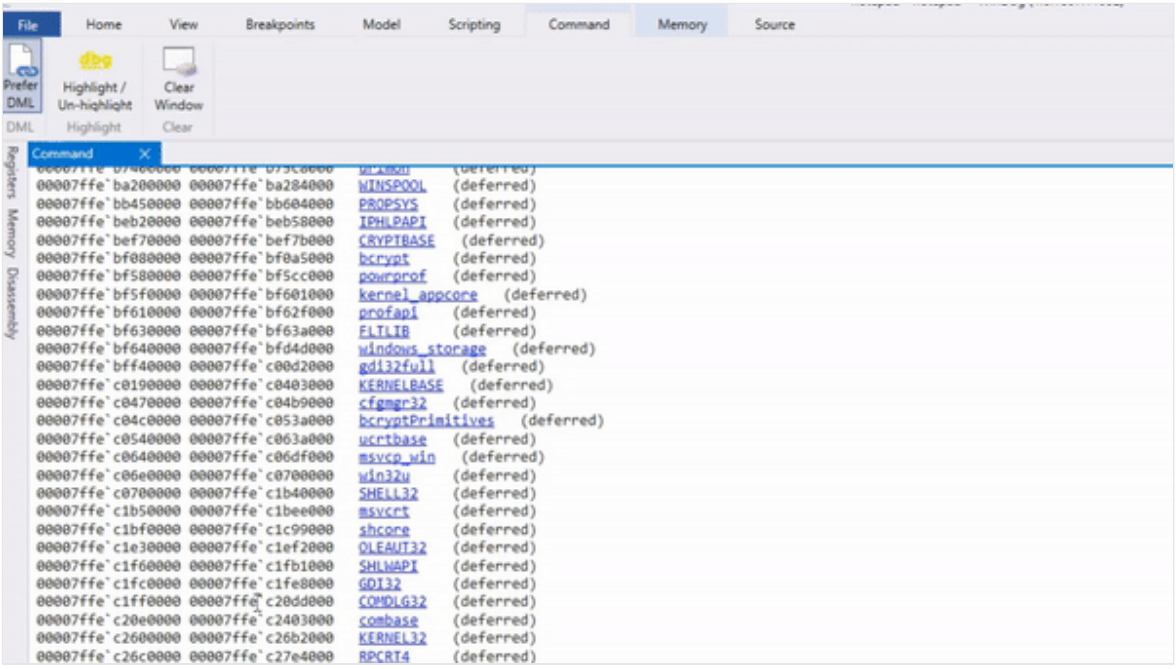


- **Присоединение к процессу** . Новое представление присоединения предоставляет подробное представление о выполняемых процессах, упрощенной настройке и поддержке поиска.




# Улучшенные окна инструментов

- **Команда** — в командном окне улучшена поддержка DML, выделение текста, поиск (включая Regex).




- Источник — окно **исходного** кода предоставляет выделение синтаксиса и другие общие улучшения, аналогичные большинству современных текстовых редакторов.

DisplayGreeting.cpp

```cpp
#include "stdafx.h"
#include <array>
#include <stdio.h>
#include <string.h>

void GetCppConGreeting(wchar_t* buffer, size_t size)
{
    wchar_t const* const message = L"HELLO FROM THE WINDBG TEAM. GOOD

    wcscpy_s(buffer, size, message);
}

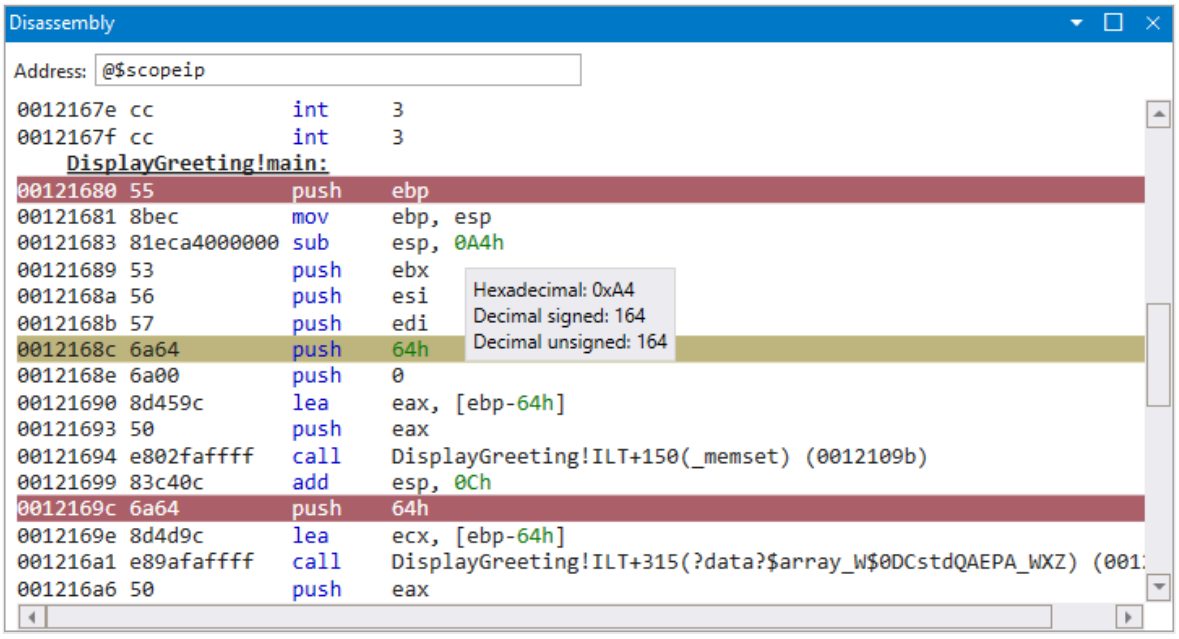

int main()
{
    std::array <wchar_t, 50> greeting{};

    GetCppConGreeting(greeting.data(), sizeof(greeting));

    wprintf(L"%ls\n", greeting.data());

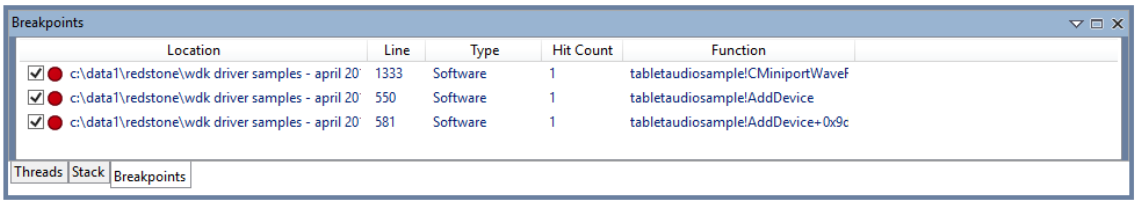
}
```

- **Дизассембли** - окно дизассемблера также улучшено, подсветка текущей инструкции остается там, где она находится при прокрутке.



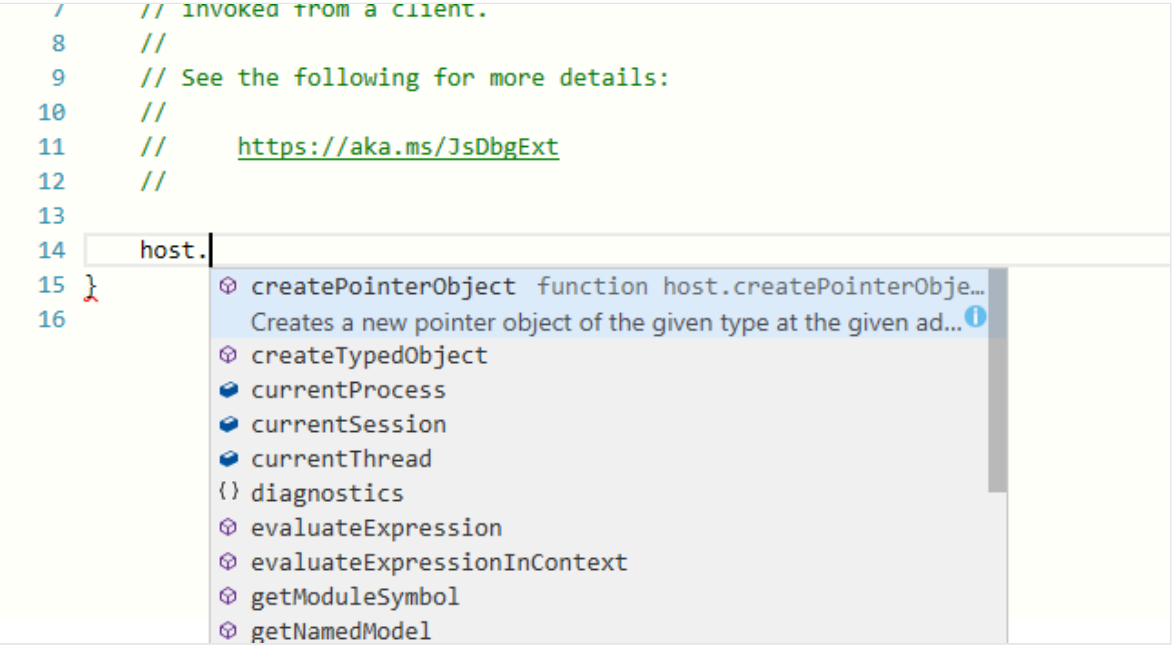

- **Точки останова** — окно точек останова показывает все ваши текущие точки останова, переключатель с одним щелчком и счетчик попаданий.

  Дополнительные сведения см. в точках останова.



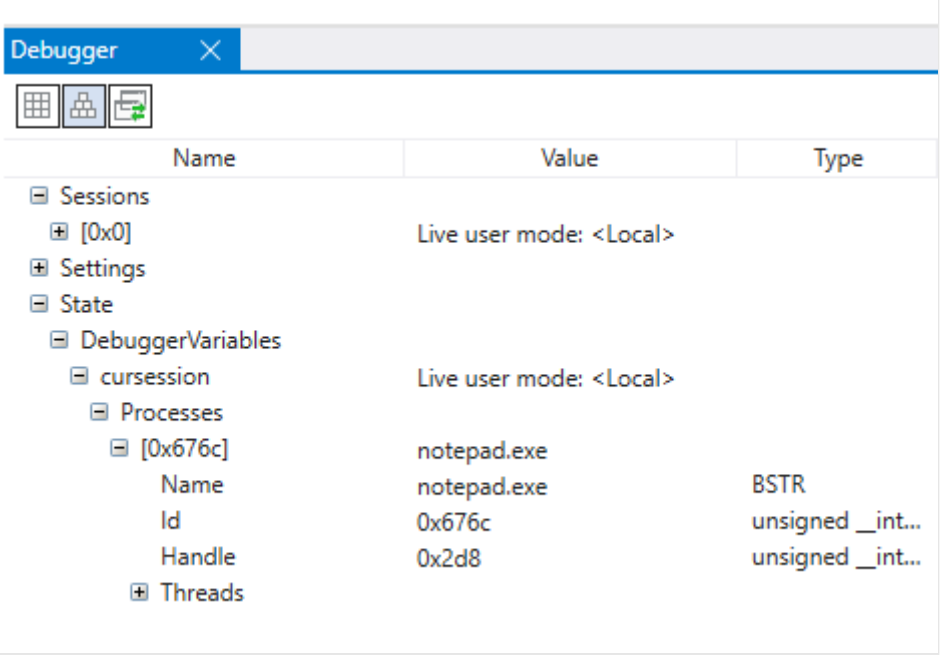
- **Скриптирование** — новое окно сценариев упрощает разработку расширений JavaScript и NatVis с выделением ошибок и IntelliSense.

  Дополнительные сведения см. в статье WinDbg — Scripting.




- **Модель данных** — окно модели предоставляет расширяемую и просматриваемую версию `dx` и `dx -g` позволяет создавать мощные таблицы на вершине запросов NatVis, JavaScript и LINQ.

  Дополнительные сведения см. в статье WinDbg — модель данных.




- **Локальные и контрольные окна**—локальные и контрольные окна основаны на модели данных, используемой командой `dx`. Это означает, что они пользуются теми же функциями, что и другие окна модели данных.

- **Память** — окно памяти имеет выделение и улучшенную прокрутку.
- **Журналы** — это внутренние журналы WinDbg. Его можно просмотреть для устранения неполадок или мониторинга длительных команд.

# Предоставление отзывов

Ваши отзывы помогают команде управлять разработкой и приоритетами функций WinDbg.

Чтобы сообщить о любых ошибках или предложить новую функцию, нажмите кнопку обратной связи на ленте, чтобы перейти на страницу GitHub , где можно отправить новую проблему.

# Другие ресурсы

- Чтобы узнать о новых функциях в последнем выпуске, см. примечания к выпуску.
- Ознакомьтесь с этими разделами, чтобы установить и настроить WinDbg:
  - WinDbg — параметры запуска командной строки
  - WinDbg — параметры и рабочие области
  - WinDbg — сочетания клавиш
- В этих разделах описывается, как подключиться к среде, которую требуется выполнить отладку:
  - WinDbg — запуск сеанса в пользовательском режиме
  - WinDbg — запуск сеанса режима ядра
- Просмотрите эти эпизоды программы Defrag Tools , чтобы увидеть WinDbg в действии:
  - Defrag Tools #182 — Тим, Чад и Энди обсуждают основы WinDbg и некоторые функции.
  - Defrag Tools #183 - Ник, Тим и Чад используют WinDbg и переходят к быстрой демонстрации.
  - Defrag Tools #184 — Билл и Эндрю дают пошаговую инструкцию по скриптовым функциям в WinDbg.
  - Инструменты дефрагментации #185 — Джеймс и Иветт предоставляют вводные сведения об отладке временных путешествий.
  - Defrag Tools #186 — Джеймс и JCAB рассматривают продвинутую отладку путешествий во времени.

- Дополнительные советы и рекомендации можно найти в архиве блога WinDbg.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Заметки о выпуске

01.07.2025

Логотип WinDbg с увеличивающимся стеклом, проверяющим биты.

В этом разделе содержатся сведения о новых возможностях WinDbg. Более ранние версии были выпущены как *Предварительная версия* WinDbg.

## 1.2506.12002.1

### Новые возможности

WinDbg теперь поставляется с отладчиками в режиме консоли (cdb.exe/ ntsd64.exe / kd.exe)! Чтобы различать двоичные файлы друг от друга и не мешать существующим копиям инструментов, двоичные файлы включают имя архитектуры:

- cdbARM64.exe / cdbX64.exe / cdbX86.exe
- kdARM64.exe / kdX64.exe / kdX86.exe
- ntsdARM64.exe / ntsdX64.exe / ntsdX86.exe

Дополнительные возможности:

- Заголовок окна инструментов модели данных можно настроить с помощью команды $OpenDmoToolwindow, а также изменить его с помощью контекстного меню заголовка.
- Новый параметр отладчика позволяет включить параллельное скачивание символов через HTTP. Этот параметр включен по умолчанию.

### Исправления ошибок

- Исправлен способ сохранения целевого пути для файлов дампа, открытых с помощью командной строки (-z) для успешного открытия из меню "Последние".
- Исправлен сбой в окне редактора скриптов.
- Исправлено, как WinDbg обрабатывает несогласованные окончания строк в исходных файлах, открытых в окне источника, чтобы обеспечить более согласованное представление исходных файлов.
- Исправлена проблема, из-за которой учетные данные не обновлялись для серверов символов HTTP при использовании новой функции параллельного скачивания.

- Исправлена проблема, из-за которой свойства объектов модели данных не перечислялись должным образом в расширениях JavaScript.
- Исправлена проблема, из-за которой текст столбца исчез при переупорядочении с помощью щелчка и перетаскивания.
- Исправлена проблема, из-за которой настройка фильтрации событий с помощью параметров командной строки не применялась к целевому объекту отладки при запуске.
- Исправлена проблема, из-за которой иногда при откреплении окна инструментов появлялось пустое окно.
- Исправлена проблема, из-за которой не удалось выполнить отладку мини-модулей эмулированных процессов X64 на компьютерах ARM64.
- Исправлена проблема, из-за которой неоднозначные точки останова вызывали сбой в отладчике.
- Исправлена проблема, из-за которой GDBServer произошел сбой при завершении процесса при отладке динамического процесса Linux.
- Исправлено несколько проблем, связанных с отладкой расширений JavaScript (.scriptdebug), включая лучшее перечисление свойств и исправление проблемы с сбоем.

## Общие улучшения пользовательского интерфейса

- Всплывающее окно предложения команды выравнивается с курсором и отображает только предлагаемые завершения, а не все перед курсором.
- Диалоговое окно запуска целевой ошибки отладки содержит дополнительные сведения, а для открытых сбоев дампа предоставляет гиперссылку на не удалось открыть TSG-файл дампа.
- Заголовки инструментальных окон имеют то же контекстное меню, что и закрепленная вкладка.
- Визуальные улучшения в окнах "Локальные", "Контрольные" и "Модели".
- Отладка временных путешествий (TTD) обновлена до версии 1.11.532. Дополнительные сведения см . в заметках о выпуске TTD.

# 1.2502.25002.0

## Новые возможности

### Командное окно IntelliSense

Командная строка ввода автоматически отображает предлагаемые завершения, когда первый типизированный символ является `.` (точка), `!` (восклицательный знак) или `$` (доллар). Предложения также можно вызвать с помощью сочетания клавиш `Ctrl + Space`

## Отладка моментальных снимков

При локальной отладке **динамических процессов** отладчик может создавать моментальные снимки процесса и переключаться между моментальными снимками и динамическим процессом с помощью значков на вкладке "Расширения". Аналогичным образом отладчик может подключиться к моментальному снимку процесса, не приостанавливая его с помощью `snapshot:pid=<process pid in hex>` строки протокола в разделе "Подключение к удаленному отладчику".

## Новые API модели данных

- `Debugger.Utility.Symbols.GetTypeInformation`

Он вернет сведения о типе, связанные с любым входным объектом. Существует необязательный аргумент, который позволяет автоматически разыменование указателей на один уровень, если вы действительно хотите получить тип под любым указателем (или ссылкой).

- `Debugger.Utility.Control.CreateStepFilter(<Type>, <Pattern>)`

Создает новый фильтр шагов. Он находится в этом пространстве имен, рядом с API для создания точек останова.

- `Debugger.State.StepFilters`

Возвращает список фильтров шагов с идентификатором свойств, описанием, типом, шаблоном и IsEnabled. Есть метод .Remove() для объектов, фильтрующих шаги.

## Улучшен опыт отладки CLR без clrcomposition.dll

Распространяемый компонент `dbgeng.dll` теперь поддерживает улучшенный обход стека для кадров CLR без необходимости присутствия `clrcomposition.dll`.

## Случайные улучшения

- Улучшена поддержка отладки ядра Linux (особенно для ядер 6.11+ )
- Поддержка сжатых страниц KDUMP ZSTD

## Исправления ошибок

- Обновите SOS до версии 9.0.557512.
- Исправлена регрессия производительности при проверке отзыва подписей при загрузке msdia140.dll.
- Исправлен сбой при чтении контекста XSTATE.
- Исправлено выделение отключенной точки останова в окне дизассемблирования для различия с включенной точкой останова.
- Исправлен сбой при закрытии плавающего окна инструмента
- Исправлена проблема, из-за которой иногда текущая позиция окна временных шкал не заполнялась после открытия недавнего слежения.
- Исправлена проблема, из-за которой окна дизассемблера и памяти иногда не обновлялись после ввода нового значения.
- Исправлена цветовая схема строки заголовка в темной теме.
- Исправлена презентация отложенных точек останова в окне точек останова, где расположение будет отображаться 0x0 вместо исходного выражения, введенного при создании. Кроме того, он имеет отдельный значок, указывающий, что он не привязан.
- Исправлена проблема в locals/Watch, когда щелкнуть правой кнопкой мыши видимый элемент приведет к прокрутке представления вверх.
- Исправлены проблемы с natvis, см. исходный отчет об ошибках.
- Исправлен спам отладчика kd с сообщением `Error getting port connector information`, см. исходный отчет об ошибке.
- Исправлена проблема с `gc`, которая автоматически инициировала условные точки останова, если условие было ложным.
- Регистры YMM и ZMM теперь можно просматривать при отладке трассировки TTD.

## Исправления доступности

- Улучшенная навигация по клавиатуре для локальных, контрольных точек и окон точек останова.
- Улучшена удобочитаемость окон ленты, локальных параметров, контрольных точек и точек останова.
- Улучшенная удобство использования средства чтения с экрана в окнах "Локальные", "Контрольные точки" и "Точки останова"
- При редактировании элементов в секциях "Локальные", "Просмотр" и "Точки останова" будут, по возможности, поддерживаться предыдущие фокус клавиатуры, выбор и прокрутка.
- Улучшена общая поддержка масштабирования текста Windows

## Общие улучшения пользовательского интерфейса

- Добавлена команда для перемещения фокуса между текстовым полем ввода команд в область вывода команд в качестве режима переключения метки (например, командной строки). Ярлык `Ctrl + M`
- Добавлен элемент контекстного меню «Перейти к дизассемблированию» в окно точек прерывания.
- Добавлено контекстное меню в окно дизассемблирования для перехода к инструкции, установки указателя инструкции, перехода к источнику, включения/отключения точки останова, переключения состояния точки останова.

## Отладка временных путешествий (TTD)

- Обновлено до версии 1.11.481.
- Смотрите заметки о выпуске TTD

# 1.2410.11001.0

## Изменение, нарушающее совместимость

- dbghelp.dll динамически связывается с msdia140.dll. Это должно влиять только на средства, распространяющие dbghelp.dll, не включая msdia140.dll. Обратите внимание, что msdia140.dll входит в пакеты nuget, которые включают dbghelp.dll.

## Исправления ошибок

- Исправлена проблема, из-за которой псевдонимы функции коллекции расширений не были привязаны к триггерам загрузки
- Не делайте хранилище символов неактивным при получении ошибки ERROR_INTERNET_SECURITY_CHANNEL
- Исправлена проблема, из-за которой обратная косая черта не могла быть введена в командном окне для макетов клавиатуры, отличных от английского языка.
- Исправлена проблема, из-за которой текстовое поле редактирования в точках останова, локальных и контрольных окнах не прокручивается до курсора, если текст больше, чем размещенный в нем столбец.
- Для дампов CAB или ZIP приоритеты *.dmp (один период) по сравнению с *.*.dmp (двойные периоды)

## Исправления доступности

- Изменение размера столбцов с помощью клавиатуры было исправлено для окон "Контрольные" или "Локальные".
- Улучшенная навигация по клавиатуре в WinDbgX.

# 1.2409.17001.0

## Исправления специальных возможностей

Мы исправили несколько проблем при взаимодействии с WinDbg с помощью средства чтения с экрана.

## Исправления ошибок

- Исправлена проблема, из-за которой удаление WinDbgX не позволяет запускать классическую версию WinDbg.
- Исправлена проблема, из-за которой WinDbgX изменит размер после появления во время запуска.

## Отладка временных путешествий (TTD)

- Обновлено до версии 1.11.429.
- Дополнительные сведения см . в заметках о выпуске TTD.

# 1.2407.24003.0

## Новые возможности

### Новый поставщик скриптов JavaScript

Теперь мы предоставляем новый поставщик JavaScript с обратной совместимостью на основе V8. В результате этого узлы Arm64 теперь могут использовать расширения JavaScript. Пользовательский интерфейс arm64 теперь по умолчанию используется для ядра arm64.

Новый поставщик включен по умолчанию для всех архитектур. В x86 и x64 его можно включить или отключить с помощью `.veighton` команд и `.veightoff` команд. В будущем эти команды и устаревший поставщик JavaScript будут удалены.

## Расширяемость пользовательского интерфейса через коллекцию расширений и модель данных

Расширения отладчика с манифестами коллекции расширений теперь могут добавлять значки на вкладку "Расширения" ленты WinDbg. Щелкнув эти значки, будет запускаться визуализация заданного запроса модели данных. Это означает расширяемость осуществляется путем добавления *представлений* в манифест коллекции расширений. Представление привязывает связывание имени и значка с запросом модели данных и типом представления. Документацию по манифесту коллекции расширений см. в документации по манифесту коллекции расширений

## Добавлены параметры исключения в триггер загрузки исключений коллекции расширений

Теперь расширение коллекции можно условно загрузить на основе определенных значений параметров исключения. Дополнительные сведения см . в документации по манифесту коллекции расширений.

## Параметр конфигурации для никогда не скачивать определенные символы

Теперь можно пропустить скачивание символов для указанных изображений.

`Symbols` В разделе `DbgX.xml` файла конфигурации можно добавить список символов:

XML

```
<Namespace Name="DisabledSymbolsDownload">
  <Setting Name="MSEdge.dll" Type="VT_BSTR" Value=""></Setting>
</Namespace>
```

В отладчике можно запустить `dx Debugger.Settings.Symbols.DisabledSymbolsDownload` , чтобы просмотреть текущий список символов, которые не загружаются.

Вы также можете изменить список непосредственно из отладчика следующим образом:

```
dx @$curprocess.Modules[1].DisableSymbolDownload()
dx @$curprocess.Modules[1].EnableSymbolDownload()
```

## Новая команда для отображения сведений SourceLink

Чтобы помочь в отладке проблем с загрузкой источников на основе внедренного SourceLink, теперь можно использовать команду !lmsourcesinfo module для отображения содержимого SourceLink, которое будет использовать отладчик.

## Общие улучшения пользовательского интерфейса

Мы сделали некоторые небольшие улучшения в окнах, таких как окно модулей или окно стека:

- Теперь можно сортировать представления сетки, щелкнув столбец заголовка.
- Доступны дополнительные действия правой кнопкой мыши.
- В окне стека отображаются кадры, которые встраиваются при необходимости.

## Исправления ошибок

- Исправлена проблема, из-за которой файлы PDB размером более 4 ГБ не загружались по протоколу HTTP.
  - Edge является примером такого PDB.
- Исправлена проблема кэширования в поставщике учетных данных отладчика (теперь она будет кэшировать учетные данные на сайте).
- Исправлены ошибки при оценке триггеров загрузки коллекции расширений.
- Исправлена ошибка, из-за `!address` которой может произойти сбой.
- `!heap -a -a` должен работать сейчас.

## TTD

См . заметки о выпуске TTD версии 1.11.410.

# 1.2402.24001.0

## Новые возможности

### Отладка Live Linux

Теперь вы можете выполнять прямую отладку процесса Linux. Дополнительные сведения см. в следующих статьях:

Отладка удаленного процесса Linux

Символы и источники Linux

Клиент WinDbg в обозримом будущем по-прежнему потребует Windows, так как WinDbg не работает в Linux.

## Запись подмножества выполнения с помощью отладки времени перемещения

Теперь можно указать список модулей для записи, чтобы сосредоточиться на TTD на модулях, которые вам нужны. Это может значительно снизить затраты на запись. Чтобы использовать этот простой флажок "Запись подмножества выполнения" в диалоговом окне "Настройка и запись", а затем укажите модули, которые требуется записать. (см . известные проблемы ниже).

Для более точного управления можно точно записать программу с помощью нового API записи в процессе. Узнайте, как с помощью примера кода и документации.

## Окно компактных регистров

Новое окно регистрации выглядит так же, как выходные данные команды `r` . Обратите внимание, что редактирование из нового окна регистрации еще не поддерживается.

## Новые параметры для `e` (ввод значений) в команды памяти

Команды `e` (ввод значений) в память теперь поддерживают физические адреса так же, как `d` команды (отображаемая память). Обратите внимание, что эти параметры поддерживаются только в режиме ядра:

Развернуть таблицу

| Вариант | Описание |
|---|---|
| `/p` | Использует адреса физической памяти для отображения. Диапазон, указанный диапазоном, будет взят из физической памяти, а не виртуальной памяти. |
| `/p[c]` | Аналогично тому, что `/p` кэшированная память будет считываться. Квадратные скобки вокруг с должны быть включены. |
| `/p[uc]` | То же самое `/p`, что, за исключением того, что некэшированная память будет считываться. Квадратные скобки вокруг uc должны быть включены. |
| `/p[wc]` | Так же, как `/p` и в случае с памятью, объединенной записью, будет считываться. Квадратные скобки вокруг wc должны быть включены. |

Дополнительные сведения см. в разделе e, ea, eb, ed, eD, ef, ep, eq, eu, ew, eza (ВВОД значений).

## Исправления ошибок

- Поддержка дедемангинга имени Rust в отладке Linux
- Улучшения отладки СРЕДЫ CLR
  - Добавление возможности принудительной принудительной передачи кадров среды CLR в смешанном режиме
    - Добавлен параметр `ForceMixedModeStackWalker` , добавленный `config.xml` или `DbgX.xml`.
    - Добавлена `!forceclrmixedmodeunwind` команда для одноуровневых расследований или если вы не можете изменить параметр конфигурации.
- Общие улучшения отладки СРЕДЫ CLR в Linux
- Устранение ряда проблем, связанных с LinuxKernel.js и переменными на ЦП
  - Добавлена `!runq` команда для LinuxKernel.js, чтобы иметь возможность дампа очередей запуска планировщика ядра на ЦП (аналогичной команде служебной программы аварийного `runq` завершения Linux).
  - Исправлены различные команды в LinuxKernel.js для соответствия изменениям ядра: в более поздних ядрах дерево таймера использует кэшированные узлы RB вместо необработанных узлов RB; в более поздних ядрах список all_bdevs исчез, поэтому блоковые устройства теперь переходят из суперблока (blockdev_superblock)
- Различные исправления natvis для работы более поздних версий Visual Studio
- Предпочитать sourcelink и устаревшую sourceinfo
  - Добавлена `!lmsourcesinfo <module>` команда для отображения потоков исходной связи из PDB (чтобы помочь в отладке проблем с sourcelink).

## Известные проблемы

При указании списка модулей для записи с помощью TTD указание нескольких модулей не работает в этом выпуске.

# 1.2308.2002.0

Многие исправления ошибок и небольшие улучшения.

# 1.2306.12001.0

Многие исправления ошибок и небольшие улучшения.

# 1.2303.30001.0

> **Примечание**
>
> Отладчик был переименован из *WinDbg Preview* в *WinDbg*.
>
> Устаревший отладчик, выпущенный с помощью средств отладки для Windows, будет ссылаться как *WinDbg (классическая версия)*.

## Отладка временных путешествий в ARM64

Кроме того, WinDbg выполняется изначально в ARM64, мы также включили отладку времени (TTD) собственных процессов ARM64. Теперь вы можете записывать собственные процессы ARM64 (и процессы x86 и ARM32) и выполнять их отладку даже на компьютере x64. Поддержка процессов с поддержкой x64 или x64 (ARM64EC) на устройствах ARM64 пока недоступна.

Общая производительность TTD также была улучшена, при этом затраты на запись значительно сократились для процессов, использующих общие инструкции SIMD. В общей сложности были оптимизированы 65 новых инструкций, начиная от MMX до SSE 4.2, AVX и AVX2.

## Неоднозначное разрешение точки останова

Неоднозначные точки останова позволяют отладчику задавать точки останова в определенных сценариях, когда выражение точки останова разрешается в несколько расположений. Дополнительные сведения см. в разделе "Неоднозначное разрешение точек останова".

# 1.2107.13001.0

## Поиск регулярных выражений

Поиск WinDbg Preview теперь включает возможность поиска с помощью регулярных выражений — Regex. Включите regex в окне поиска, выполнив клавиши CTRL+F, а затем переключите кнопку, помеченную `.*` рядом с полем поиска.

## Ограниченный режим

Режим ограниченного доступа — это необязательный параметр, который может ограничить тип сеансов отладки WinDbg Preview, может запускаться только к удаленным сеансам отладки и файлам дампа. Дополнительные сведения см. в статье WinDbg — ограниченный режим.

## Поддержка теневой переменной

В предыдущих версиях WinDbg вызывает ошибки "Неоднозначный символ" при попытке оценить (?)) или отобразить переменную (dx), которая делится своим именем с другой переменной в области. Windbg теперь будет диамбигуировать переменные, которые имеют то же имя, добавляя `@n` к имени переменной. Например: foo, foo`@0` `@1`

Это изменение также будет отражено в окне "Локальные". Ранее отображается только ближайшая переменная в области.

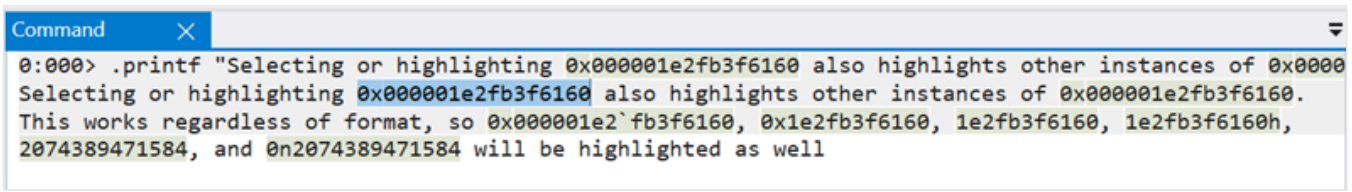
# 1.2104.13002.0

## Выбор и поиск смарт-номеров

Удобная функция повышения производительности WinDbg Preview — это возможность обнаруживать шаблоны в командном окне. Для этого выделите любой текст и все остальные экземпляры этого текста. Так как это выделение полезно при просмотре шаблонов памяти, теперь оно также будет выделять эквивалентные числа в других радиксах, независимо от формата числа в шестнадцатеричном, десятичном или научном нотации. Дополнительные сведения о схемах нумерирования см. в разделе n (Set Number Base).

Пример:

При выборе `0x000001e2fb3f6160` все остальные экземпляры выделяются независимо от формата.




Снимок экрана: командное окно WinDbg с выделенным несколькими форматами чисел.

Эта функция также работает с полувременного выделения. Ctrl+ Double Click on a number, чтобы выделить все его экземпляры. Вы можете отслеживать несколько важных чисел в

журнале команд таким образом (чтобы очистить выделение, ctrl+ Double Click on the number again). Наконец, эта функция также работает при поиске чисел с помощью CTRL+F.

## Расширенный доступ к исходному коду

Команда исходного пути .srcpath, Lsrcpath (Set Source Path) была обновлена, чтобы включить новый тег — *DebugInfoD*. Дополнительные сведения см. в статье "Расширенный доступ к исходному коду".

## Состояния узлов и гостевых процессов WOW в модели данных

При отладке 32-разрядного процесса WOW из 64-разрядного контекста пользователи теперь могут получить доступ к состояниям узла и гостя в модели данных.

32-разрядные примеры состояния гостевого состояния:

```
dx @$curprocess.Environment
```

```
dx @$curthread.Environment
```

```
dx @$curthread.Stack
```

64-разрядные примеры состояния узла:

```
dx @$curprocess.NativeEnvironment
```

```
dx @$curthread.NativeEnvironment
```

```
dx @$curthread.NativeStack
```

### Улучшения отладки JavaScript

JavaScript, загруженный в пользовательский интерфейс, теперь можно напрямую отлаживать в консоли с помощью команды scriptdebug. Дополнительные сведения см. в статье "Скрипты отладчика JavaScript" — отладка JavaScript.

## Улучшения в специальных возможностях

С помощью Предварительной версии WinDbg мы стремимся создать отладчик, включающий инженеров с ограниченными возможностями, мы постоянно совершенствуем специальные возможности. Были сделаны следующие улучшения.

- Ссылки на командное окно теперь можно щелкнуть с помощью клавиатуры (SHIFT+ВВОД)
- Улучшенная навигация по клавиатуре главного меню
- Улучшенная навигация по клавиатуре ленты
- Повышенная контрастность элементов пользовательского интерфейса

## Новый тип доступа к данным Overwrite для отладчика временных путешествий

Отладчик по времени (TTD) теперь предоставляет тип доступа к данным Overwrite. Запросы памяти, такие как `dx @$cursession.TTD.Memory()` теперь, имеют дополнительный столбец со старыми значениями записей.

## Другие исправления, улучшения и обновления

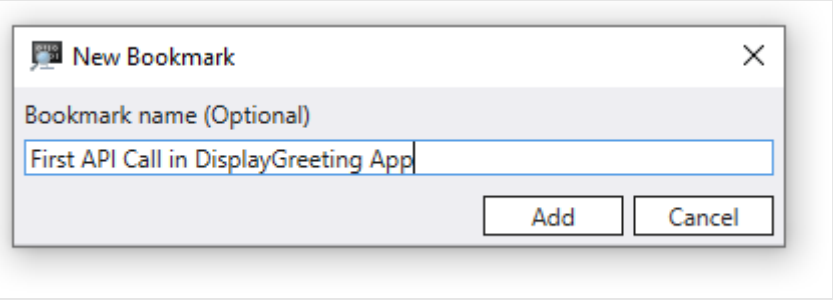
- Добавлена функция для автоматического обнаружения и применения обходного решения для аппаратного решения USB 3.1, когда узел отладки ядра и целевой объект являются контроллерами USB 3.1.
- Добавлено новое сочетание клавиш для пользовательского интерфейса: Ctrl+Shift+щелчок по ссылке DML скопирует её в буфер обмена.

# 1.0.2007.01003

**Закладки временной шкалы**

Закладка важных позиций поездок по времени в WinDbg вместо ручного копирования позиции в блокнот. Закладки упрощают просмотр разных позиций трассировки относительно других событий и их анимацию.

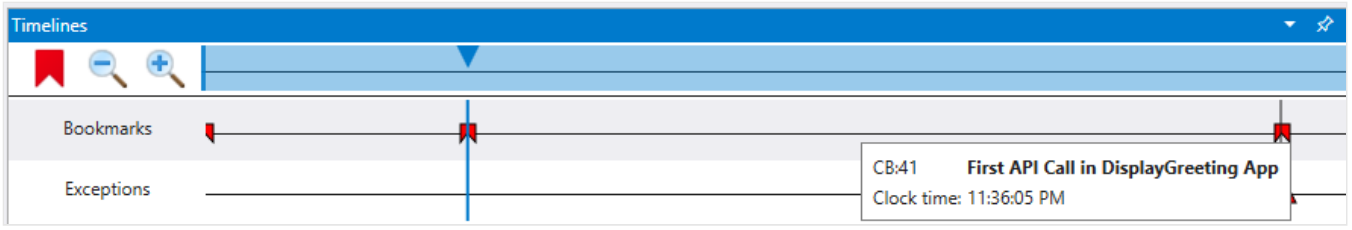
Вы можете указать описательное имя для закладок.



Снимок экрана: диалоговое окно создания закладки с примером имени первого вызова API в приложении "Приветствие".

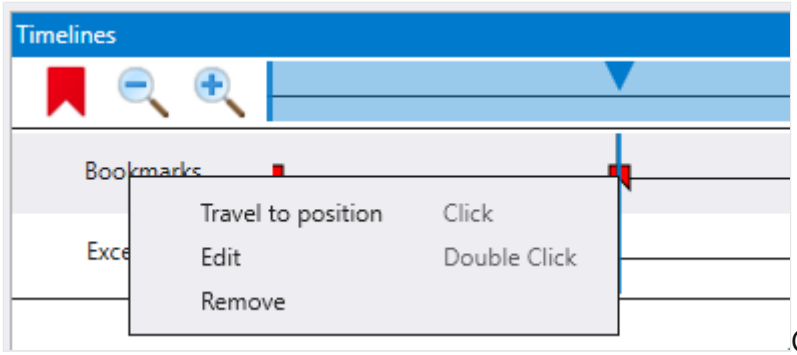
Доступ к закладкам через окно временной шкалы, доступное в *режиме просмотра* > *временной шкалы*. При наведении указателя мыши на закладку будет отображаться имя

закладки.




Снимок экрана: временная шкала WinDbg с тремя закладками и подсказкой с именем закладки.

Вы можете выбрать и сохранить закладку (или щелкнуть правой кнопкой мыши), чтобы перейти к этой позиции, переименовать или удалить закладку.

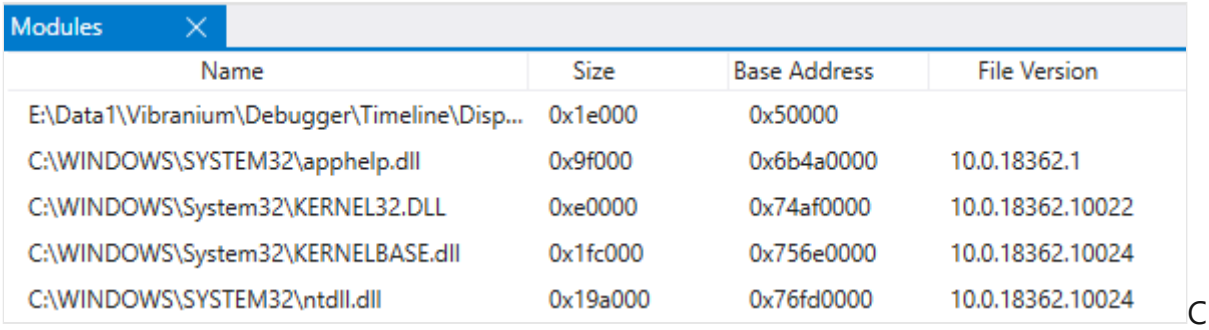



Снимок экрана: всплывающее меню закладки правой кнопкой мыши с параметрами перемещения по расположению, редактированию и удалению.

**Окно модулей**

В новом окне отображаются модули и связанные с ними сведения, доступные на ленте "Вид". В нем отображается следующее:

- Имя модуля, включая расположение пути
- Размер в байтах загруженного модуля
- Базовый адрес, по которому загружается модуль
- Версия файла




Снимок экрана: окно представления модулей WinDbg с пятью перечисленными модулями.

**Имена и описания потоков, доступные в динамической отладке**

Имена потоков, заданные из SetThreadDescription, теперь доступны при отладке в реальном режиме пользователя. Имена потоков доступны с помощью команды "~" или модели данных отладчика.

dbgconsole

```
0:000> ~
   0  Id: 53a0.5ffc Suspend: 1 Teb: 000000b1`db1ed000 Unfrozen "Hello world!"
   7  Id: 53a0.9114 Suspend: 1 Teb: 000000b1`db1ef000 Unfrozen
   8  Id: 53a0.2cc4 Suspend: 1 Teb: 000000b1`db1f1000 Unfrozen
   9  Id: 53a0.5c40 Suspend: 1 Teb: 000000b1`db1f3000 Unfrozen

0:000> dx @$curthread
@$curthread                 : ConsoleTestApp!ILT+25(mainCRTStartup)
(00007ff7`fac7101e)  [Switch To]
    Id               : 0x5ffc
    Name             : Hello world!
    Stack
    Registers
    Environment
```

**Поддержка переносимого PDB**

Добавлена поддержка переносимых PDB. В формате переносимой PDB (база данных программы) описывается кодировка отладчика информации, созданной компиляторами языков общей инфраструктуры языка (CLI) и используемыми отладчиками и другими средствами. Дополнительные сведения см. в статье "Переносимые символы PDB".

**Другие изменения и исправления ошибок**

- WinDbg теперь поддерживает отладку дампа ядра AMD64 и Linux.
- Улучшения записи времени и другие исправления.

# 1.0.1912.11001

**Временная шкала** TTD — мы добавили новое окно, отображающее визуальное представление важных событий в трассировке: исключения, точки останова, вызовы функций и доступ к памяти. Временные шкалы автоматически открываются и отображаются исключения (при наличии) и точек останова. Дополнительные сведения см. в разделе "Предварительная версия WinDbg — временная шкала".

**Переключился на хром окна по умолчанию — пользовательский хром** окна, который мы использовали, в то время как более красивый, вызывал некоторые проблемы масштабирования и изменения размера для заметного количества людей, поэтому мы решили удалить его на время.

**Меню файлов улучшена навигация** по клавиатуре. Меню файлов теперь гораздо проще перемещаться с помощью клавиатуры.

**Другие изменения и исправления ошибок**

- Теперь окно стека и окно локальных переменных будет отключено при запуске целевого объекта и не будет отображать сообщение "Неопределенная ошибка" при отсутствии целевого объекта.
- Добавлен столбец "Службы" в диалоговое окно присоединения, чтобы легко найти, какие службы выполняются.
- Исправлена ошибка, из-за которой обнаружение архитектуры не работало при запуске приложений с аргументами.
- Окно дизассемблирования улучшило дизассембли при загрузке частных символов.
- jsprovider.dll теперь загружается автоматически, поэтому мы удалили кнопку Load JSProvider на ленте скрипта.

# 1.0.1908.30002

**Улучшения для объектов** - вызовов TTD Теперь включают имена параметров, типы и значения. При запросе между трассировками для вызовов функций можно получить полностью типизированные параметры и их значения, что упрощает фильтрацию результатов по параметрам.

**Поддержка Open Enclave** — Предварительная версия WinDbg теперь может выполнять отладку приложений Open Enclave (OE) для получения дополнительных сведений, см. в разделе "Отладка Open Enclave".

**Дампы ELF Core** - В рамках поддержки Open Enclave, WinDbg может открывать дампы ядра ELF и двоичные файлы, а также символы DWARF (DWARF 5 в настоящее время не поддерживаются) как из анклавов, так и приложений Linux. При открытии основного дампа из приложения, отличного от Windows, базовые окна и команды должны работать правильно, но большинство расширений и команд, относящихся к Windows, не будут работать. ФАЙЛЫ ELF и DWARF будут загружены с серверов символов после ключевых соглашений, определенных здесь . Анклава являются единственным поддерживаемым сценарием, но мы открыты для отзывов о открытии других дампов ядра Linux.

**Изменение** формата файла TTD . Мы внесли основное обновление в формат файла для трассировок TTD, которые прерывают совместимость вперед. Предыдущие версии WinDbg Preview не смогут открывать трассировки, записанные с помощью этих (и будущих) версий WinDbg Preview, но эти (и будущие) версии смогут открывать как новые, так и старые трассировки.

**Другие изменения**

- Теперь TTD будет использовать 64-разрядный модуль для индексирования и соответствующей биты обработчика отладчика для воспроизведения, чтобы свести к минимуму потенциальные проблемы с памятью при индексировании и проблемах SOS при повторе.
- При запуске "dx" без параметров теперь отображается корневое пространство имен для упрощения просмотра.
- Теперь можно изменить расположение символа по умолчанию и исходного кэша с помощью меню параметров.
- Улучшенная поддержка записи AVX-512 (запись AVX-512 приведет к большему, чем обычное замедление).
- Мы включили автономное лицензирование.

# 1.0.1905.12001

**Улучшения устранения** ошибок SymSetDiaSession . Исправление в прошлом месяце, чтобы устранить ошибку, вызванную приложениями, внедряющими DbgHelp в наш процесс, по-прежнему не работает в некоторых сценариях. Мы улучшили эту ошибку и продолжаем отслеживать отзывы об этой ошибке.

**Настройка** цвета акцента — для многих сценариев требуется несколько экземпляров OpenDbg, а перемещение между ними может быть запутанным и занять некоторое время, чтобы выяснить, какой из них является "правильным". Мы добавили возможность изменить цвет синего акцента, чтобы помочь визуально различать сеансы и сделать переключение между ними проще.

Просто выберите **ленту "Вид**" и выберите параметр цвета акцента в последнем разделе. При запуске будущих сеансов из последних целевых объектов цвет акцента будет сохранен в рамках рабочей области целевого объекта.

**Улучшения** исходной маркеризации. Теперь в окне источника есть базовая поддержка маркеризации исходных файлов Rust и C++ SEH __try/__except/__finally/__leave.

**Улучшения** корутин — улучшена поддержка локальных переменных соутина и некоторых оптимизированных переменных.

**Параметр** символа и кэша источника по умолчанию— добавлен параметр в меню **параметров** отладки, чтобы изменить расположение кэша для символов. **Примечание.** Существует известная проблема, из-за которую это пустое приведет к сбою загрузки источника. Мы добавим проверку, чтобы предотвратить это в будущем выпуске.

**-pv** исправления — исправлена ошибка, которая может препятствовать -pv (неинвазивное подключение) работать в некоторых обстоятельствах.

# 1.0.1904.18001

**Исправление ошибки** SymSetDiaSession. У нас были отчеты за некоторое время ошибки, которая предотвращает запуск WinDbg Preview в некоторых ситуациях. Существует несколько внешних приложений, которые пытаются внедрить версию DbgHelp в наш процесс перед загрузкой. Некоторые из них используют версию DbgHelp с отсутствующими функциями, что приводит к этой ошибке при попытке использовать эти функции. Мы добавили исправление для этого и будет отслеживаться, если существуют все еще сценарии, в которых она происходит.

**Элемент управления** шрифтом — мы добавили параметры для управления размером шрифта и шрифта. Существует два разных параметра, один для текстовых окон (монопространствовые окна, такие как дизассембли, источник, команда и т. д.) и один для окон инструментов (локальные, стек и т. д.). Есть еще несколько областей, которые не влияют на эти варианты, которые мы будем обновлять в будущем.

**Улучшение** выделения. Постоянное выделение текста в командном окне теперь также будет выделять текст в окнах источника и заметок.

**Улучшения** загрузки источника. Мы изменили способ загрузки исходных файлов. Ранее при открытии исходного файла операции обработчика, такие как выполнение дополнительных команд, не были возможными или непредсказуемы. Мы изменили место загрузки, чтобы повысить параллелизм и более надежную отмену операций открытия источника.

Другие изменения и исправления ошибок:

- Добавлено значение "Перейти к дизассембли" в контекстное меню исходного окна.
- Добавлен флажок "Следовать текущей инструкции" в окне дизассембли.
- Исправлена ошибка, из-за которой окно командной строки выполнялось медленно при выводе большого количества текста.
- Изменены ключи вверх и вниз страницы, чтобы выполнить аналогичные действия Visual Studio.
- При открытии ФАЙЛА ASM в исходном окне теперь будет выделен базовый комментарий, строка и выделение директив

# 1.0.1812.12001

Эта версия включает эти обновления.

**Заголовок модели данных C++ отладчика** — новый заголовок C++ DbgModel.h включен в Windows SDK для расширения модели данных отладчика через C++. Дополнительные сведения см. в обзоре модели данных C++ отладчика. В этом выпуске содержится новое расширение, которое добавляет дополнительные функции стиля API в модель данных отладчика, доступ к которой можно получить с помощью команды DX, JavaScript и нового заголовка DbgModel.h. Это расширение позволяет модели данных включать в себя знания о выполнении сборки и кода через пространство имен Debugger.Utility.Code и локальную файловую систему через пространство имен Debugger.Utility.FileSystem.

**Расширение** искусственных типов с помощью этого нового расширения API у нас есть новый пример на нашем репозитории GitHub здесь - https://github.com/Microsoft/WinDbg-Samples/tree/master/SyntheticTypes. Это расширение JavaScript считывает базовые файлы заголовков C и определяет сведения о искусственном типе для структур и союзов, определенных в заголовке. С помощью команды DX память можно просмотреть структурированным образом, как если бы у вас была PDB с сведениями о типе для этих типов.

Другие изменения и исправления ошибок:

- Предварительная версия WinDbg теперь будет более интеллектуально обрабатывать исходные окна или дизассемблированное окно на переднем плане при пошаговом выполнении.
- Повторно упорядоченное заголовок окна WinDbg Preview, чтобы иметь более важную информацию при отладке ядра.
- Контрастность фона в командном окне должна быть немного заметной.

# 1.0.1810.2001

Эта версия включает эти обновления.

- Диалоговое окно "Новые параметры", доступ к которому осуществляется из меню "Файл" или ленты "Главная".
- Диалоговое окно параметров событий и исключений. Это меню изменяет способ обработки событий и исключений отладчика, эквивалента команд sx или диалоговых фильтров событий WinDbg. Выберите **параметры** на домашней ленте, а затем нажмите кнопку "События и исключения" слева, чтобы управлять ими.
- Улучшен индексатор TTD с более высокой производительностью. Это повышает производительность индексирования файлов трассировки TTD, что делает процесс индексирования гораздо быстрее (от 2x до 10x) при увеличении размера файлов индексов (~50 % меньше). Улучшения perf наиболее заметны для трассировки более 4 ГБ размера или при использовании компьютера со многими ядрами ЦП (8+).

Новый индексатор делает его более возможным для отладки очень больших трассировок (50 ГБ+).
- Новый *флаг запуска debugArch* для указания архитектуры. Предварительная версия WinDbg пытается запустить подсистему отладчика с правильной битностью в целевой объект, чтобы лучше поддерживать отладку управляемого кода. Существуют обстоятельства, когда он не может определить правильную биту или вы можете переопределить то, что он решает. Используйте -debugArch x86|amd64 для управления архитектурой обработчика отладчика.

Другие изменения и исправления ошибок:

- Исправлена ошибка, из-за которой черные полосы отображались на полноэкранном отладчике с открытым окном с плавающей запятой.
- Исправлена ошибка, из-за которой параметры символов непреднамеренно очищались.
- Журнал команд теперь сохраняется при запуске из последних целевых объектов.
- В окне модели данных теперь можно изменять значения.
- Неиндексированные трассировки TTD теперь будут более ясными, что они не индексированы.
- Улучшенная производительность окна локальных параметров
- Добавлена кнопка ленты для сохранения журналов командного окна в файл.
- Добавили. SelectMany(<проекция>) для набора методов LINQ по умолчанию.

# 1.0.1807.11002

Эта версия включает эти обновления.

**Автоматическое сохранение и загрузка точек останова**. Это первый шаг для замены рабочих областей. Мы начинаем этот маршрут, включив сохранение и загрузку точек останова. Запуск того, что вы отладили ранее на вкладке "Последние" в меню файлов, теперь загрузит точки останова из этого сеанса. План состоит в том, чтобы расширить эту функцию, чтобы сохранить дополнительные сведения в сеансах. Аппаратные точки прерывания (ba) и другие различные свойства точек прерывания, такие как контексты, специфичные для потоков и процессов, а также условия, в настоящее время не сохраняются.

Незначительные изменения и исправления ошибок:

- Добавлены параметры командной строки -x, -xe, -xd, -xn и -xi для управления обработкой исключений и событий. Эти параметры командной строки ведут себя так же, как и их счетчики команд.

- Окно заметок теперь поддерживает полужирное, подчеркивание и курсив форматирования.
- Исправлены некоторые проблемы с увеличением и прокруткой.
- При выборе текста в окнах команды, памяти, источников или дизассемблии теперь отображается светлое выделение над другими экземплярами выделенного текста.
- Исправлена ошибка, из-за которой прерывание загрузки символов приводило к сбою загрузки символов для остальной части сеанса.
- Теперь NatVis перезагрузит правильно при перезапуске сеанса.

# 1.0.1805.17002

Эта версия включает эти обновления.

**Новое окно** дизассемблирования — теперь включается окно дизассемблии:

- Прокрутка вверх или вниз будет непрерывно загружать больше дизассемблии по возможности.
- Выделение синтаксиса для чисел, адресов кода и опкодов.
- При выборе символа кода откроется окно дизассемблирования в это расположение.
- При наведении указателя мыши на числа отображаются подсказки, преобразующие это число в другие базы.
- Заголовки, обозначающие начало функции.

**Быстрое исходное окно — исходное окно** было обновлено, чтобы быть быстрее и эффективнее ресурсов.

Незначительные изменения и исправления ошибок:

- Исправлены проблемы с кэшированием символов
- Исправлены некоторые случаи, когда переключение начального разрыва не было пригодным для использования, если целевой объект не сломался в
- Если вы нажимаете вкладку в командном окне без возможности, курсор будет оставаться в поле ввода
- Предварительная версия WinDbg теперь автоматически обнаруживает бит при открытии CAB-файлов

# 1.0.1804.18003

Эта версия включает эти обновления.

**Улучшения индикации состояния символов и их отмены** - Иногда отладчик отображает *BUSY* при загрузке символов, и трудно определить, что он делает и почему, если не

включена команда !sym noisy. Мы обновили Предварительную версию WinDbg, чтобы получить более эффективное взаимодействие с тем, что он делает при загрузке символов, чтобы помочь устранить любые проблемы. Помимо простого просмотра того, что происходит, мы внесли некоторые изменения, которые должны сделать отмену символов более надежными, и окно журналов будет содержать некоторые из сведений, которые обычно выводятся при включении !sym шумно. Если вы попали в представление —> журналы вы получите полный шумный символ загрузки выходных данных без необходимости включить его и попытаться перезагрузить символы.

**Окно** экспериментальных заметок — предварительная версия WinDbg теперь имеет окно для создания заметок. Просто нажмите "Вид" —> "Заметки", чтобы открыть его. Если вы копируете и вставляете его, ссылки DML будут сохранены и по-прежнему работают, как если бы это было командное окно. Вы также можете сохранять и загружать файлы заметок на ленте "Заметки" при открытии окна.

**Экспериментальное более производительное окно исходного кода** - Чтобы повысить производительность WinDbg Preview, разработано новое экспериментальное окно исходного кода, и оно довольно эффективно. Есть еще несколько пробелов вокруг контекстных меню и выделения синтаксиса, но мы хотим дать всем возможность попробовать его, прежде чем он завершится, чтобы дать нам ранние отзывы. Запустите $UseFastSourceWindow, чтобы использовать его. Если вы хотите вернуться к старому, запустите $UseMonacoSourceWindow. Настройки будут сохранены для последующих сеансов, Вам необходимо закрыть и заново открыть исходные окна, чтобы получить новую версию.

**API JSProvider версии 1.2** — для расширений JavaScript, которые объявляют поддержку API версии 1.2:

- Любой объект с методом .compareTo, который завершает скрипт, будет иметь настраиваемый компратор для него (операторы сравнения будут работать в средство оценки DX и в других местах: например: IModelObject::Compare)
- Любой объект с методом .equals, который завершает скрипт, будет иметь в нем настраиваемый оператор равенства (== и != будет работать в средство оценки DX и в другом месте: например: IModelObject::IsEqualTo)
- Объекты собственной или модели данных, вводимые в скрипт, будут иметь .compareTo и .equals на них, которые позволяют получить доступ к любому пользовательскому компратору или пользовательским реализациям равенства.

Незначительные изменения и исправления ошибок:

- Теперь .server перечисляет полное доменное имя для упрощения использования при возникновении проблем с доменом вокруг коротких имен.
- Ctrl+G теперь работает в исходном окне.

- Добавлена адресная строка в окно дизассемблирования.
- Предварительная версия WinDbg теперь будет обрабатывать _NT_SYMBOL_PATH более ожидаемым образом.
- Добавлен параметр командной строки -server.
- Запросы модели данных TTD теперь можно отображать постепенно, поэтому если вы прерываете его, вы по-прежнему увидите некоторые результаты. Эта функция по-прежнему является экспериментальной и необязательной. Запустите `dx @$cursession.TTD.AsyncQueryEnabled = 1` , чтобы включить его.
- Команда dps теперь содержит ссылки на исходные файлы, к ним относятся.

# 1.1801.19001.0

Эта версия включает эти обновления.

**Выделение** текста — теперь можно выделить все экземпляры выделенного текста непосредственно в отладчике. Чтобы использовать эту функцию, просто выделите текст в командном окне, а затем выберите "Выделить" на ленте команды или нажмите клавиши CTRL+ALT+H. При использовании одного из тех, кто уже выделен, текст удаляет выделение.

Если вы предпочитаете использовать команды, можно использовать команду "$hl":

`$hl ["someValueHere"]` — выделение текста (или отмена выделения, если уже выделено)

`$hl clearAll` — очистка всех выделенных записей

`$hl caseSensitive [1|0]` — установка сопоставления выделения для конфиденциального регистра или не учитывает регистр (по умолчанию не учитывается регистр)

В этом выпуске также содержатся некоторые незначительные исправления ошибок.

# 1.1712.15003.0

Эта версия включает эти обновления.

**Запросы** памяти TTD . Теперь вы можете запрашивать TTD для доступа к памяти аналогично тому, как вы запрашиваете вызовы сегодня. Это позволяет находить все операции чтения, записи и выполнения, которые получают доступ к определенному диапазону памяти.

Пример чтения и записи: `dx @$cursession.TTD.Memory(startAddress, endAddress, "rw")`

Уникальный пример выполнения: `dx @$cursession.TTD.Memory(startAddress, endAddress, "ec")`

**Изменения** параметров. Предварительная версия WinDbg теперь автоматически сохраняет параметры между сеансами, включая путь к символам и путь к источнику.

**Улучшения JavaScript**

- 64-разрядные числа и числовые числа в JavaScript теперь содержат метод modulo, позволяющий выполнять операцию с модулем true 64-разрядной версии.
- Объекты, определенные в JavaScript, теперь могут реализовать пользовательское сравнимое или удваиваемое понятие, которое будет работать в dx с помощью стандартных операторов C++ или в операциях LINQ. Чтобы использовать это, скрипт должен объявить в массиве initializeScript, который поддерживает новую версию API узла, вставив запись "new host.apiVersionSupport(1, 2)". После этого эти функции можно использовать в любом запросе LINQ окна модели данных или dx. Если метод реализует .compareTo(other), он сравним (операторы сравнения работают в dx и LINQ). Если метод возвращает отрицательное значение, например "это < другое". Если метод возвращает ноль, "this == other". Если метод возвращает положительное значение "это > другое". Если метод реализует .equals(other), он является эквавируемым (== работает в dx и LINQ). Метод должен возвращать значение true или false.

Незначительные изменения и исправления ошибок:

- Исправлена ошибка, из-за которой окна стека и локальных параметров не работали во время отладки загрузки.
- Обновление выходных данных LM для более точного отчета ProductVersion и аналогичных полей
- Включена кнопка "Выйти назад" во время сеансов TTD
- Добавлена поддержка -lsrcpath
- Заголовки в локальных, контрольных и модельных окнах теперь не исчезают при прокрутке вниз
- Когда ALT+Tabbing обратно в WinDbg Preview, в командном окне будет правильно сохранено расположение курсора
- Добавлено сочетание клавиш CTRL+ALT+V для переключения подробного режима
- Теперь можно отключить автоматическую прокрутку командного окна, выбрав и удерживая (или щелкнув правой кнопкой мыши) вкладку командного окна и выбрав "Отключить автоматическую прокрутку".
- Теперь можно отлаживать дочерние процессы с помощью расширенной страницы запуска исполняемого файла.

# 1.0.14.0

Эта версия включает эти обновления.

**Улучшенный интерфейс** сервера обработки — новое уведомление в меню "Файл", чтобы показать, к чему подключен сервер обработки и с которым вы взаимодействуете, был добавлен. В рамках этих изменений при завершении сеанса отладки подключение сервера обработки будет сохраняться и может быть отключено в меню "Файл".

**Новые предустановленные параметры макета на ленте "Вид"** — на ленте "Вид" появился новый параметр "Макеты". В настоящее время существует три макета: значение по умолчанию, одно ориентировано на дизассембли и один минимальный.

**Лента отладки 'Перемещение во времени'** — есть расширенная лента 'Перемещение во времени', которая будет отображаться при отладке трассировки отладки перемещения во времени.

**Метаданные из скриптов** JavaScript— расширения JavaScript теперь могут возвращать метаданные для свойств и других конструкций. Это означает, что расширение может предоставлять строки справки, указывать радикс отображения значений и многое другое. Метаданные предоставляются путем размещения дескриптора метаданных в объекте с помощью функции Symbol.metadataDescriptor или явного вызова host.metadata.defineMetadata. Возвращаемые функции, итерированные значения и другие контексты значений могут возвращать метаданные для их значения через host.metadata.valueWithMetadata.

**Обновления** API JavaScript. Некоторые потенциально критические изменения на уровне источника были внесены в API в поставщике JavaScript (включая новые проецируемые методы и свойства в собственных объектах). Существующие расширения не увидят никаких потенциально разрушающих изменений, без указания поддержки новой версии API JsProvider. Поддержка новой версии API указывается путем размещения записи host.apiVersionSupport в массиве, возвращаемой инициализациейScript, с утверждением поддержки версии 1.1. может быть? .. со значением, указывающим поддержку версии 1.1.

Изменения в API версии 1.1 включают:

- host.getModuleSymbol и host.getModuleType возвращают значение NULL, если они не могут найти символ вместо исключения.
- Все собственные объекты имеют свойство адреса в дополнение к .targetLocation. Если у объекта нет адреса, при доступе к свойству возникнет исключение.
- Все собственные объекты имеют новые методы .getObjectValue и .setObjectValue для доступа к свойствам объекта, который может конфликтовать с именами мест JavaScript на объекте (например, "адрес").

**Дополнительные изменения JavaScript**

- Расширения JavaScript теперь могут добавлять и удалять свойства объектов модели данных с помощью Object.defineProperty и оператора удаления. Добавление или регистрация класса JavaScript в качестве родительской модели или сигнатуры типов по-прежнему является наиболее предпочтительным способом управления объектной моделью.
- Расширения JavaScript теперь могут изменять глобальные переменные в модулях в целевом объекте отладки с помощью нового API host.setModuleSymbol.
- Все математические функции, которые находятся на 64-разрядном типе библиотеки (например, .add, .subtract, .умножение, деление и т. д.) теперь присутствуют в числах JavaScript.
- Функции и свойства JavaScript теперь могут возвращать значения, которые являются перечислениями с помощью пользовательского маршалинга. Функция или метод доступа к свойствам может возвращать host.typeSystem.marshalAs(value, type...) для создания такого пользовательского маршалинга.
- Команда точки останова в отладчике скрипта теперь может прерывать имена функций в дополнение к позициям строки или столбца.
- Объекты типа в расширениях JavaScript имеют доступ к их содержащим модулю через свойство .containingModule.

Незначительные изменения и исправления ошибок:

- Исправлено форматирование вкладок условной ленты, которые будут менее запутанными.
- Повторно работал DML, чтобы быть более строгим в синтаксическом анализе для повышения производительности.
- Различные исправления с производительностью и поведением CTRL+F.
- Добавлено предупреждение при выполнении без повышенных привилегий перед попыткой использовать TTD.
- Добавлен параметр переопределения автоматического обнаружения битов целевого объекта.
- Отключены различные параметры меню файлов и ленты, если они не могут использоваться (например, Go, когда в файле дампа).

Известные проблемы:

- SOS не будет работать для x86-трассировок.

# 1.0.13.0

Эта версия добавляет трассировку путешествия по времени. Отладка перемещения по времени позволяет записывать процесс, а затем воспроизводить его позже как вперед, так и назад. Отладка временных путешествий (TTD) помогает упростить отладку проблем, позволяя "перемотка" сеанса отладчика вместо необходимости воспроизводить проблему, пока не найдете ошибку. Дополнительные сведения см. в статье "Отладка временных путешествий" — обзор.

## 1.0.12.0

Эта версия была первым выпуском WinDbg Preview. Общие сведения о функциях, доступных в WinDbg Preview, основных функциях WinDbg.

## См. также

WinDbg — установка

WinDbg — параметры запуска командной строки

# WinDbg — параметры запуска командной строки

18.07.2025



## Starting WinDbg

После установки WinDbg WinDbgX.exe может быть запущен из любого места в каталоге.

## Параметры запуска командной строки

dbgsyntax

```
WinDbgX [options]
```

В следующих таблицах перечислены доступные параметры командной строки.

### General Options

Развернуть таблицу

| Option | Description |
|---|---|
| -c *"command"* | Выполняет командную строку после подключения отладчика. Эта команда должна быть заключена в кавычки. Несколько команд можно разделить точкой с запятой. |
| -v | Включает вывод подробных данных в отладчике. |
| -T *Title* | Задает заголовок окна. |
| -logo *LogFile* | Log Open. Начинает запись информации в лог-файл. Если файл уже существует, он будет перезаписан. |
| -loga *LogFile* | Log Append. Начинает запись информации в лог-файл. Если файл существует, он будет добавлен. |
| -e *EventHandle* | Сигнализирует о событии с заданным дескриптором после следующего исключения в целевом объекте. |
| -? | Отображает сводку доступных команд. |

## Kernel Options

Развернуть таблицу

| Option | Description |
|---|---|
| -k [ConnectType] | Запускает сеанс отладки ядра. If **-k** is used without any *ConnectType* options following it, it must be the final entry on the command line. |
| -kqm | Запускает KD в тихом режиме. |
| -kl | Запускает сеанс отладки ядра на том же компьютере, что и отладчик. |
| -kx *ExdiOptions* | Запускает сеанс отладки ядра с помощью драйвера EXDI. Дополнительные сведения об EXDI см. в разделе "Настройка транспорта отладчика EXDI". |
| -d | После перезагрузки отладчик переходит на целевой компьютер сразу после загрузки модуля ядра. |

## Параметры пользовательского режима

Развернуть таблицу

| Option | Description |
|---|---|
| -o | Отладка всех процессов, запущенных целевым приложением (дочерними процессами). |
| -g | Игнорирует начальную точку останова в целевом приложении. |
| -G | Игнорирует окончательную точку останова в целевом приложении. |
| -pv | Указывает, что отладчик должен подключаться к целевому процессу ненавязчиво. |
| -hd | Указывает, что отладочная куча не должна использоваться. |
| -cimp | Указывает, что все процессы, созданные, будут использовать неявную командную строку, заданную сервером, а не строку командной строки пользователя из клиента. |

## Target Options

Развернуть таблицу

| Option | Description |
|---|---|
| -remote *ClientTransport* | Подключается к серверу отладки, который уже запущен. For an explanation of the possible *ClientTransport* values, see Activating a Debugging Client. Если этот параметр используется, он должен быть первым параметром в командной строке. |

| Option | Description |
| --- | --- |
| -server *ServerTransport* | Создает сервер отладки, к которому можно получить доступ другим отладчикам. For an explanation of the possible *ServerTransport* values, see Activating a Debugging Server. |
| -premote *SmartClientTransport* | Создает смарт-клиент и подключается к серверу обработки, который уже запущен. Описание возможных значений SmartClientTransport см. в разделе "Активация смарт-клиента". |
| -p *PID* | Указывает идентификатор десятичного процесса для отладки. |
| -tid *TID* | Указывает ID потока, который должен быть возобновлен при запуске сеанса отладки. |
| -psn *ServiceName* | Указывает имя службы, содержащейся в процессе отладки. Используется для отладки процесса, который уже запущен. |
| -pn *ProcessName* | Указывает имя процесса для отладки. |
| -z *DumpFile* | Указывает имя файла аварийного дампа для отладки. Если путь и имя файла содержат пробелы, это должно быть окружено кавычками. |
| -debugArch x86 -или-amd64 | Переопределите автоматическое определение и установите целевую битность для отладчика. |
| -loadSession | Загрузите сохраненный файл конфигурации сеанса. |
| -setupFirewallRules | Настраивает необходимые правила брандмауэра в локальной системе, чтобы разрешить отладку ядра с помощью KDNET. |
| -openPrivateDumpByHandle *Handle* | *Только внутреннее использование Майкрософт*. Указывает дескриптор файла аварийного дампа для отладки. |
| -benchmarkStartup | *Только внутреннее использование Майкрософт*. Запускает тест запуска и добавляет результат в файл. |

## Symbol Options

Развернуть таблицу

| Option | Description |
| --- | --- |
| -y *SymbolPath* | Указывает используемый путь к символам. Separate multiple paths with a semicolon (;). Если путь содержит пробелы, он должен быть заключен в кавычки. For details, and for other ways to change this path, see Symbol Path. |
| -n | Шумная загрузка символов. Активирует детализированный вывод данных обработчика символов. |

| Option | Description |
|---|---|
| -i *ImagePath* | Задает используемый путь поиска изображений. |
| -sdce | Причиняет отладчику отображение сообщений об ошибке доступа к файлу во время загрузки символов. |
| -ses | Позволяет отладчику выполнять строгую оценку всех файлов символов и игнорировать любые сомнительные символы. |
| -sicv | Заставляет обработчик символов игнорировать запись CV. |
| -sins | Позволяет отладчику игнорировать путь символа и переменные среды пути к исполняемому изображению. |
| -snc | Приводит к отключению отладчика перевода C++. |
| -snul | Отключает автоматическую загрузку символов для неквалифицированных имен. |
| -sup | Вызывает обработчик символов для поиска открытой таблицы символов во время каждого поиска символов. |
| -sflags | Задает все параметры обработчика символов одновременно. |

### Параметры исходного пути

Развернуть таблицу

| Option | Description |
|---|---|
| -srcpath | Указывает исходный путь для использования на сервере отладки. |
| -lsrcpath | Указывает исходный путь для использования на локальном клиенте. |

Если вы находитесь в локальном сеансе отладчика, srcpath и lsrcpath фактически одинаковы (сервер — это локальный сеанс). Для удаленной отладки существуют ситуации, в которых можно установить их на разные значения. For more information about remote debugging see, Remote Debugging.

### Exception handling

Развернуть таблицу

| Option | Description |
|---|---|
| -x | Включите обработку второго шанса только для исключений, связанных с нарушением доступа. |

| Option | Description |
| --- | --- |
| -xe *Exception* | Включите обработку исключений первого шанса для указанного исключения. |
| -xd *Exception* | Включите обработку исключений второго шанса для указанного исключения. |
| -xn *Exception* | Для данного исключения отключите обработку первой и второй попытки и отображайте только сообщение на консоли. |
| -xi *Exception* | Полностью игнорируйте данное исключение, отключив обработку первого и второго случаев и не выводя ничего в консоль. |

Список исключений, которые можно указать, см. в разделе "Определения событий" и "Значения по умолчанию".

### Post Mortem

Развернуть таблицу

| Option | Description |
| --- | --- |
| -I | Устанавливает WinDbg в качестве отладчика по умолчанию для анализа после сбоя в системе. |
| -IS | Задает WinDbg в качестве отладчика после смерти по умолчанию для системы автоматически, и сообщается только об ошибках. |

### Deprecated Options

Развернуть таблицу

| Option | Description |
| --- | --- |
| -Q | Нерекомендуемый параметр командной строки. |
| -QY | Нерекомендуемый параметр командной строки. |
| -QS | Нерекомендуемый параметр командной строки. |
| -QSY | Нерекомендуемый параметр командной строки. |
| -WX | Нерекомендуемый параметр командной строки. |

Общие сведения о параметрах запуска см. в разделе "Параметры Command-Line WinDbg".

Вы можете использовать ?? чтобы перечислить поддерживаемые параметры командной строки.




# See Also

WinDbg Features

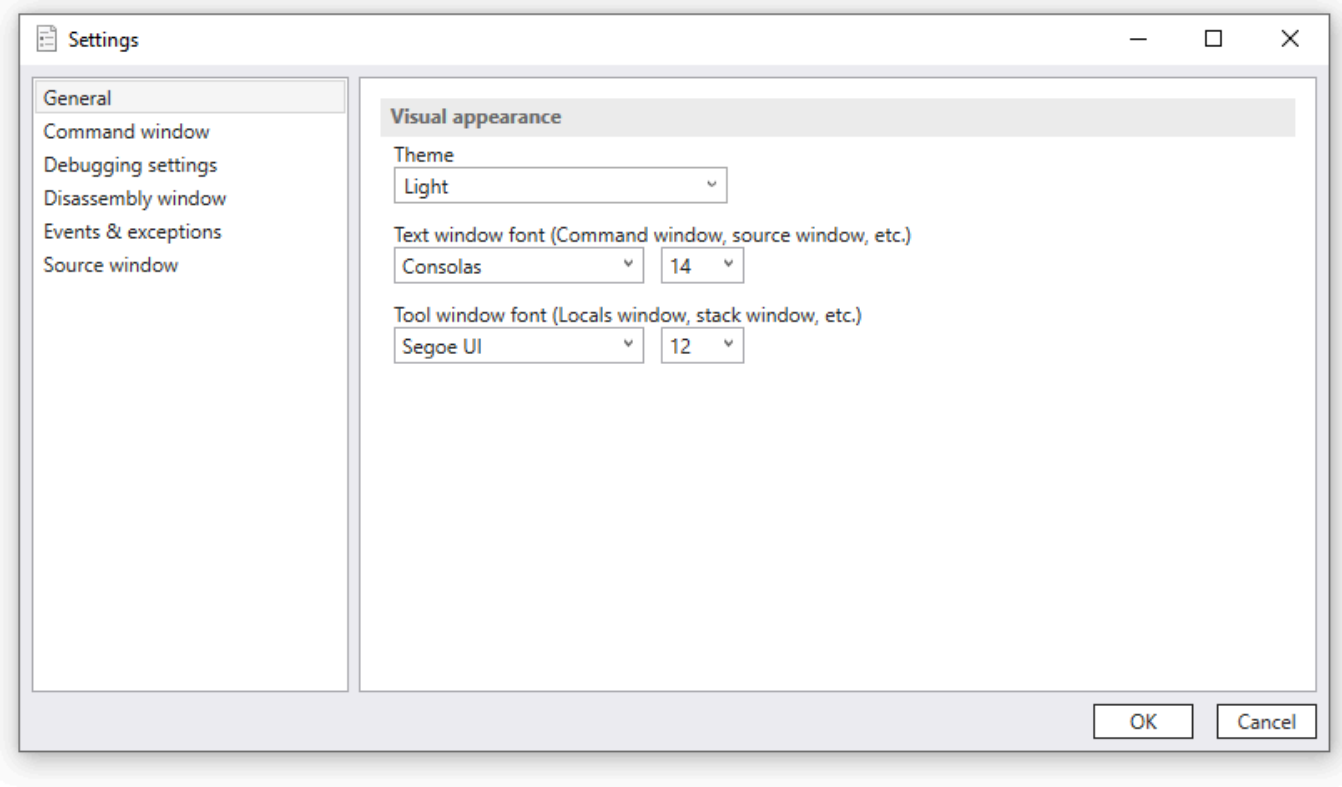
# WinDbg — параметры и рабочие области

18.07.2025

В этом разделе описывается установка и настройка отладчика WinDbg.

## Настройки

Используйте меню параметров, чтобы задать такие вещи, как путь к источнику и символу, а также выбрать светлую и темную тему для отладчика.




В настоящее время есть шесть параметров диалоговых панелей:

- Общая информация
- Командное окно
- Параметры отладки
- окна дизассемблирования
- События и исключения
- Окно источника

Дополнительные сведения о настройке путей см. в разделе "Путь символов" для отладчика Windows и отладки исходного кода в WinDbg (классический).

# Пространства для работы

Рабочие пространства позволяют сохранять сведения о конфигурации в файле целевой информации о подключении.

Параметры в рабочих областях сохраняются при закрытии отладчика или могут быть сохранены вручную с помощью *файла ->Save Workspace*.

Рабочие области автоматически загружаются при запуске из списка последних целевых объектов или их можно загрузить вручную в меню файлов.

Помимо сведений о целевом подключении, следующие параметры хранятся в файле рабочих областей.

## Общие параметры

> **Примечание**
>
> Этот список и формат не является окончательным и подлежит изменению.

Развернуть таблицу

| Настройки | По умолчанию | Описание |
|---|---|---|
| FinalBreak | правда | Если значение true, игнорирует окончательную точку останова (-g параметр командной строки). |
| Отладка исходного кода | правда | Переключается между режимом источника или сборки. |
| DebugChildProcesses | неправда | (Только в пользовательском режиме) При значении true целевое приложение будет отлаживать запущенные им дочерние процессы. ( параметр командной строки-o). |
| Неинвазивный | неправда | Указывает неинвазивное подключение (-pv параметр командной строки). |
| NoDebugHeap | неправда | Указывает, что отладочная куча не должна использоваться (параметр командной строки-hd). |
| Многословный | неправда | Если включен подробный режим, некоторые команды отображения (например, дамп регистров) создают более детализированные данные. ( параметр командной строки-v). |
| Elevate | - | Используется внутри WinDbg — не изменяйте. |

| Настройки | По умолчанию | Описание |
|---|---|---|
| Можно перезапустить | - | Используется в WinDbg — не изменяйте. |
| UseImplicitCommandLine | неправда | Используйте неявную командную строку (-cimp параметр командной строки). Это запускает отладчик с неявной командной строкой, а не явным процессом для выполнения. |

Дополнительные сведения о параметрах командной строки см. в разделе "Параметры" Command-Line WinDbg.

## Параметры символов

Развернуть таблицу

| Настройки | По умолчанию | Описание |
|---|---|---|
| Переопределение параметров символа | 0 | Явное маскирование параметра символа в виде одного шестнадцатеричного числа. |
| СледуетПереопределитьПараметрыСимвола | неправда | Если задано *значение true* , переопределите все параметры символов, перечисленные ниже, с помощью предоставленной маски параметра символа, описанной выше. |
| SymOptExactSymbols | неправда | Этот параметр приводит к тому, что отладчик выполняет строгую оценку всех файлов символов. |
| SymOptFailCriticalErrors | неправда | Этот параметр символа приводит к отключению диалоговых окон доступа к файлам. |
| SymOptIgnoreCvRec | неправда | Этот параметр приводит к тому, что обработчик символов игнорирует запись CV в загруженном заголовке изображения при поиске символов. |
| SymOptIgnoreNtSympath | неправда | Этот параметр приводит к тому, что отладчик игнорирует параметры переменной среды для пути символа и пути к исполняемому изображению. |

| Настройки | По умолчанию | Описание |
|---|---|---|
| SymOptNoCpp | неправда | Этот параметр символа отключает перевод C++. Если задан этот параметр символа, :: заменяется на __ во всех символах. |
| SymOptNoUnqualifiedLoads | неправда | Этот параметр символа отключает автоматическую загрузку модулей обработчика символов. Если этот параметр задан и отладчик пытается сопоставить символ, он будет выполнять поиск только модулей, которые уже загружены. |
| SymOptAutoPublics | неправда | Этот параметр символа приводит к тому, что DbgHelp будет выполнять поиск в таблице открытых символов в PDB-файле только в качестве последнего способа. Если при поиске данных закрытого символа обнаружены какие-либо совпадения, открытые символы не будут искать. Это повышает скорость поиска символов. |
| SymOptDebug | неправда | Этот параметр символа включает загрузку шумных символов. Это указывает отладчику отображать сведения о поиске символов. |

Дополнительные сведения о параметрах символов см. в разделе "Параметры символов".

## Параметры макета окна

Макет окна сохраняется глобально и не сохраняется в файле рабочих областей.

## XML-файл рабочих областей

Сведения о рабочей области и целевом подключении хранятся в формате XML.

В следующем файле показан пример файла конфигурации рабочих областей.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<TargetConfig Name="C:\paint.dmp" LastUsed="2017-08-03T21:34:20.1013837Z">
```

```
    <EngineConfig />
    <EngineOptions>
      <Property name="FinalBreak" value="true" />
      <Property name="SourceDebugging" value="true" />
      <Property name="DebugChildProcesses" value="false" />
      <Property name="Noninvasive" value="false" />
      <Property name="NoDebugHeap" value="false" />
      <Property name="Verbose" value="false" />
      <Property name="SymbolOptionsOverride" value="0" />
      <Property name="ShouldOverrideSymbolOptions" value="false" />
      <Property name="SymOptExactSymbols" value="false" />
      <Property name="SymOptFailCriticalErrors" value="false" />
      <Property name="SymOptIgnoreCvRec" value="false" />
      <Property name="SymOptIgnoreNtSympath" value="false" />
      <Property name="SymOptNoCpp" value="false" />
      <Property name="SymOptNoUnqualifiedLoads" value="false" />
      <Property name="SymOptAutoPublics" value="false" />
      <Property name="SymOptDebug" value="false" />
      <Property name="Elevate" value="false" />
      <Property name="Restartable" value="true" />
      <Property name="UseImplicitCommandLine" value="false" />
    </EngineOptions>
    <TargetOptions>
      <Option name="OpenDump">
        <Property name="DumpPath" value="C:\paint.dmp" />
      </Option>
    </TargetOptions>
  </TargetConfig>
```

Обратите внимание, что этот формат файла продолжает развиваться по мере добавления дополнительных функций в отладчик WinDbg.

---

## См. также

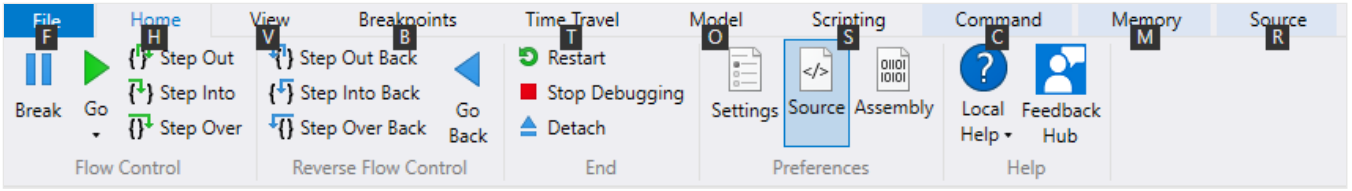
Функции WinDbg

# Сочетания клавиш WinDbg

12.03.2025

В этом разделе приведены сочетания клавиш для отладчика WinDbg.

Все параметры меню ленты доступны с помощью **ALT+** первой буквы параметра. Например, **ALT+ H** перейти в меню "Главная", **ALT + H + S**, чтобы остановить отладку.




## Управление потоком

Развернуть таблицу

| Нажатие клавиши | Описание |
|---|---|
| F5 | Продолжать |
| F10 | Переступить |
| F11 | Войдите в |
| SHIFT+F11 | Выйти |
| F7 | Перейти к строке |
| CTRL+SHIFT+I | Установка указателя инструкции на выделенную строку |
| CTRL+Break или ALT+DEL | Ломать / Перерыв / Прекращение |
| CTRL+SHIFT+F5 | Перезапуск |
| SHIFT+F5 | Остановка отладки |
| ALT+H, D | Отделять |

## Командное окно

Развернуть таблицу

| Нажатие клавиши | Описание |
|---|---|
| CTRL+M | Переключение фокуса между областью вывода (режимом разметки) и полем ввода |
| CTRL+ПРОБЕЛ | Всплывающее окно списка завершения команд триггера |
| ВКЛАДКА | Активация следующего автозаполнения |
| SHIFT+TAB | Вызов предыдущего автозаполнения |
| CTRL+F | Открыть поле поиска |
| ↑ / ↓ | Просмотреть историю команд |
| Shift+↑ / ↓ | Навигация по истории команд, соответствующая текущему тексту в командной строке |
| CTRL+[ | Выберите предыдущий раздел области вывода |
| CTRL+] | Выберите следующий раздел области вывода |

## Настройка

Развернуть таблицу

| Нажатие клавиши | Описание |
|---|---|
| F6 | Присоединение к процессу |
| CTRL+R | Подключитесь к удалённому серверу |
| CTRL+D | Открытие файла дампа |
| CTRL+K | Подключение к отладчику ядра |
| CTRL+E | Процесс запуска |
| CTRL+P | Запуск пакета приложения |

## Точки останова

Развернуть таблицу

| Нажатие клавиши | Описание |
| --- | --- |
| F9 | Переключить точку останова на выделенной строке |
| CTRL +F9 | Включение состояния точки останова в выделенной строке |
| Shift + F9 | Добавление точки останова |
| CTRL+ALT+K | Переключение начального разрыва |

## Оконное управление

Развернуть таблицу

| Нажатие клавиши | Описание |
| --- | --- |
| CTRL+TAB | Открытие функции изменения окна |
| Alt+1 | Открыть или навести фокус на командное окно |
| ALT+2 | Открыть/сфокусироваться на просмотровом окне |
| ALT+3 | Сфокусироваться на окне локальных переменных |
| Alt+4 | Откройте окно регистров и сфокусируйтесь на нем |
| ALT+5 | Открыть/переключить внимание на окно памяти |
| ALT+6 | Откройте или сфокусируйтесь на окне стека |
| ALT+7 | Открыть и сфокусироваться на окне дизассемблера |
| ALT+8 | Открыть/сфокусироваться на окне точек останова |
| ALT+9 | Открытие и фокус на окне темы |

## Скриптинг

Развернуть таблицу

| Нажатие клавиши | Описание |
| --- | --- |
| CTRL+SHIFT+O | Открытие скрипта |
| Ctrl+Shift+Ввод | Выполнение скрипта |
| CTRL+S | Сохранение скрипта |
| ALT+S, N | Новый скрипт |

| Нажатие клавиши | Описание |
| --- | --- |
| ALT+S,U | Разрыв связи скрипта |

## Навигация по стеку

Развернуть таблицу

| Нажатие клавиши | Описание |
| --- | --- |
| Ctrl+↑ / ↓ | Перемещение вверх и вниз кадра стека |

## Справка

Развернуть таблицу

| Нажатие клавиши | Описание |
| --- | --- |
| F1 | Откройте файл справки |
| SHIFT+F1 | Поиск в интернете (окно источника) |

## Разное.

Развернуть таблицу

| Нажатие клавиши | Описание |
| --- | --- |
| CTRL+ALT+V | Переключить подробный режим |

# См. также

возможности WinDbg

# WinDbg — запуск сеанса пользовательского режима

18.07.2025



В этом разделе описывается запуск сеанса пользовательского режима с отладчиком WinDbg.

Выберите "*Файл*", "*Начать отладку*" и выберите один из следующих четырех вариантов:

- *Запустите исполняемый файл. Запускает исполняемый* файл и присоединяет его, просматривая целевой объект.
- *Запуск исполняемого файла (дополнительно)* — запускает исполняемый файл и подключает его с помощью набора диалоговых окон с дополнительными параметрами.
- *Присоединение к процессу* — присоединение к существующему процессу.
- *Запуск пакета приложения — запускает пакет* и присоединяет его к приложению.

Здесь описаны все четыре варианта.

## Запуск исполняемого файла

Используйте этот параметр для запуска исполняемого файла и присоединения к нему.

Перейдите к нужному исполняемому файлу в диалоговом окне предоставленного файла и откройте его.

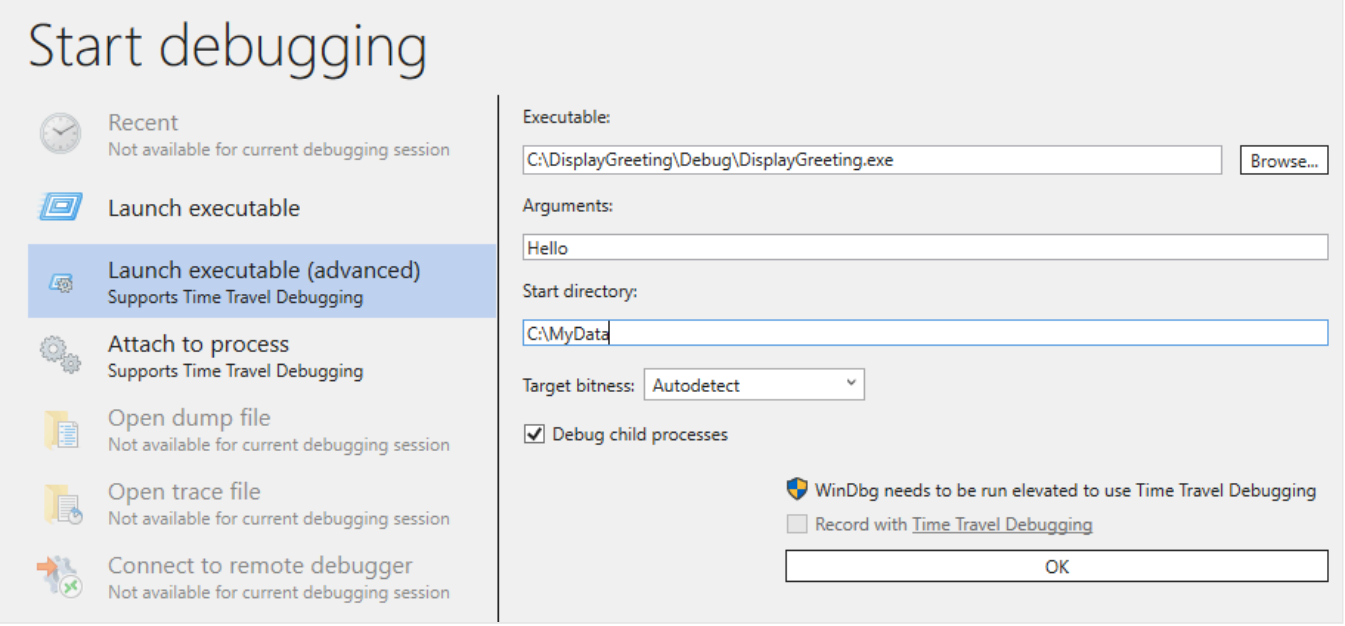
## Запуск исполняемого файла (расширенные настройки)

Используйте этот параметр для запуска исполняемого файла и присоединения к нему с помощью набора текстовых полей с расширенными параметрами.

Укажите следующие параметры:

- Путь к исполняемому файлу, например *C:\Windows\notepad.exe*
- Необязательные аргументы для предоставления исполняемому файлу при запуске
- Необязательное расположение каталога запуска

- Целевая битность, авто 32 или 64.
- Включение дочернего процесса отладки
- Запись с отладкой перемещения по времени




# Присоединение к процессу

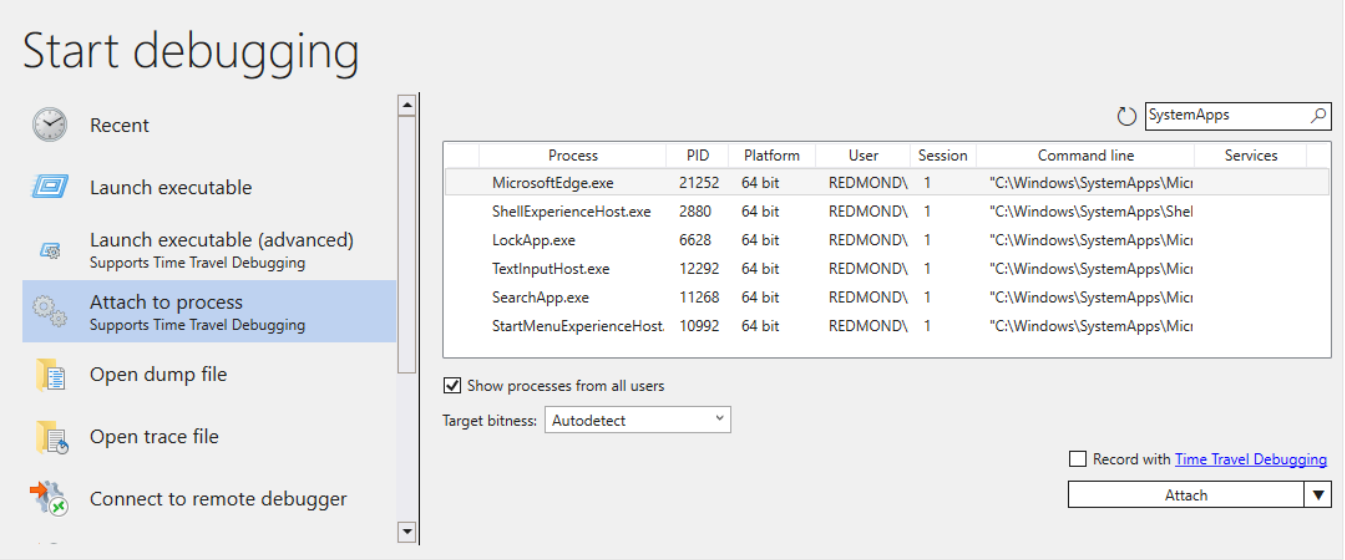
Используйте этот параметр для подключения к существующему процессу.

Выберите "*Показать процесс*" *от всех пользователей* , чтобы отобразить дополнительные процессы.

Используйте поле поиска, чтобы отфильтровать список процессов, например, выполнив поиск SystemApps.

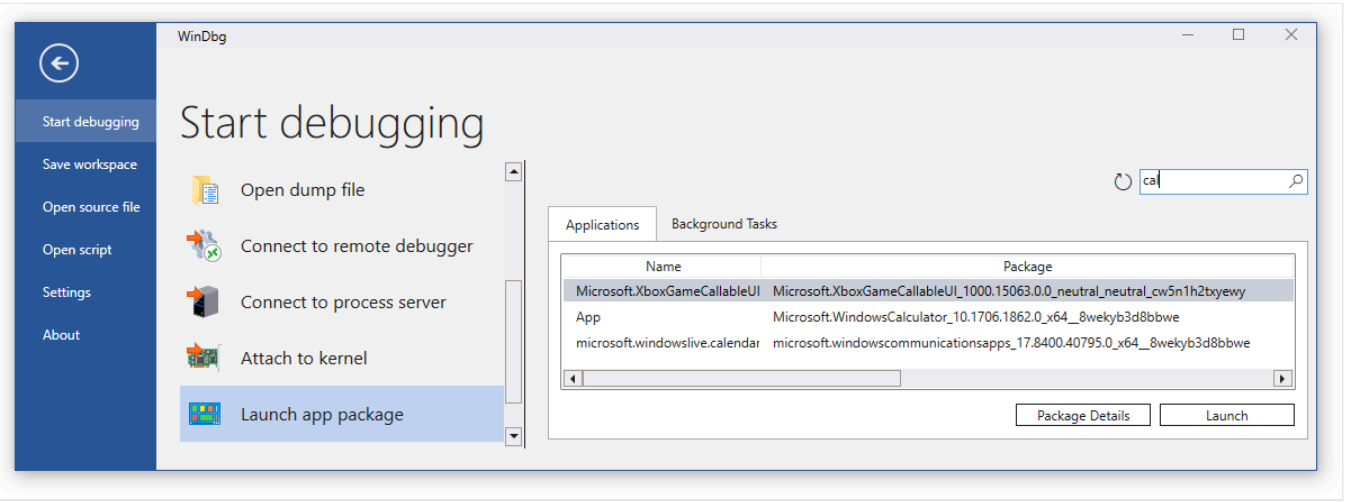
> ⓘ **Примечание**
>
> Для подключения к предметам с щитом UAC потребуется повысить уровень прав отладчика.

Используйте выпадающее меню на кнопке "*Присоединить*", чтобы выбрать *неинвазивное присоединение*.




# Запуск пакета приложений

Используйте этот параметр для запуска и подключения к пакету приложения с помощью одной из вкладок "Приложения" или "Фоновые задачи". Используйте поле поиска, чтобы найти определенное приложение или фоновую задачу. Нажмите кнопку "Сведения о пакете", чтобы отобразить сведения о пакете.




# См. также

Функции WinDbg

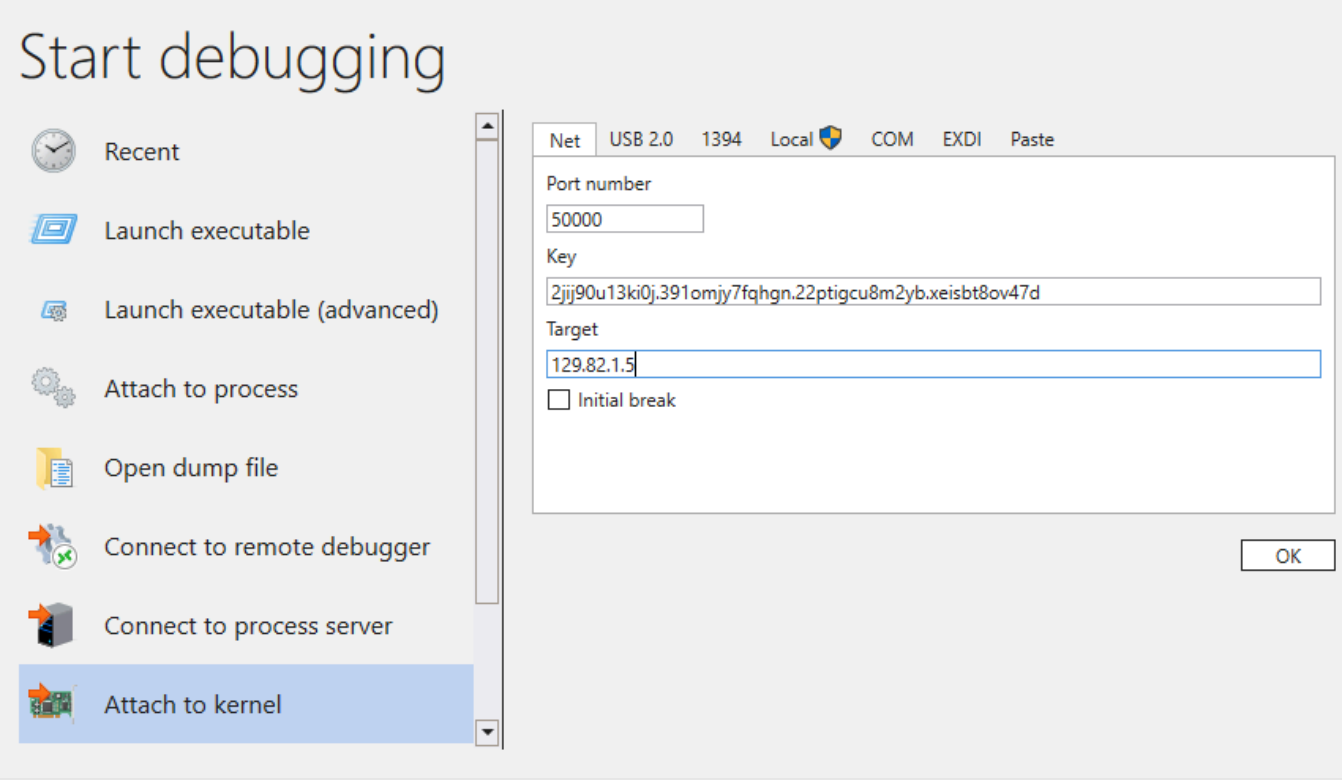
# WinDbg — запуск сеанса режима ядра

18.07.2025

В этом разделе описывается запуск сеанса режима ядра с помощью WinDbg.

Процесс очень похож на то, как это было сделано с предыдущими версиями WinDbg. Выберите вкладку для типа используемого транспорта, заполните обязательные поля и нажмите кнопку "OK".

> ⓘ **Примечание**
>
> Для отладки локального ядра требуется запустить WinDbg с повышенными привилегиями.




Вкладка "*Вставка*" позволяет вставлять пользовательские строки подключения.

Если вы не знакомы с настройкой сеанса режима ядра отладчика, см. статью "Начало работы с WinDbg" (Kernel-Mode)

## См. также

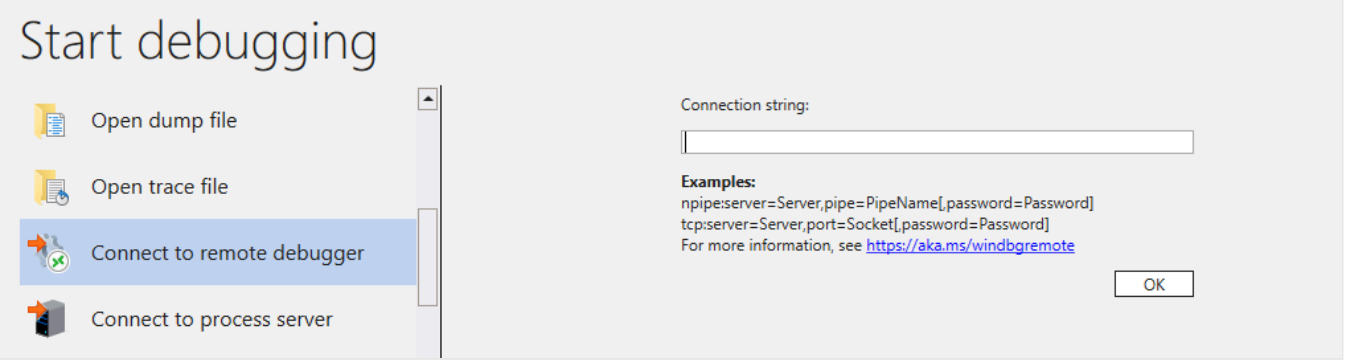
# WinDbg — запуск удаленного сервера обработки и сеанса дампа

18.07.2025

В этом разделе описывается, как запустить удаленный сеанс сервера процессов и сеанс дамп-файлов с помощью отладчика WinDbg.

## Удаленный сервер отладки

Используйте этот параметр для подключения к удаленному серверу отладки.




Удаленная отладка включает два отладчика, работающих в двух разных расположениях. Отладчик, выполняющий отладку, называется сервером отладки. Второй отладчик, называемый клиентом отладки, управляет сеансом отладки из удаленного расположения. Чтобы установить удаленный сеанс, сначала необходимо настроить сервер отладки, а затем подключиться к нему с клиентом отладки.

Для получения дополнительной информации об удаленных сеансах см. раздел "Функции удаленного WinDbg".

## Сервер отладки процесса

Удаленная отладка с помощью сервера обработки включает в себя запуск небольшого приложения, называемого сервером обработки на серверном компьютере. Затем на клиентском компьютере запускается отладчик в режиме пользователя. Так как этот отладчик будет выполнять всю фактическую обработку, он называется смарт-клиентом.

Дополнительные сведения о сеансах сервера обработки см. в разделе "Серверы обработки" (режим пользователя).

## Открытие файла дампа

Чтобы открыть файл дампа, перейдите к нужному файлу в диалоговом окне предоставленного файла и откройте его.

Дополнительные сведения о различных типах файлов дампа см. в статье Анализ файлов аварийного дампа с помощью WinDbg.

---

## См. также

Функции WinDbg

# WinDbg — меню "Файл"

Статья • 23.05.2024

В этом разделе описывается, как использовать меню файла.

## Запуск отладки

При первом открытии меню файла вы увидите *начало отладки* и последние целевые объекты отладчика. Используйте *запуск отладки* , чтобы настроить новые и открыть предыдущие сеансы отладчика.

### Последние

Последний список содержит список последних рабочих областей и подключений отладчика. Дополнительные сведения о рабочих областях параметров см. в статье "Настройка WinDbg" Параметры и рабочих областей.

Вы можете использовать меню правой кнопкой мыши для управления рабочими областями, например закрепление, переименование и перемещение их. А также редактирование их в блокноте.




### Создание нового сеанса

Используйте другие вкладки в *разделе "Запуск отладки* ", чтобы запустить новый сеанс отладчика, например присоединение или запуск процесса. Дополнительные сведения о запуске нового сеанса см. в статье WinDbg . Запуск сеанса в пользовательском режиме и WinDbg — запуск сеанса режима ядра

## Сохранение рабочей области

Сохраните *рабочую область* для сохранения текущей рабочей области.

Сведения о подключении к сеансу хранятся в файлах конфигурации рабочей области. Файлы рабочей области хранятся с расширением файла .debugTarget.

Расположение по умолчанию для файлов рабочей области:

Консоль

```
C:\Users\*UserName*\AppData\Local\DBG\targets
```

## Файл с открытым исходным кодом

Используйте *файл с открытым исходным кодом* для открытия исходного файла. Это происходит, если вы хотите работать с дополнительными исходными файлами, которые не были загружены из-за выполнения кода. Дополнительные сведения о работе с исходными файлами см. в статье "Отладка исходного кода" в WinDbg (классическая версия)

## Открыть скрипт

Используйте *скрипт Open, чтобы открыть существующий скрипт* Javascript или NatVis. Дополнительные сведения о работе со скриптами см. в меню WinDbg — Scripting Menu.

## Настройки

Используйте меню параметров, чтобы задать путь к источнику и символу, а также выбрать светлую и темную тему для отладчика. Дополнительные сведения о параметрах см. в статье "Настройка WinDbg" — параметры и рабочие области.

## О программе

*Используйте* сведения о версии сборки для отладчика. Вы также можете использовать этот экран для просмотра заявления о конфиденциальности Майкрософт.

## Exit

Используйте *Exit* для выхода из отладчика.

## См. также

Функции WinDbg

## Обратная связь

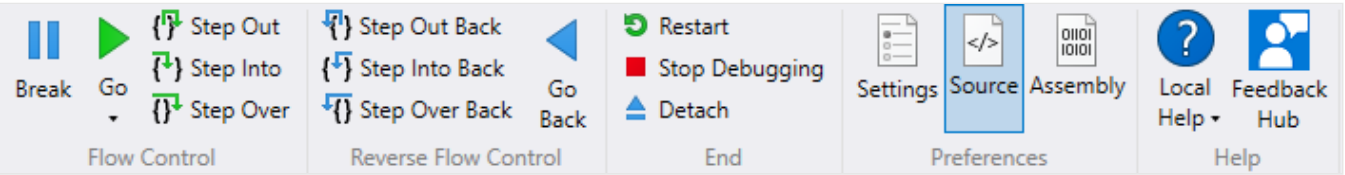
**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# WinDbg — меню "Главная"

18.07.2025

В этом разделе описывается, как работать с домашним меню.




## Управление потоком

Используйте кнопки *управления потоком*, чтобы войти в подключенный отладочный объект, возобновить выполнение кода на объекте и пошагово входить в код и выходить из него.

## Управление обратным потоком

Используйте кнопки *управления обратным потоком* , чтобы вернуться к времени. Дополнительные сведения см. в статье "Отладка временных путешествий" — обзор.

## Конец

Используйте кнопки *end* для перезапуска, отсоединения и остановки отладки.

## Предпочтения

Используйте кнопки "*Параметры*", чтобы переключаться между исходным кодом и **представлениями сборки** и получать доступ к меню "Параметры".

## Помощь (Поддержка)

Используйте кнопки *справки* для выполнения следующих действий:

- Просмотр локальной справки в автономном режиме
- Онлайн-справка — самая актуальная

# Отзывы о GitHub

- Если вы найдете ошибки или запрос на функцию, нажмите кнопку обратной связи на ленте, чтобы перейти на страницу GitHub , где можно отправить новую проблему.

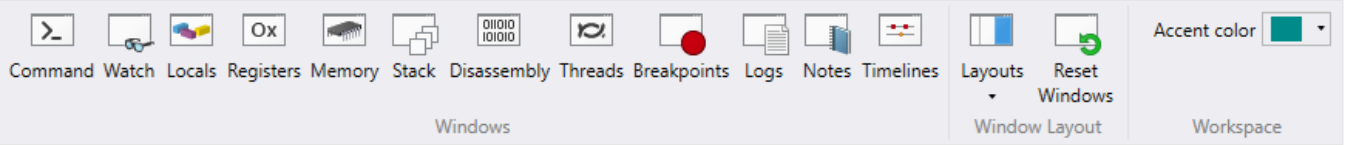
# См. также

Функции WinDbg

# WinDbg — представление меню

18.07.2025

В этом разделе описывается, как работать с меню представления в WinDbg.




Меню просмотра откроет новое окно для каждого элемента, или переместит фокус на существующее окно, если оно уже открыто.

## командование

В командном окне можно ввести команды отладчика. Дополнительные сведения о командах отладчика см. в разделе "Команды отладчика".

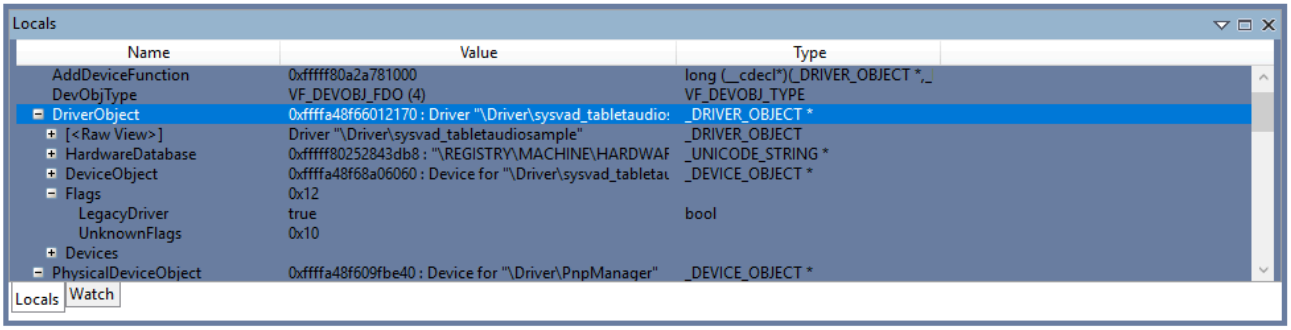
## Смотреть

Окно просмотра позволяет просматривать локальные переменные и регистры.

Локальные и контрольные окна основаны на модели данных, используемой командой DX. Это означает, что локальные и окна наблюдения смогут воспользоваться всеми расширениями NatVis или JavaScript, загруженными вами, и поддерживать полные запросы LINQ, так же как команда dx. Дополнительные сведения о модели данных см. в статье WinDbg — Модель данных.

## Местные жители

В окне "Локальные" отображаются сведения обо всех локальных переменных в текущей области. В окне локальных языков будут выделены значения, которые изменились во время предыдущего выполнения кода.




# РегистрЫ

Регистры отображают содержимое регистров процессоров, когда они доступны. Дополнительные сведения о регистрах см. в разделе "Регистры" и "Просмотр и редактирование регистров" в WinDbg (классическая версия).

# Память

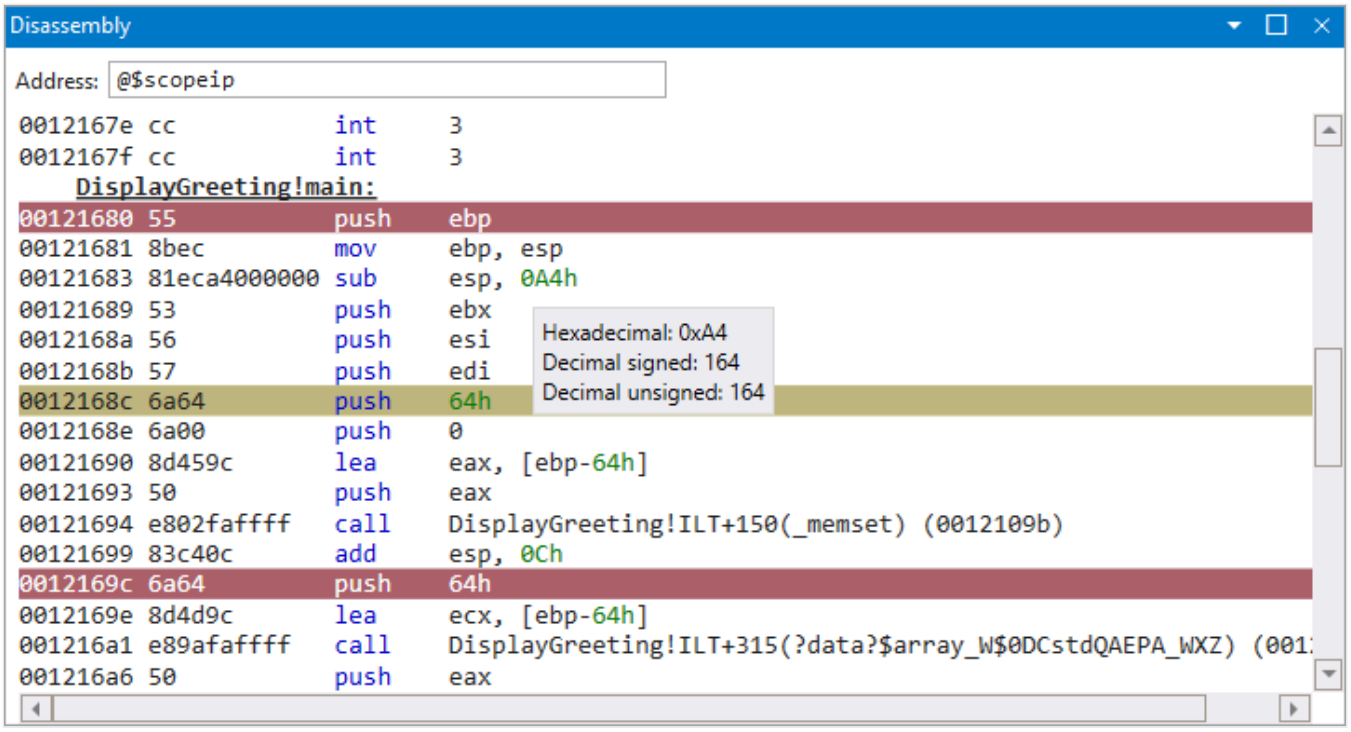
Используйте окно памяти для отображения расположений памяти. Помимо предоставления адреса памяти, можно использовать значения Pseudo-Register, такие как $scopeip и $eventip для проверки памяти. Для использования значений псевдо-регистров в окне памяти добавьте символ @ перед ними, например `@$scopeip`. Дополнительные сведения см. в Pseudo-Register синтаксисе

# Стек

Используйте окно стека для просмотра текущего стека вызовов. Окно стека обеспечивает базовое выделение текущего кадра.

# Разборка

Окно дизассемблирования выделяет текущую инструкцию и сохраняет эту позицию при прокрутке.

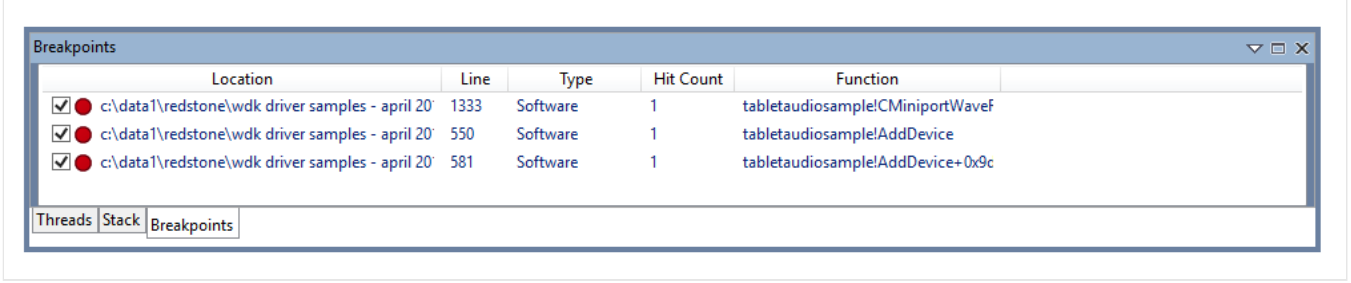



# Потоки

Окно потоков выделяет текущий поток.

# Точки останова

Используйте окно точек останова для просмотра, включения и очистки точек останова.




# Записи

Этот журнал содержит внутренние компоненты WinDbg. Его можно просмотреть для мониторинга длительных процессов и устранения неполадок самого отладчика.

Вы можете продолжить создание традиционного журнала команд отладчика с помощью команды .logopen. Дополнительные сведения об этом см. в статье "Сохранение файла журнала в WinDbg".

# Примечания.

Используйте параметр "Заметки", чтобы открыть окно принятия заметок.

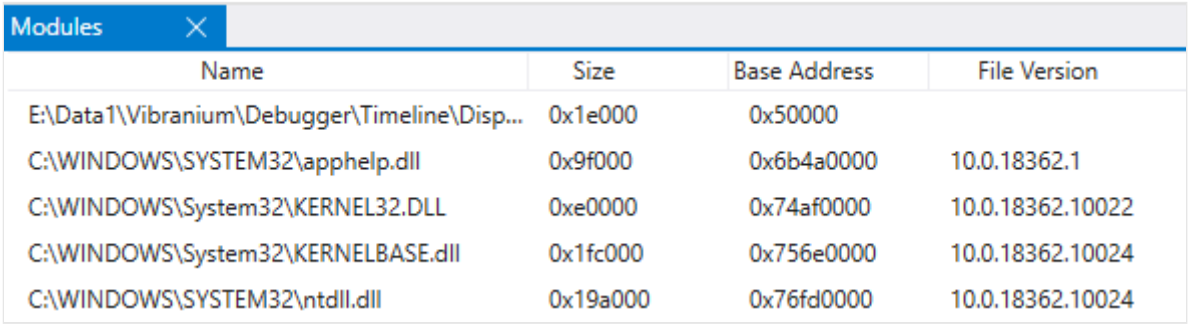
# Временные шкалы

Используйте временную шкалу для открытия или привлечения фокуса к окну временной шкалы. Дополнительные сведения о временных шкалах см. в статье WinDbg — Временная шкала.

# Модули

Используйте модули для отображения загруженных модулей и связанных с ними сведений. В модулях отображается следующее:

- Имя модуля, включая расположение пути
- Размер в байтах загруженного модуля
- Базовый адрес, по которому загружается модуль
- Версия файла




# Макеты

Используйте раскрывающемся меню "Макеты", чтобы выбрать из трех макетов окна.

# Сбросить Windows

Используйте эту функцию для сброса окон отладчика до их позиций по умолчанию.

# Цвет акцента

Используйте меню вытягивания, чтобы задать цвет акцента для отладчика.

# См. также

Функции WinDbg

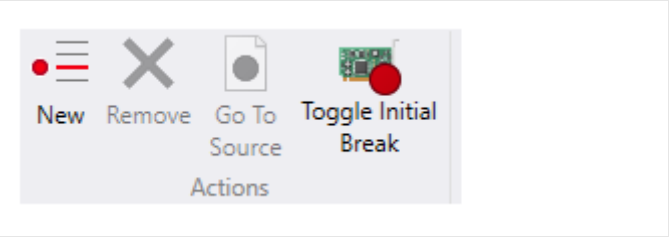
# WinDbg — меню точек останова

18.07.2025

В этом разделе описывается, как работать с точками останова с помощью отладчика WinDbg.

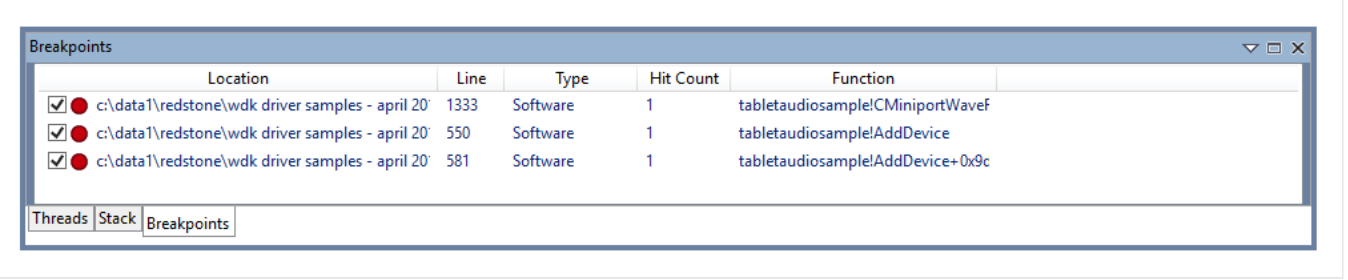
## Меню точек останова

Используйте меню точек останова для создания новых и удаления существующих точек останова, а также переключения начальной точки останова (начальная точка останова в настоящее время является только ядром).




## Окно точек останова

Используйте окно точек останова, открытое в меню "Вид", для просмотра, задания и очистки точек останова. Вы также можете дважды щелкнуть точку останова, чтобы открыть исходный файл.




Окно точки останова ведет постоянно обновляемый счетчик каждого срабатывания точки останова.

Общий процесс работы с точками останова аналогичен предыдущим версиям WinDbg. Дополнительные сведения о настройке точек останова см. в разделе "Настройка точек останова" в WinDbg (классическая модель).

# WinDbg — меню "Путешествие по времени"

18.07.2025



В этом разделе описывается, как работать с меню перемещения по времени в WinDbg.




Дополнительные сведения о путешествии по времени см. в статье "Отладка путешествия по времени" — обзор.

## Трассировка

### Трассировка индекса

Используйте параметр трассировки индекса для принудительного переиндексирования трассировки времени.

### События

Используйте выпадающее меню событий, чтобы отобразить события *исключений* или *загрузки модулей*.

## Позиция

### Начать путешествие во времени

Используйте перемещение по времени, чтобы начать переход к началу файла трассировки временных данных.

### Конец путешествий во времени

Используйте функцию путешествия во времени, чтобы начать переход к концу файла трассировки времени путешествия.

# Прочее

## Информация

Используйте параметр «Информация», чтобы отобразить данные о трассировке, такие как размер и количество потоков.

## Временные шкалы

Используйте параметр "Временная шкала" для доступа к временной шкале отладчика. Дополнительные сведения см. в разделе "Временная шкала WinDbg".

# См. также

Функции WinDbg

# WinDbg — меню модели данных

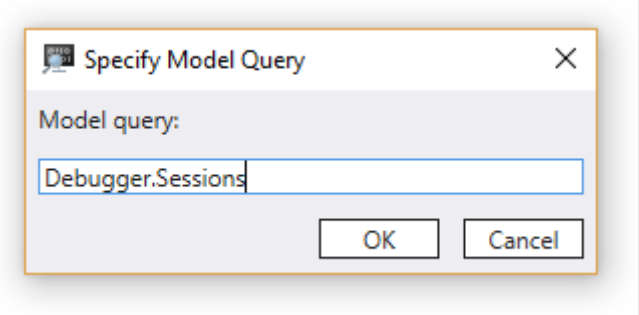
01.07.2025

В этом разделе описывается, как работать с меню модели данных в отладчике WinDbg.

## Запрос новой модели

Используйте диалоговое окно "Запрос новой модели" для создания нового запроса модели. Вы можете разместить здесь все, что вы бы вставили в обычную команду `dx`.

Например, укажите `Debugger.Sessions` , чтобы проверить объекты сеансов отладчика.

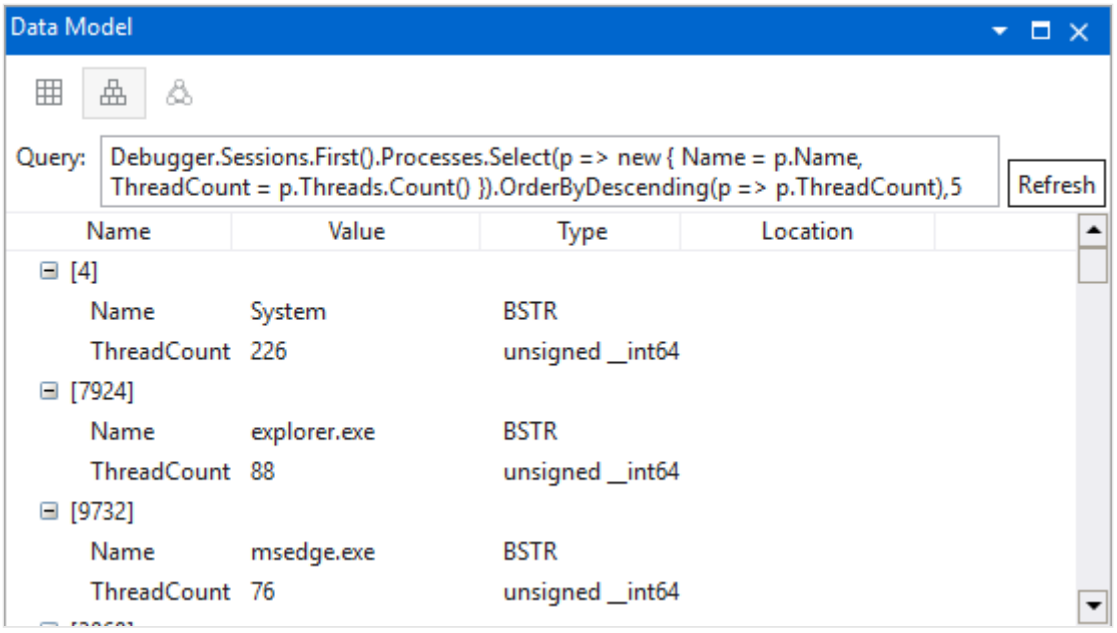



Общую информацию об объектах отладчика можно найти в dx (Отобразить выражение объектной модели отладчика).

Используйте запросы LINQ для более глубокого анализа сеанса. В этом запросе показаны первые 5 процессов, выполняющих большинство потоков.

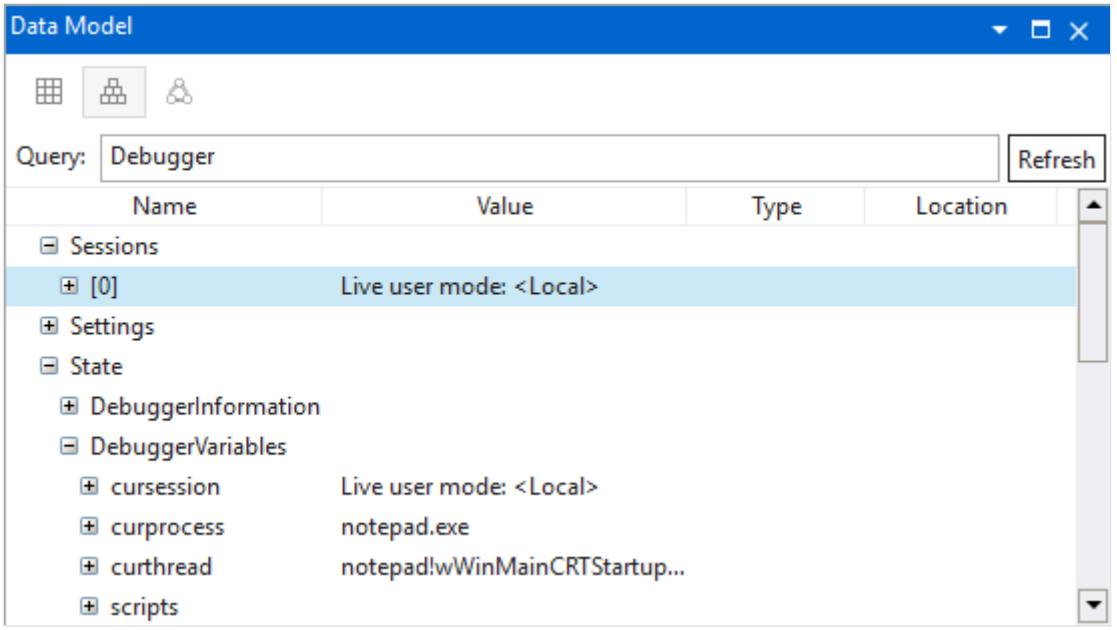
dbgcmd

```
Debugger.Sessions.First().Processes.Select(p => new { Name = p.Name, ThreadCount =
p.Threads.Count() }).OrderByDescending(p => p.ThreadCount),5
```




# Обозреватель моделей данных

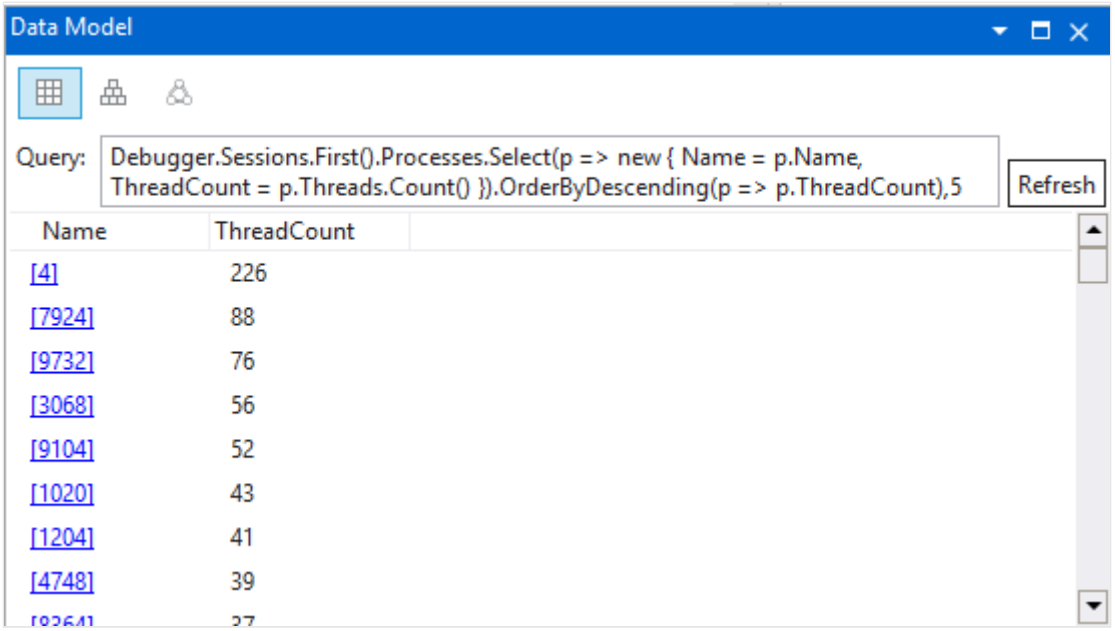
Используйте обозреватель моделей данных для быстрого просмотра каждого объекта модели данных в `Debugger` пространстве имен.




## Режим отображения

Используйте режим отображения для переключения между режимами отображения сетки, иерархии и графа. Вы можете щелкнуть правой кнопкой мыши заголовки столбцов, чтобы скрыть или отобразить больше столбцов.

Режим сетки может быть полезен для углубленного изучения объектов. Например, ниже показан запрос самых популярных тем, использованный ранее, в виде сетки.

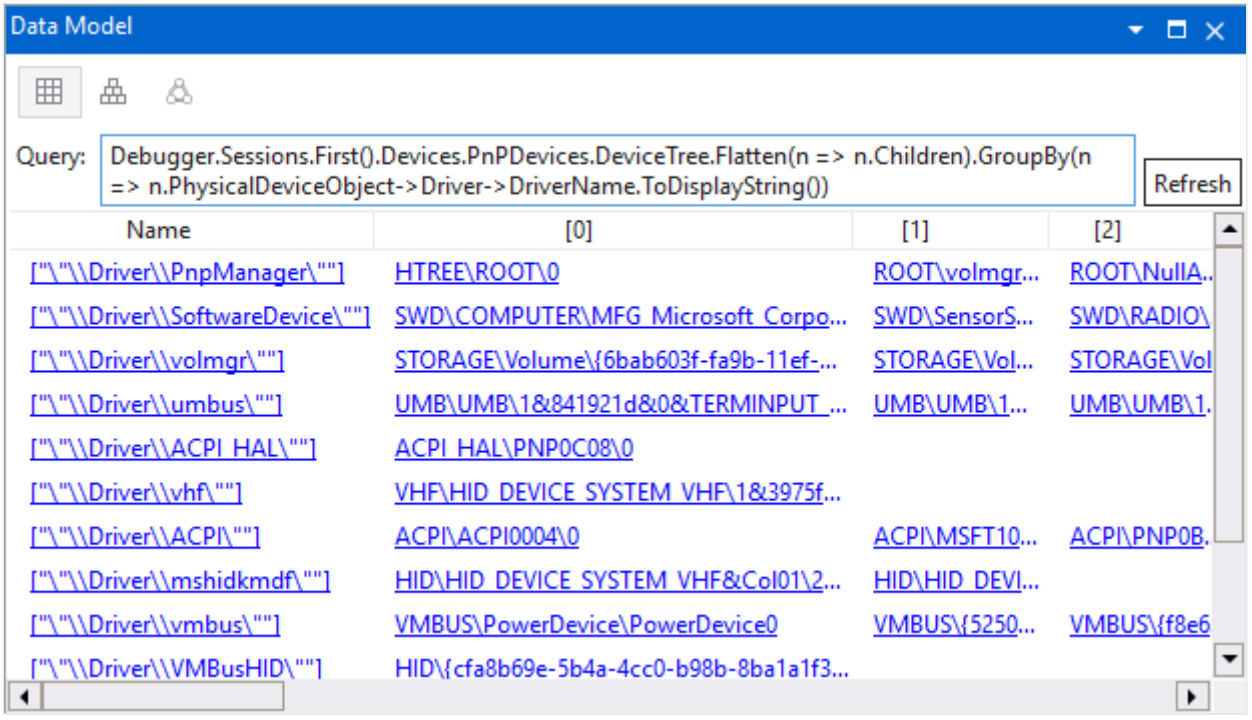



Щелкнув любой подчеркнутый элемент, откроется новая вкладка и выполните запрос, чтобы отобразить эти сведения.

В этом запросе отображаются устройства в дереве подключаемых устройств, сгруппированных по имени драйвера объекта физического устройства для сеанса ядра.

dbgcmd

```
Debugger.Sessions.First().Devices.DeviceTree.Flatten(n => n.Children).GroupBy(n =>
n.PhysicalDeviceObject->Driver->DriverName.ToDisplayString())
```

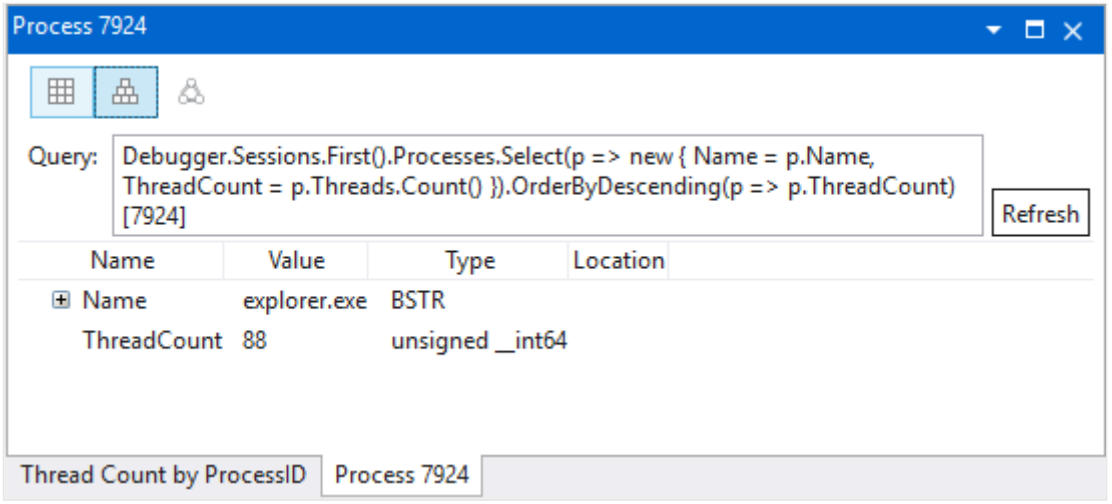



## Изменение запроса

Измените текстовое поле запроса, чтобы изменить запрос, используемый в окне активной модели данных.

## Изменение окна или заголовка вкладки

Новые окна универсальной модели данных задаются `Data Model` заголовком, но можно настроить заголовок по желанию, вызвав `Change Title` элемент контекстного меню из заголовка вкладки или окна.




> **Примечание**
>
> Заголовок не может быть пустым или содержать запятую.

---

# См. также

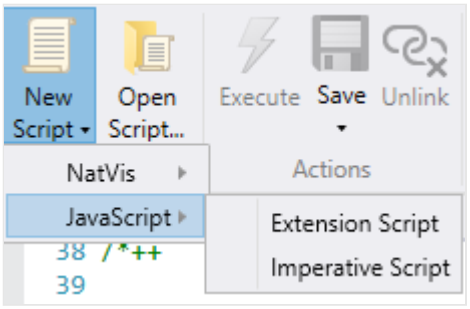
dx (выражение объектной модели отладчика)

Функции WinDbg

# WinDbg — меню сценариев

18.07.2025

В этом разделе описывается использование поддержки сценариев в WinDbg.




В окне скрипта WinDbg представлены подсветка синтаксиса, IntelliSense и распознавание ошибок.

Используйте кнопки ленты, чтобы:

- Создание скрипта
- Открытие существующего скрипта
- Выполнение скрипта
- Сохранение скрипта
- Отключить скрипт

Вы также можете автоматически выполнять скрипты, щелкнув правой кнопкой мыши в окне скрипта и выбрав "*Выполнить скрипт*" *в разделе* "*Сохранить*". При успешной загрузке скрипта в строке заголовка скрипта появится зеленый флажок. Если в скрипте возникают ошибки, будет отображаться красный x.

## Сценарии JavaScript

Чтобы начать работу с JavaScript, сначала необходимо выполнить отладку целевого объекта. Когда вы будете готовы начать работу с JavaScript, нажмите кнопку "Загрузить поставщика JavaScript". После этого можно создать новый JavaScript, выбрав между этими двумя типами шаблонов скриптов.

- **Скрипт расширения — скрипт** , который предназначен для действия в качестве расширения для отладчика. Он управляет объектной моделью отладчика и обеспечивает непрерывную функциональность. Никаких действий при нажатии кнопки "*Выполнить* " на ленте не происходит.

- **Императивный скрипт — скрипт** , предназначенный для выполнения действия каждый и каждый раз, когда кнопка "*Выполнить* " нажимается на ленту. Такой

скрипт обычно не изменяет объектную модель отладчика.

Дополнительные сведения о работе с JavaScript см. в следующих разделах:

Скриптинг для отладки JavaScript

собственные объекты отладчика в расширениях JavaScript

Примеры скриптов отладчика JavaScript

# Скрипты NatVis

Используйте **новый скрипт**>**NatVis** , чтобы открыть следующий пустой шаблон NatVis.

XML

```
<AutoVisualizer
xmlns="https://schemas.microsoft.com/vstudio/debugger/natvis/2010">
  <Type Name="">
  </Type>
</AutoVisualizer>
```

Дополнительные сведения о работе с NatVis см. в разделе "Объекты отладчика" в NatVis.

---

# См. также

Функции WinDbg

# WinDbg — заметки, команды, память и исходные меню

18.07.2025

В этом разделе описывается, как работать с меню Notes, Command, Memory и Source в WinDbg.

## Примечания.

Используйте меню, чтобы:

- Открытие файла заметок
- Сохранение файла заметок

## командование

Используйте меню команд для:

- Предпочитать DML
- Выделение и отмена выделения текущего текста (CTRL+ALT+H)
- Очистка текста командного окна
- Сохранение текста окна в DML-файл

## Память

Используйте меню памяти, чтобы:

- Настройка запроса памяти модели данных
- Задайте размер памяти, например байт или длину
- Задайте формат отображения, например шестнадцатеричное или подписанное
- Задайте формат отображения текста, например ASCII

## Исходный материал

Используйте исходное меню, чтобы:

- Открытие исходного файла
- Установка указателя команд
- Запуск на курсор

- Закройте все исходные окна

## См. также

Функции WinDbg

# WinDbg — ограниченный режим

18.07.2025

## Ограниченный режим



В этом разделе описывается, как включить функцию ограниченного режима, которая ограничивает тип сеансов отладки, которые можно запустить.

WinDbg предоставляет возможность запуска различных типов сеансов отладки. Однако в некоторых случаях может потребоваться, чтобы WinDbg не мог запускать определенные сеансы отладки. Ограниченный режим ограничивает сеансы отладки WinDbg до только удаленных сеансов отладки или загрузки файлов дампа.

Ограниченный режим может быть включен с помощью политики Windows Defender Application Control (WDAC) или с помощью ключа реестра.

## Настройка с помощью элемента управления приложениями Защитника Windows (WDAC)

Режим ограниченного доступа можно включить политикой управления приложениями в Защитнике Windows (WDAC). Политика WDAC может запретить локальным администраторам изменять параметры политики после развертывания политики. Чтобы включить ограниченный режим политикой WDAC, настройте политику с помощью следующего параметра:

XML

```
<Settings>
    <!-- Other settings -->
    <Setting Provider="Microsoft.WindbgX" Key="Settings"
ValueName="EnableRestrictedMode">
        <Value>
            <Boolean>true</Boolean>
        </Value>
    </Setting>
</Settings>
```

# Настройка с помощью раздела реестра

Чтобы включить ограниченный режим через ключ реестра, задайте для указанного ключа `HKLM\SOFTWARE\Microsoft\WinDbg\EnableRestrictedMode` реестра значение DWORD 1.

В этом примере показано, как использовать команду reg add , чтобы добавить ключ и задать для него значение 1.

PowerShell

```
PS C:\WINDOWS\system32> reg add HKLM\SOFTWARE\Microsoft\WinDbg /v
EnableRestrictedMode /t REG_DWORD /d 1
The operation completed successfully.
```

# См. также

Функции WinDbg

WinDbg — параметры запуска командной строки

# Основы WinDbg

18.07.2025



Развернуть таблицу

| Название | Описание |
| --- | --- |
| WinDbg — заметки о выпуске | Новые возможности WinDbg |

# Модель данных отладчика

Развернуть таблицу

| Название | Описание |
| --- | --- |
| команда dx | Интерактивная команда для отображения выражения объектной модели отладчика |
| Использование LINQ С объектами отладчика | язык запросов, подобный SQL |
| собственные объекты отладчика в NatVis | Использование объектов с NatVis |
| WinDbg — модель данных | Использование встроенной поддержки модели данных в WinDbg |

# Расширение модели данных

Развернуть таблицу

| Название | Описание |
| --- | --- |
| Скриптинг для отладки JavaScript | Как использовать JavaScript для создания скриптов, которые понимают объекты отладчика |
| WinDbg — скрипты | Использование Встроенного скрипта WinDbg |
| https://github.com/Microsoft/WinDbg-Samples | Сайт команды отладчика GitHub, где они используют последний пример кода JavaScript (и C++). |

| Название | Описание |
| --- | --- |
| собственные объекты отладчика в расширениях JavaScript | Описывает, как работать с общими объектами и предоставляет справочные сведения об их атрибутах и поведении. |

# Основы TTD

Развернуть таблицу

| Название | Описание |
| --- | --- |
| Отладка временных путешествий — обзор | Обзор TTD |
| Отладка перемещения по времени — пошаговое руководство по приложению | Чтобы попробовать путешествия во времени, проверьте это руководство. |

# Запросы TTD

Развернуть таблицу

| Название | Описание |
| --- | --- |
| Общие сведения о объектах отладки временных путешествий. | Модель данных можно использовать для запроса трассировок времени перемещения. |
| https://github.com/Microsoft/WinDbg-Samples/blob/master/TTDQueries/tutorial-instructions.md | Руководство по отладке кода C++ с помощью запросов TTD для поиска проблемного кода |
| https://github.com/Microsoft/WinDbg-Samples/tree/master/TTDQueries/app-sample | Здесь доступен весь код, используемый в лаборатории. |

# Видео

Смотрите эти эпизоды Defrag Tools, чтобы увидеть WinDbg в действии.

Развернуть таблицу

| Название | Описание |
| --- | --- |
| Средства дефрагментации | Тим, Чад и Энди рассматривают основы WinDbg и некоторые из |

| Название | Описание |
| --- | --- |
| #182 | функций. |
| Defrag Tools #183 | Ник, Тим и Чад используют WinDbg и проводят быструю демонстрацию |
| Средства дефрагментации #184 | Билл и Эндрю проходят по функциям сценариев в WinDbg |
| Defrag Tools #185 | Джеймс и Иветта предоставляют вводные сведения об отладке временных путешествий |
| Средства дефрагментации #186 | Джеймс и JCAB рассматривают расширенную отладку с перемещением во времени |

# Установка и подключение

Развернуть таблицу

| Название | Описание |
| --- | --- |
| WinDbg — установка | Направления установки |
| WinDbg — запуск сеанса в пользовательском режиме | Режим пользователя |
| WinDbg — запуск сеанса режима ядра | Режим ядра |

# Отладка временных путешествий — обзор

Статья • 15.11.2024



## Что такое отладка временных путешествий?

Отладка перемещения по времени — это средство, позволяющее записывать трассировку процесса при выполнении, а затем воспроизводить его позже как вперед, так и назад. Отладка временных путешествий (TTD) помогает упростить отладку проблем, позволяя "перемотка" сеанса отладчика вместо необходимости воспроизводить проблему, пока не найдете ошибку.

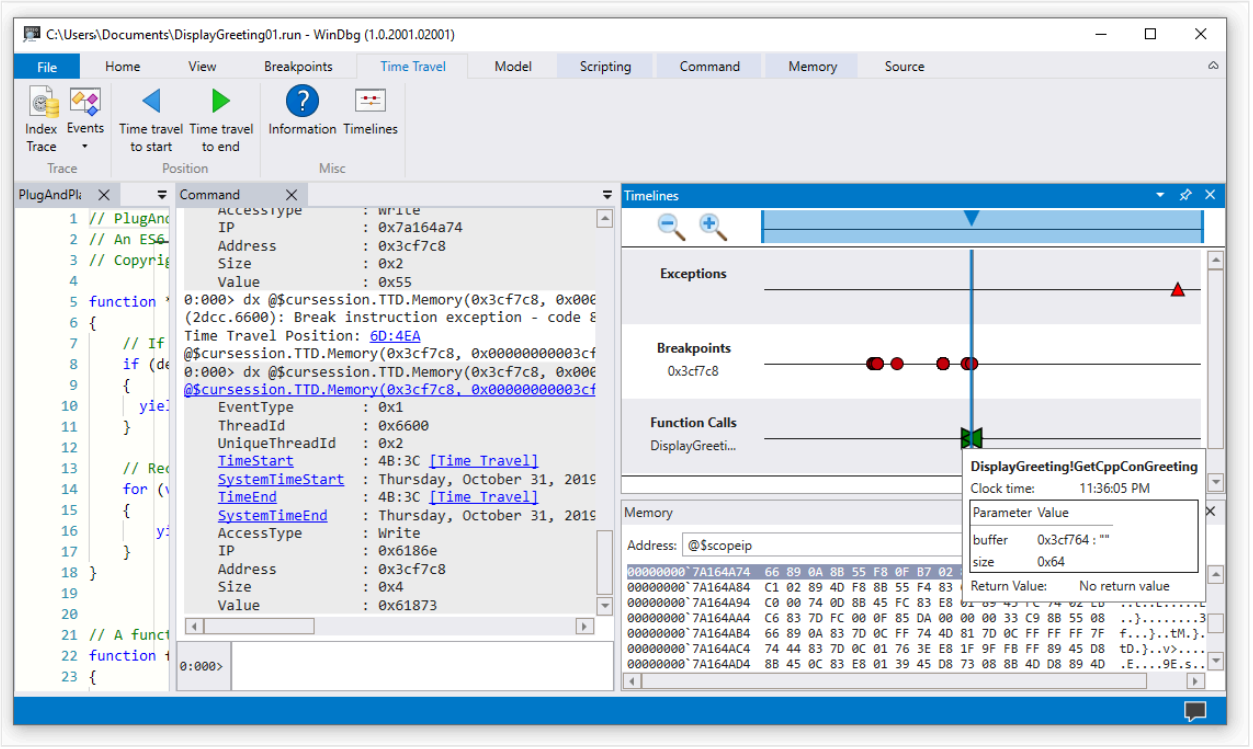
TTD позволяет вам вернуться к времени, чтобы лучше понять условия, которые приводят к ошибке и воспроизвести его несколько раз, чтобы узнать, как лучше всего устранить проблему.

TTD может иметь преимущества по сравнению с файлами аварийного дампа, которые часто пропускают состояние и путь выполнения, что привело к окончательному сбою.

В случае, если вы не можете выяснить проблему самостоятельно, вы можете поделиться трассировкой с коллегой, и они могут посмотреть точно то, что вы смотрите. Это позволяет упростить совместную работу, чем динамическую отладку, так как записанные инструкции одинаковы, в то время как расположения адресов и выполнение кода будут отличаться на разных компьютерах. Вы также можете поделиться определенной точкой во времени, чтобы помочь вашему коллеге выяснить, где начать.

TTD является эффективным и работает, чтобы добавить как можно меньше накладных расходов, так как он фиксирует выполнение кода в файлах трассировки.

TTD включает набор объектов модели данных отладчика, позволяющий запрашивать трассировку с помощью LINQ. Например, можно использовать объекты TTD для поиска при загрузке определенного модуля кода или поиска всех исключений.




## Требования

Отладка перемещения по времени интегрирована с WinDbg, обеспечивая простой процесс записи и воспроизведения.

Чтобы использовать TTD, необходимо запустить отладчик с повышенными привилегиями. Установите WinDbg с помощью учетной записи с правами администратора и используйте эту учетную запись при записи в отладчике. Чтобы запустить отладчик с повышенными привилегиями, выберите и удерживайте (или щелкните правой кнопкой мыши) значок WinDbg в меню , а затем выберите "**Больше > запуска от имени администратора**".

## Заметки о выпуске

TTD продолжает развиваться, чтобы получить последние сведения, см . заметки о выпуске отладки по времени.

## Запись может содержать личную или связанную с безопасностью информацию

Созданный файл трассировки, содержащий запись, может содержать личную или связанную с безопасностью информацию, включая, но не обязательно ограничивается путями к файлам, реестром, памятью или содержимым файла. Точные сведения зависят от целевого действия процесса во время записи.

Помните об этом при совместном использовании файлов записи с другими людьми.

## служебная программа записи TTD.exe командной строки

Помимо записи трассировок в пользовательском интерфейсе WinDbg, доступна TTD.exe служебная программа командной строки для записи трассировки.

У вас могут быть сценарии, в которых требуется только средство записи командной строки TTD: запись на компьютере без установки отладчика, расширенных сценариев записи, автоматизации тестирования и т. д. В этих сценариях можно установить только средство записи командной строки TTD с помощью URL-адреса. Дополнительные сведения см. в статье "Отладка перемещения по времени" — TTD.exe служебная программа командной строки.

## Сравнение средств отладки

В этой таблице перечислены преимущества и минусы различных доступных решений отладки.

Развернуть таблицу

| Подход | Плюсы | Минусы |
|---|---|---|
| Динамическая отладка | Интерактивный интерфейс, просматривает поток выполнения, может изменить целевое состояние, знакомый инструмент в знакомом параметре. | Нарушение взаимодействия с пользователем может потребовать усилий для повторного воспроизведения проблемы, может повлиять на безопасность, но не всегда вариант для рабочих систем. При повторном выполнении трудно вернуться с точки сбоя, чтобы определить причину. |
| Подавленное состояние | Не кодирование заранее, низкой навязчивости на основе триггеров. | Последовательные моментальные снимки или динамические дампы предоставляют простое представление "с течением времени". Накладные расходы, по сути, равны нулю, если они не используются. |
| Данные телеметрии и журналы | Упрощенный, часто связанный с бизнес-сценариями / действиями | Проблемы возникают в непредвиденных путях кода (без телеметрии). Отсутствие глубины данных, статически скомпилированное в код. |

| Подход | Плюсы | Минусы |
| --- | --- | --- |
| | пользователей, понятным машинным обучением. | |
| Отладка временных путешествий (TTD) | Отлично подходит для сложных ошибок, без написания кода заранее, автономной повторяемой отладки, анализа понятно, захватывает все. | Большие издержки во время записи. Может собирать больше данных, необходимых. Файлы данных могут стать большими. |

# Видео

Дополнительные сведения о TTD см. в этих видео.

Defrag Tools 185 — Ivette и JamesP переходят по основам TTD и демонстрации некоторых функций в WinDbg

Дефрагментация инструментов 186 — Jordi и JCAB демонстрация более отличных функций TTD в WinDbg

CppCon (YouTube) - Jordi, Кен и ДжеймсМ представили TTD в WinDbg в CppCon 2017

# Основы файла трассировки

## Размер файла трассировки

Файл трассировки может получить большой размер, и пользователю TTD необходимо убедиться, что есть достаточно свободное место. Если вы записываете программу даже через несколько минут, файлы трассировки могут быстро расти до нескольких гигабайтов. TTD не задает максимальный размер файлов трассировки, чтобы обеспечить сложные длительные сценарии. Быстрое повторное создание проблемы позволит максимально уменьшить размер файла трассировки.

## Файлы трассировки и индекса

Файл трассировки (`.run`) сохраняет выполнение кода во время записи.

После остановки записи создается файл индекса (`.idx`) для оптимизации доступа к данным трассировки. Файлы индекса также создаются автоматически, когда WinDbg открывает файлы трассировки.

Файлы индекса также могут быть большими, как правило, в два раза больше, чем файл трассировки.

Вы можете повторно создать файл индекса из файла трассировки с помощью `!tt.index` команды.

```
dbgcmd
```

```
0:000> !tt.index
Successfully created the index in 10ms.
```

Ошибки записи и другие выходные данные записи записываются в файл журнала WinDbg.

Все выходные файлы хранятся в расположении, настроенном пользователем. Расположение по умолчанию находится в папке документов пользователей. Например, для User1 файлы TTD будут храниться здесь:

```
Консоль
```

```
C:\Users\User1\Documents
```

Дополнительные сведения о работе файлов трассировки см. в статье "Отладка по пути времени" — работа с файлами трассировки.

# Вещи, чтобы искать

## Несовместимость с вирусом

Вы можете столкнуться с несовместимостью из-за того, как TTD перехватчики в процесс записывают их. Обычно возникают проблемы с антивирусным или другим системным программным обеспечением, которое пытается отслеживать и теневые вызовы памяти системы. Если возникают проблемы с записью, например недостаточное сообщение о разрешении, попробуйте временно отключить любое антивирусное программное обеспечение.

Другие служебные программы, которые пытаются заблокировать доступ к памяти, также могут быть проблемными, например, набор средств microsoft Enhanced Mitigation Experience Toolkit.

Другим примером среды, которая конфликтует с TTD, будет платформа электронных приложений. В этом случае трассировка может записываться, но

также возможна взаимоблокировка или сбой записываемого процесса.

## Только в пользовательском режиме

В настоящее время TTD поддерживает только операцию в пользовательском режиме, поэтому трассировка процесса режима ядра невозможна.

## Воспроизведение только для чтения

Вы можете вернуться к времени, но вы не можете изменить историю. Вы можете использовать команды памяти для чтения, но нельзя использовать команды, которые изменяют или записывают в память.

## Системные защищенные процессы

Некоторые защищенные системы Windows процессы, такие как защищенный процесс light (PPL), защищены, поэтому TTD не может внедриться в защищенный процесс, чтобы обеспечить запись выполнения кода.

## Влияние записи на производительность

Запись приложения или процесса влияет на производительность компьютера. Фактические затраты на производительность зависят от объема и типа кода, выполняемого во время записи. Вы можете ожидать около 10x-20x производительности в типичных сценариях записи. Иногда не будет заметного замедления пользовательского интерфейса. Но для более ресурсоемких операций, таких как диалоговое окно "Открыть файл", вы увидите влияние записи.

## Ошибки файла трассировки

В некоторых случаях могут возникать ошибки файла трассировки. Дополнительные сведения см. в статье "Отладка перемещения по времени" — устранение неполадок.

# Расширенные возможности отладки перемещения по времени

## Временная шкала

Временные шкалы — это визуальное представление событий, происходящих во время выполнения. Эти события могут быть расположениями: точки останова, чтение и запись памяти, вызовы функций и возвраты, а также исключения. Дополнительные сведения о временных шкалах см. в статье WinDbg — Временная шкала.

## Поддержка модели данных отладчика

- **Встроенная поддержка** модели данных — TTD включает поддержку модели данных. Использование запросов LINQ для анализа сбоев приложений может быть мощным инструментом. Окно модели данных в WinDbg можно использовать для работы с расширяемой и просматриваемой версией dx и dx -g, позволяя создавать таблицы с помощью запросов NatVis, JavaScript и LINQ.

Общие сведения о модели данных отладчика см. в статье WinDbg — модель данных. Дополнительные сведения о работе с объектной моделью отладчика TTD см. в статье "Отладка временных путешествий" — общие сведения о объектах отладки временных путешествий.

## Поддержка сценариев

- **Автоматизация** сценариев — поддержка сценариев для JavaScript и NatVis позволяет выполнять автоматизацию исследования проблем. Дополнительные сведения см. в статье "Отладка перемещения по времени" — служба автоматизации JavaScript.

Общие сведения о работе с JavaScript и NatVis см. в статье WinDbg — Scripting.

## служебная программа командной строки TTD.exe

Доступна программа командной строки TTD.exe для записи трассировок. Дополнительные сведения см. в статье "Отладка перемещения по времени" — TTD.exe служебная программа командной строки.

## Поддержка управляемого кода TTD

Расширение отладки SOS (sos.dll), работающее в 64-разрядном режиме, можно использовать для отладки управляемого кода с помощью TTD в WinDbg. Дополнительные сведения см. в разделе Отладка управляемого кода с помощью отладчика Windows.

# Начало работы с TTD

Ознакомьтесь с этими разделами, чтобы записать и воспроизвести файл трассировки, а также сведения о работе с файлами трассировки и устранение неполадок.

- Отладка перемещения по времени — запись трассировки
- Отладка перемещения по времени — воспроизведение трассировки
- Отладка перемещения по времени — работа с файлами трассировки
- Отладка временных путешествий — устранение неполадок
- Отладка перемещения по времени — пошаговое руководство по приложению

В этих разделах описываются дополнительные расширенные функциональные возможности отладки во время перемещения.

- Отладка временных путешествий — введение в объекты отладки временных путешествий
- Отладка перемещения по времени — автоматизация JavaScript

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Заметки о релизе отладки путешествия во времени

23.06.2025



В этом разделе содержатся сведения о новых возможностях отладки временных путешествий.

## 1.11.532

Это техническое обновление, которое делает более надежным процесс записи. Этот выпуск TTD совпадает с июньским выпуском Windbg 2025 года. Одна из новых возможностей — объект модели данных Position теперь отображает процент в трассировке.

### Изменено

- Прочее обслуживание инфраструктуры.
- Используйте реализацию стандарта ISO для ключевого слова volatile в C++. (1.11.518)
- Добавьте процент к проекции модели данных "Позиция". (1.11.514)

### Исправлено

- Увеличьте надежность обработки декодированных инструкций TTD. (1.11.530)
- Удалите использование XSAVE в эмуляторе и оптимизируйте внутреннюю передачу регистра. (1.11.509)

## 1.11.506

Это небольшой выпуск, который совпадает с выпуском WinDbg в апреле 2025 года.

### Изменено

- Щелкните позицию TTD в модели данных (например, в списке модулей), чтобы перейти к этой позиции в трассировке. (1.11.492)

  Примечание. В командном окне не будет отображаться обновленная позиция TTD до выполнения следующего шага или выполнения команды.

## Исправлено

- Добавьте имя процесса в .out файл при присоединении к PID для облегчения устранения неполадок. (1.11.486)

# 1.11.481

Мы обновили команду !tt, чтобы предоставить вам более эффективные способы перехода по трассировке:

- Дробные проценты можно использовать для сужения пространства поиска (!tt 23.65)
- Найдите предыдущее/следующее время изменения значения регистра (!tt br ebx)
- Найдите предыдущий или следующий раз, когда доступ к диапазону памяти осуществляется (!tt ba- [addr] [range])
- Найдите предыдущий или следующий раз выполнения, когда процесс переходит к другому модулю (!tt bm)
- Найдите предыдущее/следующее выполнение при переходе к определенному модулю (!tt bm ntdll)

Дополнительные сведения см. в разделе !tt (путешествие по времени).

Некоторые важные исправления:

- Пропали сообщения об ошибке: "64-разрядное значение теряет точность при преобразовании в число" при использовании `@$cursession.TTD.Data.Heap()` на 32-разрядной трассировке.
- Анализ параметра справки (`-?`, `-help`) теперь правильно обнаружен в любой точке командной строки.
- `dx @$cursession.TTD.Calls()` больше не требует, чтобы адреса соответствовали началу функции. Вместо этого адрес будет сопоставлен с началом ближайшей функции сопоставления.
- TTD правильно сообщает целевую версию ОС из команды vertarget.
- Использование "-monitor" с именем хостинговой службы перестало записывать несвязанные хостинговые службы.

## Добавлено

- Регистрация точек останова в трассировках TTD (1.11.431)

## Изменено

- Исправлена запись служб по имени с помощью режима монитора (1.11.477)
- Захват сведений о ОС фактической целевой системы для использования отладчиком (1.11.473)
- Исправлена передача регистров XMM между эмулятором и CONTEXT (1.11.469)
- Разрешить вызовы по адресам внутри функции (1.11.459)
- Поддержка символов в виде адресов и размеров в командной строке !tt (1.11.454)
- Повышение согласованности и расширение возможностей команд навигации TTD (1.11.453)
- Повышение согласованности базы данных модуля в условиях поврежденных данных (1.11.430)

## Исправлено

- Добавление имени процесса в выходные данные при присоединении к PID (1.11.486)
- TTD. Data.Heap() сообщает "Ошибка: 64-разрядное значение теряет точность при преобразовании в число" в некоторых случаях (1.11.471)
- Улучшите надежность записи процесса с включенными теневыми стеками (1.11.466)
- Добавление навигации по модулю через !tt bm и модель данных (1.11.462)
- Исправлены некоторые проблемы при синтаксическом анализе командной строки. (1.11.444)
- Исправление lodsd, загрузка двойного слова по адресу (обнуление верхней части rax) (1.11.434)
- Исправлены некоторые ошибки libfuzzer (1.11.433)

# 1.11.429

Это обновление TTD содержит несколько исправлений ошибок, а также некоторые внутренние изменения для повышения надежности.

Примечание: в версии 1.11.410 была введена регрессия в эмуляцию инструкции Intel/AMD LODSD. Исправление для этого появится в следующем выпуске.

Исправления:

- Улучшение устойчивости чтения пакетов и других различных изменений для повышения надежности.
- Исправлена регрессия при эмуляции инструкции AVX VBROADCAST[I/F]128.
- Исправлен доступ к записи исключений в ARM64 в более новых сборках Windows.

# 1.11.410

Улучшена доступность: Интерфейс прогресса теперь правильно масштабируется при изменении размера текста.

Команда `@$cursession.TTD.Calls()` в отладчике теперь поддерживает подстановочные знаки, соответствующие большому количеству функций. Теперь можно запрашивать большое количество функций (`@$cursession.TTD.Calls("kernel32!*")`).

Автоматизация: новый `-onMonitorReadyEvent` параметр командной строки указывает, когда монитор записи (`-monitor` коммутатор) готов к записи новых процессов.

Исправления:

- Исправьте некоторые условия гонки при инициализации записывателя.
- Исправьте способ записи системных вызовов, чтобы точки останова работали правильно.
- Устранение нескольких проблем, связанных с выборочной записью модуля.

Исправления для ARM64:

- Исправлена ошибка, предотвращающая запись TTD на обычных ЦП уровня ARM64v8.0.
- Улучшено обмен сообщениями при попытке использовать в ARM64 трассировку процесса x86 или x64.

Исправления AMD/Intel (включает некоторые проблемы, сообщаемые Google):

- Исправлена неправильная эмуляция LODS: вместо отсчета неиспользуемых битов RAX они теперь сохраняются правильно.
- Исправлена эмуляция инструкции pop ax в процессах x86/x64, которая неправильно обнуляла верхние биты регистра (например, "pop ax" очищала верхние биты rax).
- Прямая эмуляция инструкции XGETBV (быстрее).
- Прямая эмуляция всех перемещений AVX512 SIMD (быстрее).

# 1.11.316

Исправлена регрессия, которая приводила к случайным сбоям при записи программ с длительными непрерывными последовательности инструкций с большим объемом данных.

Исправления для ARM64:

- Запись в процессах ARM64 с включенной функцией PAC теперь поддерживается.
- Исправлены инструкции ANDS и TST, которые не смогли очистить флаги переноски и переполнения.

Исправления AMD/Intel:

- Исправлена ошибка, из-за которой TTD неправильно эмулировал "xchg r8,rax" и "xchg r8w, ax" как NOP.

# 1.11.304

TTD теперь реализует и публикует общедоступный API для управления записью во время активного процесса записи. Документация и пример можно найти в GitHub.

Теперь TTD может интегрироваться с отключенной записью с помощью нового `-recordMode` переключателя. По умолчанию TTD использует `-recordMode Automatic` , что приводит к записи всех потоков. Если `-recordMode Manual` задано, TTD внедряется в целевой процесс, но не записывает ничего, пока не будет сказано сделать это через вызов API.

Запись теперь может быть ограничена определенным набором модулей с помощью коммутатора `-module` . В некоторых сценариях это может привести к значительно более быстрой записи и меньшим файлам трассировки. Можно указать несколько `-module` коммутаторов.

Теперь в дистрибутив включены соответствующие компоненты записи и воспроизведения. В случае несовместимости между отладчиком и записывающей программой командной строки или ошибкой при воспроизведении, компоненты воспроизведения можно скопировать в отладчик в качестве временного обходного решения, пока не будет выпущен новый отладчик.

Установленное расположение файла можно найти в PowerShell, выполнив следующие действия.

```
ls (Get-AppxPackage | where Name -eq
```

```
'Microsoft.TimeTravelDebugging').InstallLocation
```

## Добавлено

- Добавьте переключатель -recordmode для разрешения внедрения без автоматической записи (1.11.296)
- Добавление переключателя -module и использование для создания конфигурации SR (1.11.291)
- Пользовательские данные проекта, записанные API в процессе, в модель данных (1.11.286)
- Добавьте новый TTDLiveRecorder.dll и подключите его вместе с TTDRecordCPU.dll (1.11.283)
- Добавление компонентов воспроизведения в MSIX и исправление поиска пакета SDK (1.11.265)

## Изменено

ничего

## Исправлено

- Обход ошибки в сериализаторе JSON nlohmann, который используется в некоторых внутренних инструментах (1.11.281)
  - Внесли исправление в эту библиотеку, которая станет доступна в будущем выпуске.
- Настройте выравнивание строк, чтобы избежать редкой ошибки CRT (1.11.279)
  - Было сообщено и исправлено в кодовых базах Visual Studio и операционной системы.
- Несколько небольших исправлений из отчетов о сбоях Уотсона (1.11.276)
- Исправлена регрессия, которая может привести к повреждению файла трассировки в некоторых случаях (1.11.264)

## Известные проблемы

- На ARM64 компилятор не может передавать управление ряду функций, вызываемых с высокой частотой, что в крайних случаях может привести к тому, что у рекордера закончится место в стеке и он завершится сбоем.

# 1.11.261

К заметным изменениям в этом выпуске относятся следующие:

- [ARM64] Исправлено поведение инструкций `SXTL`, `SQXTN2`, `SQXTUN2`, `UQXTN2`, `XTN2` и `TRN1`, когда целевой регистр используется в качестве источника.
- [ARM64] Исправлена проблема, из-за которой отладчик отображал регистры SIMD с нижними 64 битами, дублированными в верхние 64 бита.
- [AMD64] Исправления эмуляции AVX512 для процессоров AMD Zen4 (регистры повреждены).

## Изменено

- Реализуйте новую систему управления версиями специально для эмулятора. (1.11.260)

## Исправлено

- Исправьте неисправные инструкции ARM64, в которых регистр назначения также используется в качестве источника. (1.11.261)
- Исправьте обходное решение Zen4 для прямого возврата в эмулятор. (1.11.222)

# 1.11.202

Этот выпуск устраняет ряд проблем, возникающих при запуске служб записи или мониторинга с помощью коммутатора `-monitor` . Она также удаляет поддержку записи ARM32 из продукта.

## Изменено

- Уменьшите двоичный размер, связав компоненты TTD с UCRT в виде библиотеки DLL. (1.11.191)

## Исправлено

- Исправлена запись сервисов. (1.11.193)
- Устранение нескольких проблем при использовании -monitor. (1.11.189)
- Исправлен кадр стека функции, который сохраняет регистры с невольной изменяемостью при выполнении альтернатив на x64 (1.11.188)
- Повторное включение буферизации выходных данных для TTD (1.11.187)
- Исправлена обработка дескриптора GPO в ProcessMonitorServer (1.11.179)

## Удаленные

- Удаление кода записи ARM32 из репозитория (1.11.198)

# 1.11.173

Этот выпуск повышает видимость определенных сообщений об ошибках, извлекая их из файла .out и выводя их на консоль. Он также устраняет редкий сбой во время воспроизведения трассировки.

## Изменено

- Извлечение и вывод сообщений об ошибках из .out файла (1.11.173)

## Исправлено

- Исправьте конфликт файла при чтении .out файла из отдельного процесса. (1.11.171)
- Исправлен редкий сбой при воспроизведении трассировки. (1.11.166)

# 1.11.163

В этом выпуске добавлена поддержка записи процессов x86 на компьютерах x64.

## Изменено

- Исправлена запись x86 с установкой x64 TTD (1.11.163)

## Исправлено

- Очистка EULA (1.11.161)

# 1.11.159

Этот выпуск является первым общедоступным выпуском средства записи командной строки. Наряду с несколькими изменениями, необходимыми для включения общедоступного выпуска средства записи командной строки, этот выпуск также включает ряд исправлений ошибок, включая несколько исправлений в эмулятор ЦП.

Новый `-timestampFileName` коммутатор включает создание файла run на основе метки времени. Это полезно при записи большого количества экземпляров одного и того же

процесса и требуется свести к минимуму время запуска записи.

## Изменено

- Выберите режим внедрения по умолчанию во время выполнения, на основе которого используется трассировщик (1.11.156)
- Добавьте переключатель для включения создания файла .run, основанного на метке времени (1.11.155)
- Добавление EULA и `-accepteula` в TTD (1.11.154)
- Добавление ProcLaunchMon.sys в MSIX (1.11.153)
- Создание на арку MSIX и MSIXBUNDLE (1.11.152)
- Исправьте ряд проблем, которые возникли при тестировании TTD, созданной с помощью Clang. (1.11.146)
- Исправления Clang для TTDAnalyze (1.11.144)

## Исправлено

- Ознакомьтесь с отзывами о приложении appinstaller / общедоступном выпуске (1.11.159)
- Отзыв rc (1.11.157)
- Избегайте перезаписи регистра "Ноль", инициализировав RegisterInfo, чтобы указывать на Sink. (1.11.149)
- Исправьте инструкцию TST с помощью немедленного и усовершенствования модульного теста, чтобы охватить его и многое другое. (1.11.148)
- Консолидация решения о защищенном процессе и отключение использования защищенного процесса (1.11.147)

# 1.11.138

## Изменено

- Создание средства записи MSIX (1.11.138)
- Исправьте все проблемы, чтобы Clang смог создать TTD. (1.11.137)
- Введите -monitor X как способ записи процесса при запуске (1.11.116)

## Исправлено

- Исправить эмуляцию ARM64 "CMP ZR" (1.11.128)
- Исправлена эмуляция AVX512 на процессорах AMD Zen4 (1.11.127)

- Исправлен механизм использования TTD для поиска файлов для определенного ЦП (1.11.121)
- Исправлена регрессия x86 TTD (TTDRecordCPU.dll не загружается) (1.11.110)
- Исправлено возвращение к исходному пути на ARM64, чтобы не повредить X28 (1.11.109)

## См. также

Отладка временных путешествий — обзор

Отладка перемещения по времени — средство записи командной строки

# Отладка перемещения по времени — запись трассировки

Статья • 18.11.2024

В этом разделе описывается, как записывать трассировки отладки по времени (TTD). Существует два способа записи трассировки в WinDbg, *запуска исполняемого файла (дополнительно)* и *присоединения к процессу*.

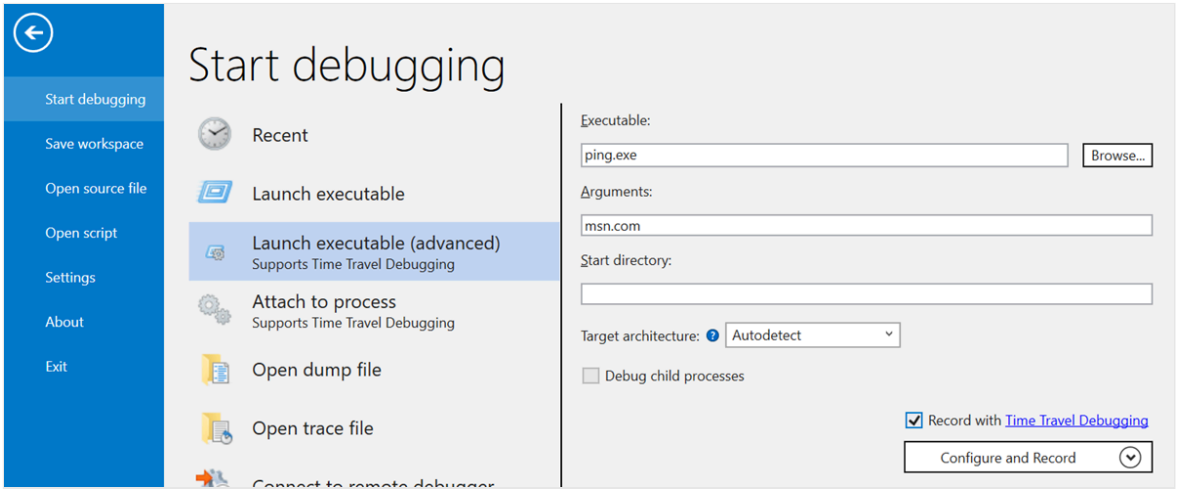
## Запуск исполняемого файла (дополнительно)

Чтобы запустить исполняемый файл и записать трассировку TTD, выполните следующие действия.

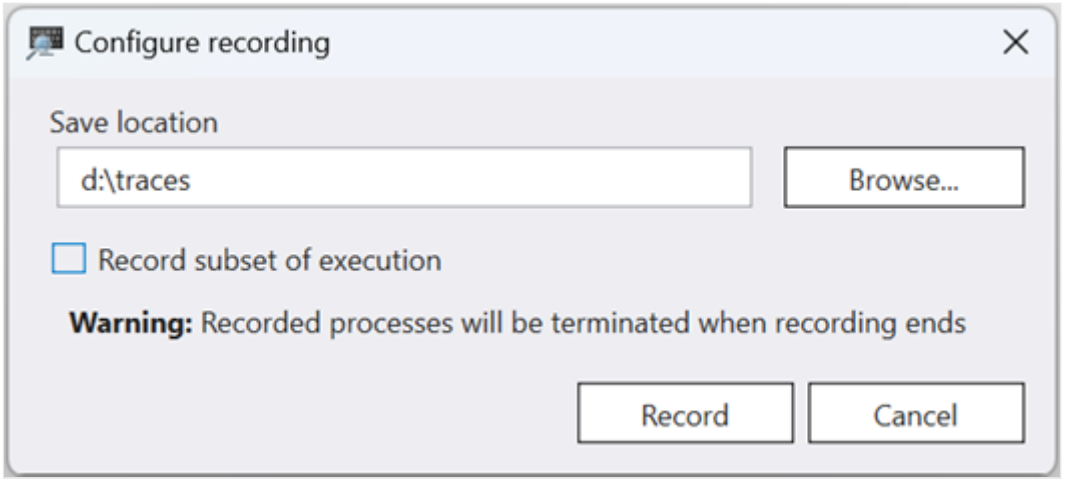
1. В WinDbg выберите **"Запуск запуска>файла" для запуска исполняемого файла>(дополнительно).**

2. Введите путь к исполняемому файлу в пользовательском режиме, который вы хотите записать, или нажмите кнопку **"Обзор** ", чтобы перейти к исполняемому файлу. Сведения о работе с меню "Запуск исполняемого файла" в WinDbg см. в статье "Запуск сеанса в режиме пользователя".

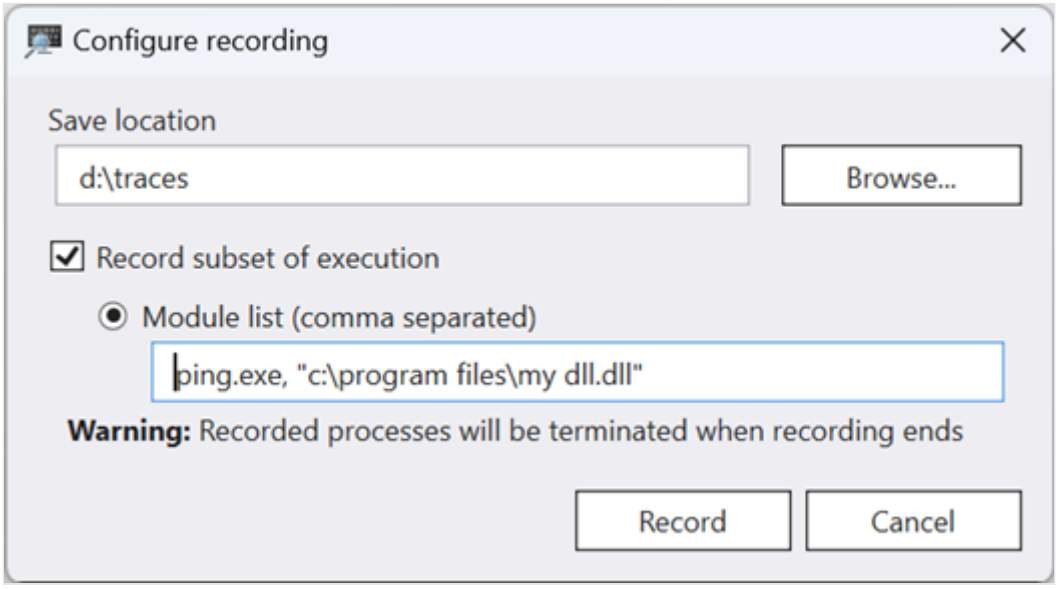
3. **Установите флажок "Запись с отладчиком** по времени", чтобы записать трассировку при запуске исполняемого файла.



4. Если выбрать "**Настроить**" и "**Запись** ", вы сможете настроить расположение для файла трассировки.

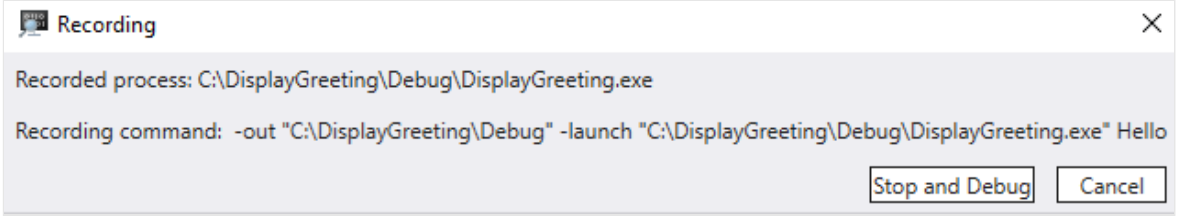



5. Чтобы ограничить запись определенными модулями, проверка "Запись подмножества выполнения" и введите в именах модулей. Например, если требуется записать выполнение notepad.exe, в текстовом поле введите "notepad.exe". Если вы хотите записать выполнение notepad.exe и kernelbase.dll, в текстовом поле введите "notepad.exe,kernelbase.dll".




6. Нажмите кнопку "**OK** ", чтобы запустить исполняемый файл и начать запись.

7. Откроется диалоговое окно записи, показывающее, что трассировка записывается.



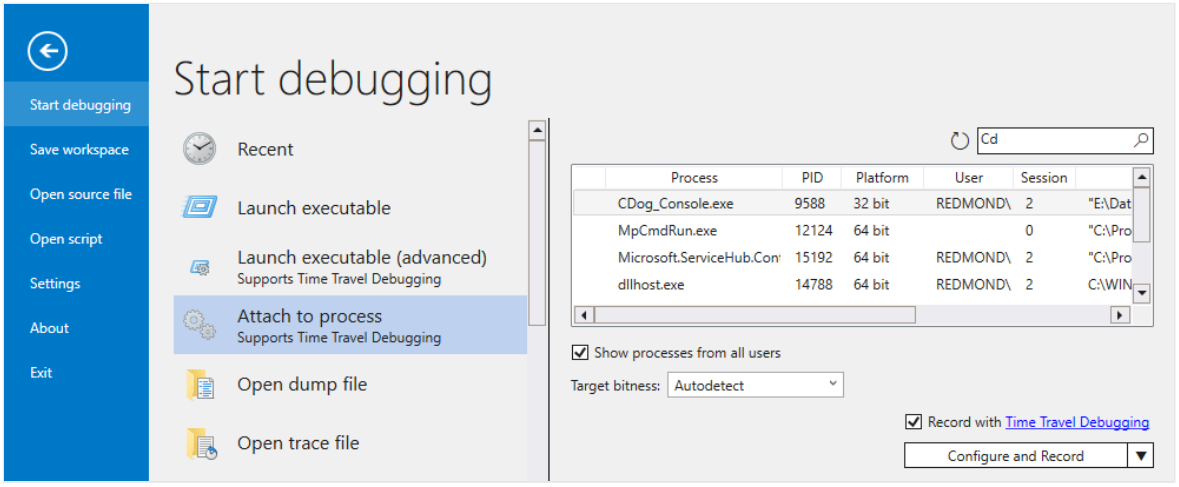
8. Узнайте , как записать сведения о записи.

# Присоединение к процессу

Чтобы подключиться к процессу и записать трассировку TTD, выполните следующие действия.

1. В WinDbg выберите "**Начать отладку>файла>" в процессе.**

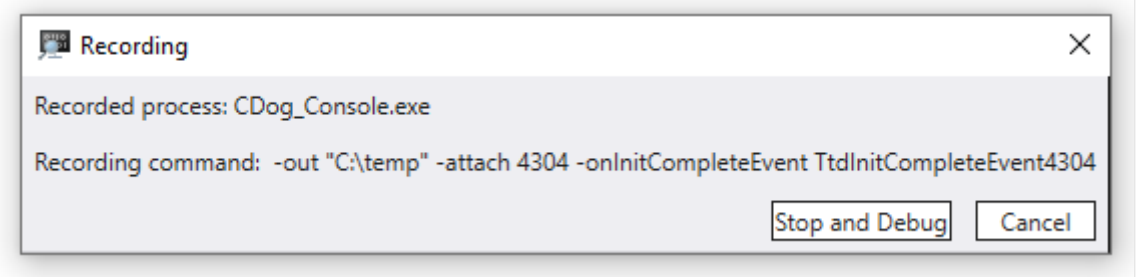
2. Выберите процесс пользовательского режима, который требуется трассировать. Сведения о работе с *подключением к меню процесса* в WinDbg см. в статье WinDbg — запуск сеанса в пользовательском режиме.




3. **Установите флажок "Процесс записи" с полем отладки** "Путешествие по времени", чтобы создать трассировку при запуске исполняемого файла.

4. Нажмите кнопку **"Присоединить** ", чтобы начать запись.

5. Откроется диалоговое окно записи, показывающее, что трассировка записывается.




6. Узнайте , как записать сведения о записи.

# Запись

1. Процесс записывается, поэтому здесь необходимо вызвать проблему, которую вы хотите отлаживать. Вы можете открыть проблемный файл или выбрать определенную кнопку в приложении, чтобы вызвать событие, интересующее вас.

2. При отображении диалогового окна записи можно:

   - **Остановка и отладка** . При выборе этого остановится запись, создайте файл трассировки и откройте файл трассировки, чтобы начать отладку.
   - **Отмена** . При выборе это приведет к остановке записи и созданию файла трассировки. Файл трассировки можно открыть позже.

3. После завершения записи закройте приложение или нажмите клавишу **Stop и debug**.

   > ⓘ **Примечание**
   >
   > Как остановка, так *и отладка* , и *отмена* завершают связанный процесс.

4. После завершения записи приложения файл трассировки будет закрыт и записан на диск. Это так, если программа завершится сбоем.

5. При открытии файла трассировки отладчик автоматически индексует файл трассировки. Индексирование обеспечивает более точный и быстрый поиск значений памяти. Этот процесс индексирования займет больше времени для больших файлов трассировки.

   dbgcmd

   ```
   ...
   00007ffc`61f789d4 c3              ret
   0:000> !index
   Indexed 1/1 keyframes
   Successfully created the index in 96ms.
   ```

   > ⓘ **Примечание**
   >
   > Ключевой кадр — это расположение трассировки, используемое для индексирования. Ключевые кадры создаются автоматически. Более крупные трассировки будут содержать дополнительные ключевые кадры. При индексировании трассировки отображается количество ключевых кадров.

6. На этом этапе вы находитесь в начале файла трассировки и готовы к перемещению вперед и назад во времени.

> **Совет**
>
> Использование точек останова — это распространенный подход для приостановки выполнения кода на каком-то событии, интересующем вас. Уникальное для TTD, вы можете задать точку останова и вернуться назад до тех пор, пока эта точка останова не будет достигнута после записи трассировки. Возможность проверить состояние процесса после того, как возникла проблема, чтобы определить оптимальное расположение точки останова, позволяет выполнять дополнительные рабочие процессы отладки. Пример использования точки останова в прошлом см. в разделе **"Отладка путешествия по времени" — пример пошагового руководства** по приложению.

## Next Steps

Теперь, когда вы записали трассировку TTD, вы можете воспроизвести трассировку обратно или работать с файлом трассировки, например совместно с ним. Дополнительные сведения см. в этих разделах.

Отладка перемещения по времени — воспроизведение трассировки

Отладка перемещения по времени — работа с файлами трассировки

Отладка временных путешествий — устранение неполадок

Отладка перемещения по времени — пошаговое руководство по приложению

## См. также

Отладка временных путешествий — обзор

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Отладка перемещения по времени — воспроизведение трассировки

Статья • 18.11.2024

В этом разделе описывается воспроизведение трассировок перемещения по времени, навигация вперед и назад во времени.

## Навигация по времени команды

Используйте конечный знак "минус" со следующими командами для перемещения назад во времени.

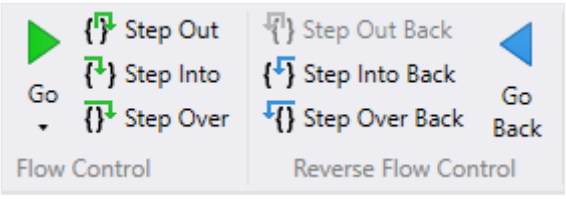
Развернуть таблицу

| Get-Help |
| --- |
| p- (шаг назад) |
| t- (обратная трассировка) |
| g- (вернуться) |

Дополнительные сведения см. в разделе Отладка перемещения по времени — команды навигации.

## Навигация по времени на кнопке ленты

Кроме того, можно использовать кнопки ленты для навигации по трассировке.




## Пример воспроизведения трассировки TTD

Используйте команду g- для выполнения в обратном направлении, пока не будет достигнуто событие или начало трассировки TTD. События, которые могут остановить обратное выполнение, совпадают с тем, что останавливает выполнение вперед. В этом примере достигается начало трассировки.

dbgcmd

```
0:000> g-
TTD: Start of trace reached.
(3f78.4274): Break instruction exception - code 80000003 (first/second
chance not available)
Time Travel Position: 29:0
ntdll!ZwTestAlert+0x14:
00007ffc`61f789d4 c3              ret
```

Используйте команду p (шаг), чтобы выполнить шаг вперед в трассировке TTD.

dbgcmd

```
0:000> p
Time Travel Position: F:1
ntdll!LdrpInitializeProcess+0x1bc5:
7774f828 740b            je      ntdll!LdrpInitializeProcess+0x1bd2
(7774f835) [br=1]
0:000> p
Time Travel Position: F:2
ntdll!LdrpInitializeProcess+0x1bd2:
7774f835 83bdd0fefffff00  cmp     dword ptr [ebp-130h],0
ss:002b:010ff454=00000000
0:000> p
Time Travel Position: F:3
ntdll!LdrpInitializeProcess+0x1bd9:
7774f83c 0f8450e8ffff    je      ntdll!LdrpInitializeProcess+0x42f
(7774e092) [br=1]
```

Для навигации по трассировке также можно использовать команду t (Trace ).

dbgcmd

```
0:000> t
Time Travel Position: F:4
ntdll!LdrpInitializeProcess+0x42f:
7774e092 33c0            xor     eax,eax
0:000> t
Time Travel Position: F:5
ntdll!LdrpInitializeProcess+0x431:
7774e094 e9f5170000      jmp     ntdll!LdrpInitializeProcess+0x1c2b
(7774f88e)
```

Используйте команду p-, чтобы выполнить шаг назад в трассировке TTD.

dbgcmd

```
0:000> p-
Time Travel Position: F:4
ntdll!LdrpInitializeProcess+0x42f:
7774e092 33c0            xor     eax,eax
0:000> p-
Time Travel Position: F:3
ntdll!LdrpInitializeProcess+0x1bd9:
7774f83c 0f8450e8ffff    je      ntdll!LdrpInitializeProcess+0x42f
(7774e092) [br=1]
```

Вы также можете использовать команду t-, чтобы перемещаться назад во времени.

# Команды навигации !tt

Используйте команду !tt для перемещения вперед или назад во времени, пропуская заданную позицию в трассировке.

!tt [позиция]

Укажите позицию времени в любом из следующих форматов для перемещения к этой точке во времени.

- Если [position] является десятичным числом в диапазоне от 0 до 100, оно перемещается примерно до этого процента в трассировку. Например, `!tt 50` перемещается на полпути через трассировку.
- Если {position} имеет значение #:#, где # — шестнадцатеричные числа, он перемещается в эту позицию. Например, `!tt 1A0:12F` перемещается в позицию 1A0:12F в трассировке.

Дополнительные сведения см. в разделе Отладка перемещения по времени — !tt (перемещение по времени).

# !Позиции

Используйте `!positions` для отображения всех активных потоков, включая их положение в трассировке. Дополнительные сведения см. в разделе Отладка перемещения по времени — !позиции (перемещение по времени).

dbgcmd

```
0:000> !positions
>*Thread ID=0x1C74 - Position: F:2
 Thread ID=0x1750 - Position: A5:0
 Thread ID=0x3FFC - Position: 200:0
 Thread ID=0x36B8 - Position: 403:0
 Thread ID=0x3BC4 - Position: 5F2:0
 Thread ID=0x392C - Position: B45:0
 Thread ID=0x32B4 - Position: C87:0
 Thread ID=0x337C - Position: DF1:0
* indicates an actively running thread
```

В этом примере показано, что в текущей позиции находится восемь потоков. Текущий поток — 3604, помеченный символом ">".

> **Совет**
>
> Другой способ отображения текущего списка потоков и их позиций — использовать команду data model dx:
>
> `dx -g @$curprocess.Threads.Select(t => new { IsCurrent = t.Id == @$curthread.Id, ThreadId = t.Id, Position = t.TTD.Position })`

Используйте команду пользовательского режима ~ (состояние потока) и отображает те же восемь потоков и помечает текущий поток следующим образом:

dbgcmd

```
0:000> ~
.  0  Id: 954.1c74 Suspend: 4096 Teb: 00fdb000 Unfrozen
   1  Id: 954.1750 Suspend: 4096 Teb: 00fea000 Unfrozen
   2  Id: 954.3ffc Suspend: 4096 Teb: 00fde000 Unfrozen
   3  Id: 954.36b8 Suspend: 4096 Teb: 00fe1000 Unfrozen
   4  Id: 954.3bc4 Suspend: 4096 Teb: 00fe4000 Unfrozen
   5  Id: 954.392c Suspend: 4096 Teb: 00fed000 Unfrozen
   6  Id: 954.32b4 Suspend: 4096 Teb: 00ff0000 Unfrozen
   7  Id: 954.337c Suspend: 4096 Teb: 00ff3000 Unfrozen
```

В выходных данных команды !position щелкните ссылку рядом с третьим потоком (3FFC), чтобы перейти к этой позиции трассировки, 200:0.

dbgcmd

```
0:002> !tt 200:0
Setting position: 200:0
(954.3ffc): Break instruction exception - code 80000003 (first/second chance
not available)
```

```
Time Travel Position: 200:0
ntdll!NtWaitForWorkViaWorkerFactory+0xc:
7775396c c21400          ret     14h
```

Используйте команду ~ (состояние потока), чтобы убедиться, что мы располагаемся в третьем потоке, 3ffc.

dbgcmd

```
0:002> ~
   0  Id: 954.1c74 Suspend: 4096 Teb: 00fdb000 Unfrozen
   1  Id: 954.1750 Suspend: 4096 Teb: 00fea000 Unfrozen
.  2  Id: 954.3ffc Suspend: 4096 Teb: 00fde000 Unfrozen
   3  Id: 954.36b8 Suspend: 4096 Teb: 00fe1000 Unfrozen
   4  Id: 954.3bc4 Suspend: 4096 Teb: 00fe4000 Unfrozen
   5  Id: 954.392c Suspend: 4096 Teb: 00fed000 Unfrozen
   6  Id: 954.32b4 Suspend: 4096 Teb: 00ff0000 Unfrozen
   7  Id: 954.337c Suspend: 4096 Teb: 00ff3000 Unfrozen
```

**Примечание**

*~s#*, где # — номер потока, также переключается на данный поток, но не изменяет текущую позицию в трассировке. Если *!tt* используется для перемещения по времени в положение другого потока, все значения, которые вы (и отладчик) считываете из памяти, будут искаться в этой позиции. При переключении потоков с *помощью ~s#*отладчик не изменяет текущую позицию внутри системы, которая используется для всех запросов к памяти. Это работает в первую очередь таким образом, чтобы *~s#* не нужно сбрасывать внутренний цикл отладчика.

## Команды расширения отладки перемещения по времени

Сведения о командах `!tt` и `!positions` см. в разделе Отладка `!index` перемещения по времени — команды расширения.

## См. также:

Отладка перемещения по времени — обзор

Отладка перемещения по времени — запись трассировки

Отладка перемещения по времени — работа с файлами трассировки

Отладка по времени — пошаговое руководство по примеру приложения

## Обратная связь

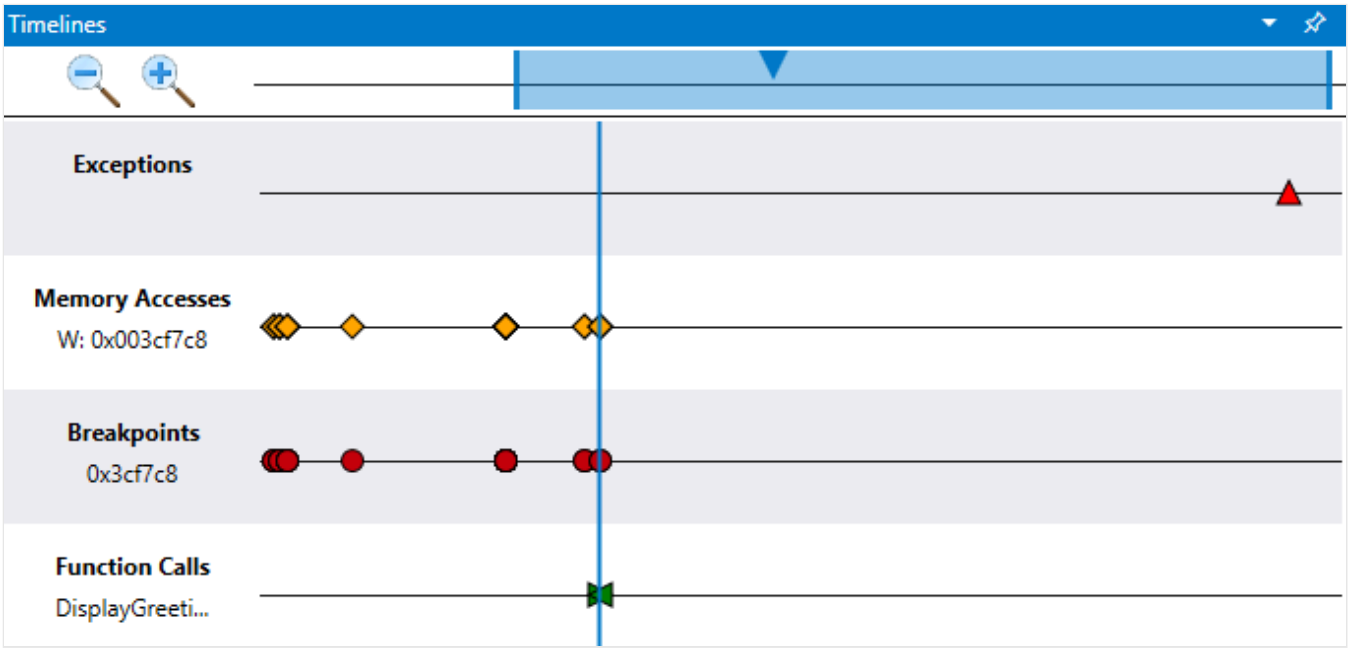
**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# WinDbg — временная шкала

18.07.2025

Отладка с временным путешествием (TTD) позволяет пользователям записывать трассировки, фиксирующие выполнение программы. Временные шкалы — это визуальное представление событий, происходящих во время процесса выполнения. Эти события могут быть точками останова, операциями чтения и записи памяти, вызовами функций и их возвращениями, а также исключениями.




Используйте окно временной шкалы для быстрого просмотра важных событий, понимания относительной позиции и легкого перехода к их расположению в файле трассировки TTD. Используйте несколько временных шкал для визуального изучения событий в трассировке перемещения по времени и обнаружения корреляции событий.

Окно временной шкалы отображается при открытии файла трассировки TTD и отображает ключевые события без необходимости вручную создавать запросы модели данных. В то же время все объекты перемещения по времени доступны, чтобы обеспечить более сложные запросы данных.

Дополнительные сведения о создании и работе с файлами трассировки временных путешествий см. в разделе "Обзор".

## Типы временных шкал

Окно временной шкалы может отображать следующие события:

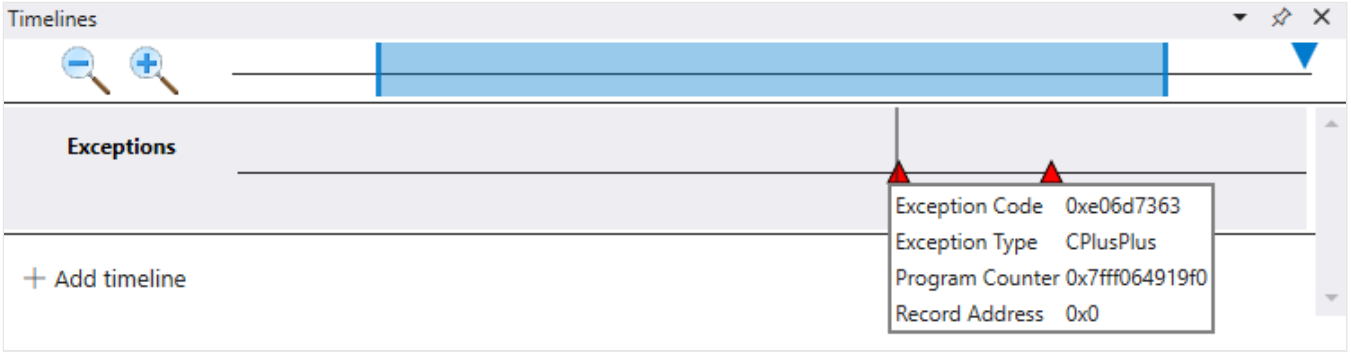
- Исключения (можно дополнительно отфильтровать определенный код исключения)
- Точки останова
- Вызовы функций (поиск в виде модуль!функция)
- Доступ к памяти (чтение и запись и выполнение между двумя адресами памяти)

Наведите указатель мыши на каждое событие, чтобы получить дополнительные сведения с помощью подсказки. Щелкнув событие, запустите запрос к событию и отобразите дополнительные сведения. Дважды щелкнув на событии, вы будете перемещены в это место в файле TTD трассировки.

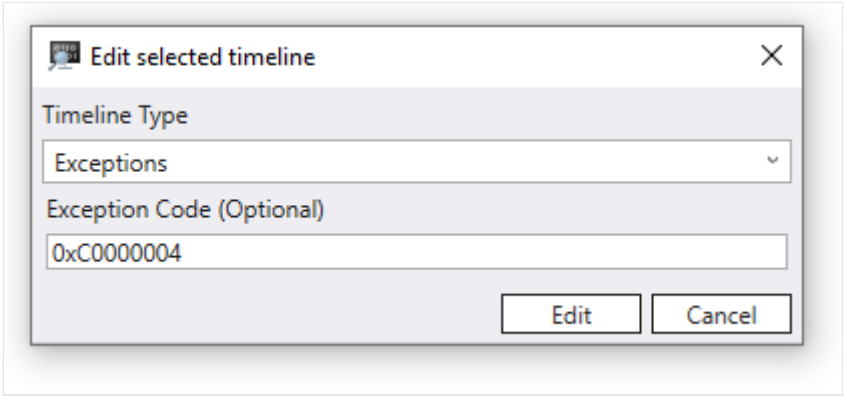
## Исключения

При загрузке файла трассировки и активации временной шкалы, в записи автоматически будут отображаться любые исключения.

Когда вы наводите указатель мыши на точку останова, отображается информация, такая как тип исключения и код исключения.



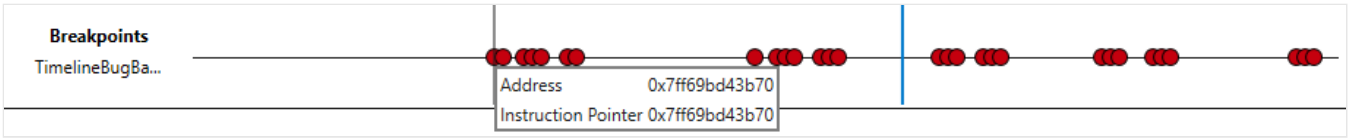

Можно дополнительно отфильтровать определенный код исключения с помощью необязательного поля кода исключения.




Можно также добавить новую временную шкалу для определенного типа исключения.

## Точки останова

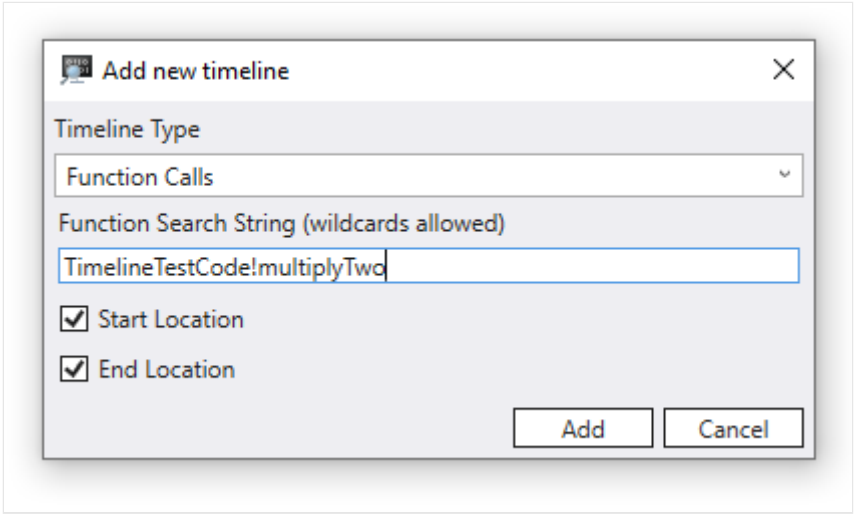
После добавления точки останова можно отобразить позиции, когда эта точка останова попадает на временную шкалу. Это можно сделать, например, с помощью команды bp Set Breakpoint. При наведении указателя мыши на точку останова отображается адрес и указатель инструкции, связанный с точкой останова.



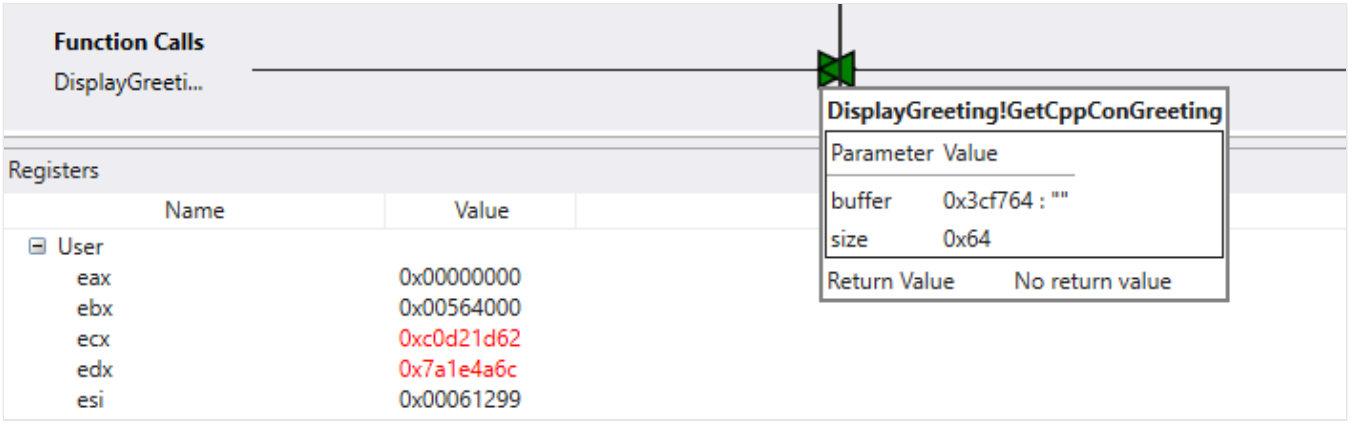

После очистки точки останова связанная временная шкала точек останова автоматически удаляется.

## Вызовы функций

На временной шкале можно отобразить позиции вызовов функций. Для этого выполните поиск в виде `module!function`, например `TimelineTestCode!multiplyTwo`. Можно также указать подстановочные знаки, например `TimelineTestCode!m*`.

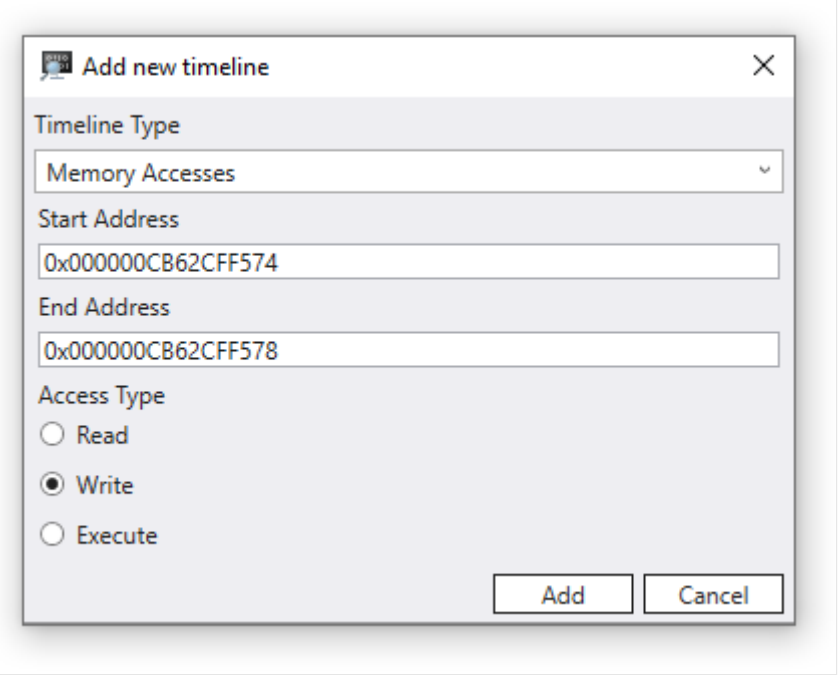



При наведении указателя мыши на функцию вызывается имя функции, входные параметры, их значения и возвращаемое значение отображаются. В этом примере показан *буфер* и *размер* , так как они являются параметрами DisplayGreeting! GetCppConGreeting.

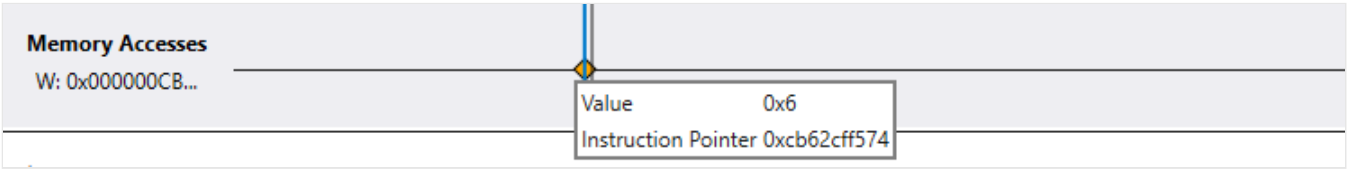


## Обращение к памяти

Используйте временную шкалу доступа к памяти, чтобы отобразить, когда определенный диапазон памяти был считыван или записан в, или где выполняется выполнение кода. Адрес запуска и остановки используется для определения диапазона между двумя адресами памяти.




При наведении указателя мыши на элемент доступа к памяти отображается значение и указатель инструкции.

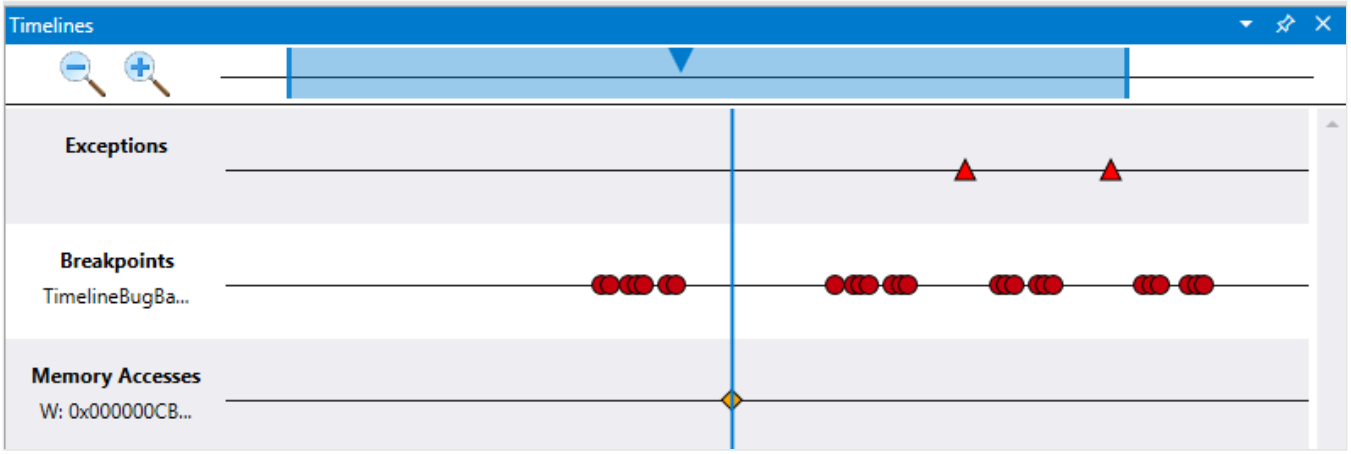



# Работа с временными шкалами

Вертикальная серая линия следует курсору при наведении указателя мыши на временную шкалу. Вертикальная синяя линия указывает текущую позицию трассировки.

Щелкните значки с изображением лупы, чтобы увеличить или уменьшить масштаб на временной шкале.

В верхней области управления временной шкалой используйте прямоугольник для сдвига представления временной шкалы. Перетащите внешние разделители прямоугольника, чтобы изменить размер текущего представления временной шкалы.




## Движения мыши

Увеличьте и уменьшите масштаб с помощью Ctrl + колесика прокрутки.

Панорамирование из стороны в сторону с помощью "Shift" + прокрутки колесика.

# Методы отладки временной шкалы

Чтобы продемонстрировать методы отладки временной шкалы, пошаговое руководство по отладке с использованием Time Travel Debugging повторно используется. В этой демонстрации предполагается, что вы выполнили первые два шага по созданию примера кода и созданию записи TTD с помощью первых двух описанных здесь шагов.

Раздел 1. Создание примера кода

Раздел 2. Запись трассировки примера DisplayGreeting

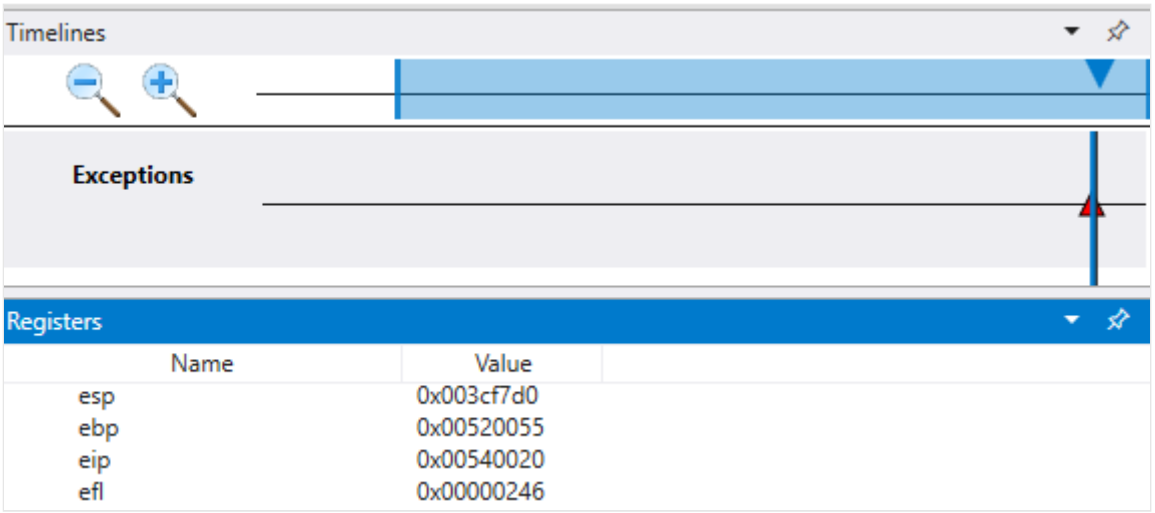
В этом сценарии первым шагом является поиск исключения в трассировке перемещения по времени. Это можно сделать, дважды щелкнув единственное исключение, которое находится на временной шкале.

Когда мы смотрим в командное окно, мы видим, что была выполнена следующая команда, когда мы кликнули на исключение.

dbgcmd

```
(2dcc.6600): Break instruction exception - code 80000003 (first/second chance not
available)
Time Travel Position: CC:0
@$curprocess.TTD.Events.Where(t => t.Type == "Exception")[0x0].Position.SeekTo()
```

Выберите "**Просмотр**>>**регистров**" , чтобы отобразить регистры на этом этапе на временной шкале, чтобы начать исследование.




В выходных данных команды обратите внимание, что стек (esp) и базовый указатель (ebp) указывают на два очень разных адреса. Это может указывать на повреждение стека — возможно, функция вернулась, а затем стек был поврежден. Чтобы проверить это, нам нужно вернуться к моменту до повреждения состояния ЦП и проверить, можем ли мы определить, когда произошло повреждение стека.

По мере этого мы рассмотрим значения локальных переменных и стека.

Выберите **"Просмотреть>>локальные" для отображения локальных** значений.

Выберите **«Вид»>>«Стек»**, чтобы отобразить стек выполнения кода.

В месте сбоя в трассировке обычно проходит несколько шагов после истинной причины в коде обработки ошибок. Используя путешествия во времени, мы можем возвращаться на одну инструкцию назад, чтобы найти истинную первопричину.

**На ленте "Главная"** используйте **команду "Шаг назад"**, чтобы вернуться на три инструкции назад. При этом продолжайте изучать стек, локальные переменные и окна регистров.

В командном окне будут отображаться позиции в перемещении во времени и регистры, когда вы отступите на три инструкции назад.

dbgcmd

```
0:000> t-
Time Travel Position: CB:41
eax=00000000 ebx=00564000 ecx=c0d21d62 edx=7a1e4a6c esi=00061299 edi=00061299
eip=00540020 esp=003cf7d0 ebp=00520055 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
00540020 ??              ???
0:000> t-
Time Travel Position: CB:40
eax=00000000 ebx=00564000 ecx=c0d21d62 edx=7a1e4a6c esi=00061299 edi=00061299
eip=00061767 esp=003cf7cc ebp=00520055 iopl=0         nv up ei pl zr na pe nc
```

```
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
DisplayGreeting!main+0x57:
00061767 c3              ret
0:000> t-
Time Travel Position: CB:3A
eax=0000004c ebx=00564000 ecx=c0d21d62 edx=7a1e4a6c esi=00061299 edi=00061299
eip=0006175f esp=003cf718 ebp=003cf7c8 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206
DisplayGreeting!main+0x4f:
0006175f 33c0            xor     eax,eax
```

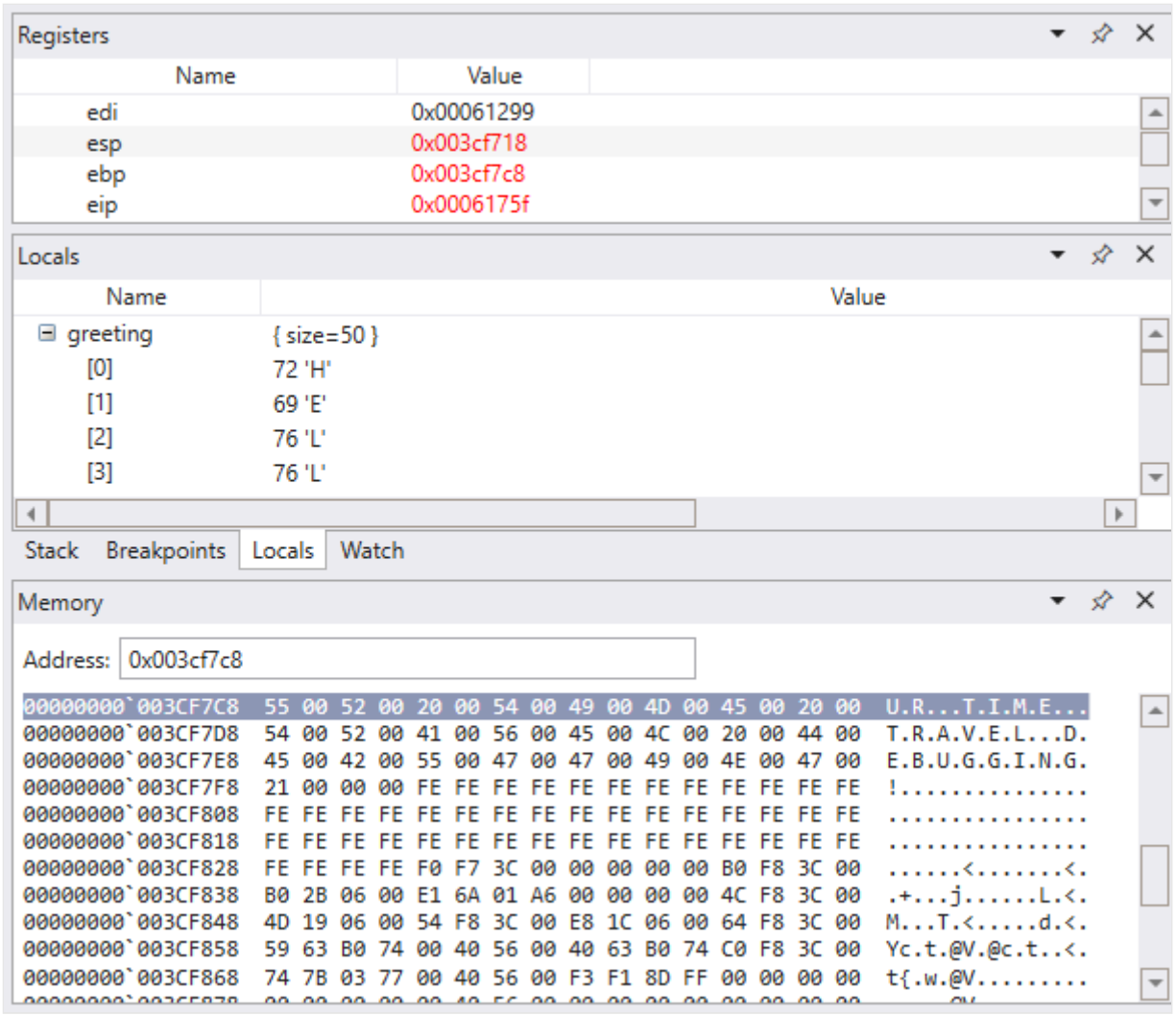
На этом этапе трассировки наш стек и базовый указатель имеют значения, которые имеют больше смысла, поэтому, по-видимому, мы приближаемся к точке в коде, где произошла коррупция.

dbgcmd

```
esp=003cf718 ebp=003cf7c8
```

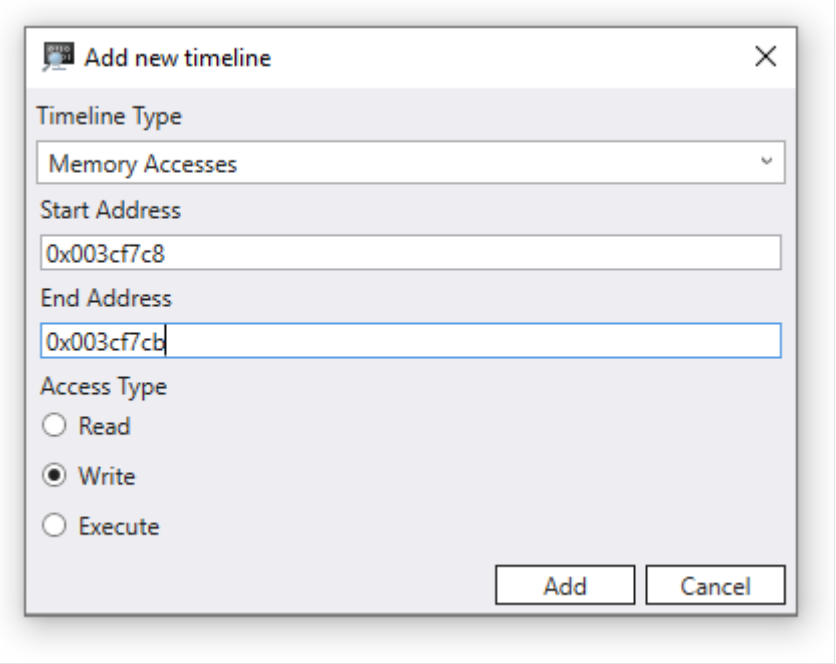
Кроме того, в окне локальных параметров содержатся значения из целевого приложения, а окно исходного кода выделяет строку кода, готовую к выполнению в исходном коде на этом этапе трассировки.

Для дальнейшего изучения можно открыть окно просмотра памяти, чтобы просмотреть содержимое рядом с адресом памяти указателя стека (esp). В этом примере имеется значение 003cf7c8. Выберите **память**>>**текст**>>**ASCII**, чтобы отобразить ASCII-текст, хранящийся по этому адресу.

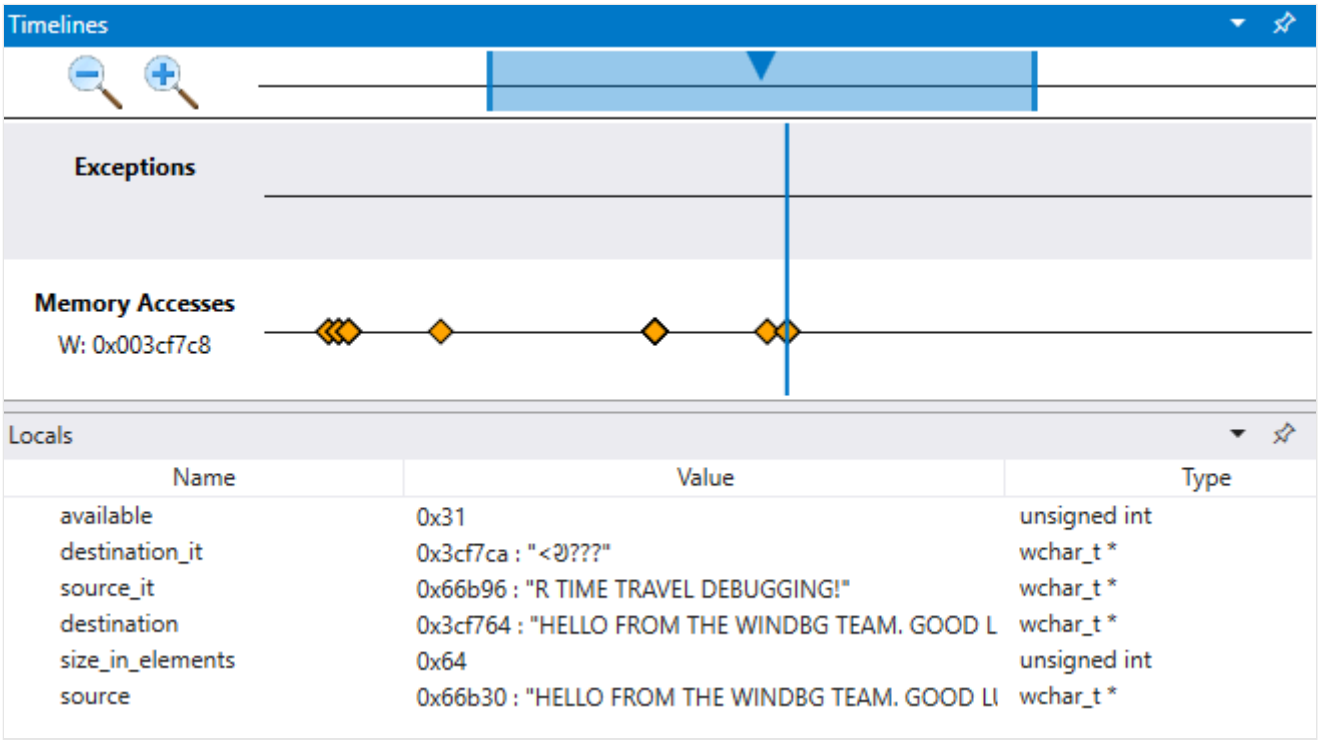



# Временная шкала доступа к памяти

После определения расположения памяти, интересующего вас, добавьте временную шкалу доступа к памяти с помощью этого значения. Нажмите **кнопку +Добавить временную шкалу** и введите начальный адрес. Мы рассмотрим 4 байта, и добавив их к начальному адресу 003cf7c8, в итоге получим 003cf7cb. По умолчанию можно просмотреть все операции записи в памяти, но вы также можете посмотреть только на записи или выполнение кода на этом адресе.




Теперь мы можем пройти по временной шкале в обратном направлении, чтобы изучить, в какой момент это место в памяти было записано в ходе этой трассировки путешествия во времени и увидеть, что мы можем найти. Щелкнув эту позицию на временной шкале, мы видим, что локальные значения отличаются значениями для копируемой строки. Конечное значение, как представляется, не завершено, как если бы длина строки не была правильной.




# Временная шкала точки останова

Использование точек останова — это распространенный подход для приостановки выполнения кода на каком-то событии, интересующем вас. TTD позволяет задать точку

останова и вернуться назад до тех пор, пока эта точка останова не будет достигнута после записи трассировки. Возможность проверить состояние процесса после возникновения проблемы, чтобы определить оптимальное расположение точки останова, позволяет использовать дополнительные рабочие процессы отладки, характерные для TTD.

Чтобы изучить альтернативный метод отладки временной шкалы, щелкните исключение на временной шкале и еще раз выполните три шага назад, используя команду **"Шаг назад** " на ленте **"Главная** ".

В этом очень небольшом примере было бы довольно легко просто просмотреть код, но если есть сотни строк кода и десятки подпрограмм, можно использовать описанные здесь техники для уменьшения времени, необходимого для выявления проблемы.

Как упоминалось ранее, базовый указатель (esp) вместо указания на инструкцию указывает на текст сообщения.

Используйте команду ba, чтобы задать точку останова для доступа к памяти. Мы установим точку останова на запись, чтобы узнать, когда происходит запись в эту область памяти.

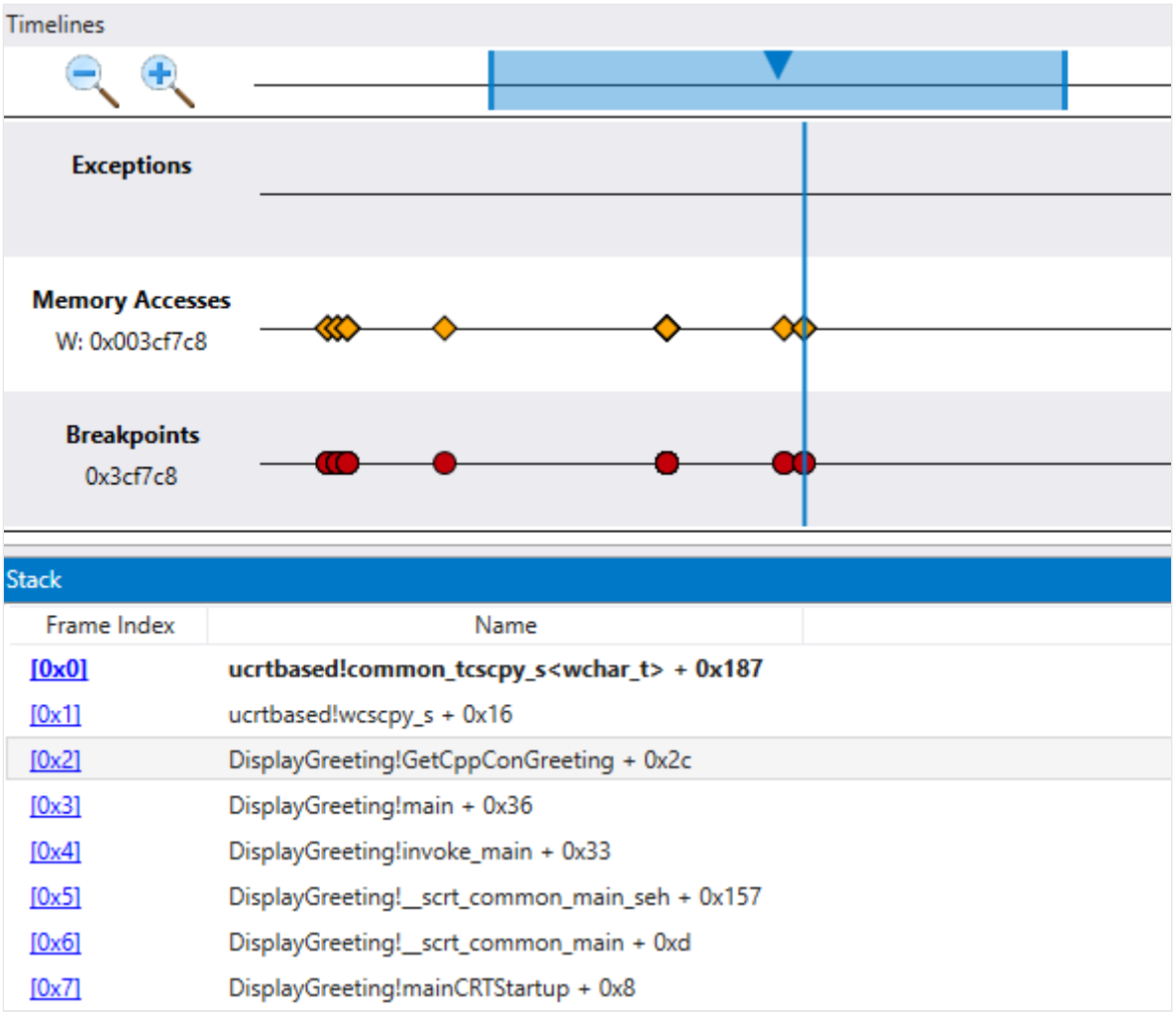
dbgcmd

```
0:000> ba w4 003cf7c8
```

Хотя мы будем использовать простую точку останова по доступу к памяти, точки останова можно создать в виде более сложных условных операторов. Дополнительные сведения см. в разделе bp, bu, bm (Set Breakpoint).

В меню "Главная" выберите **"Вернуться** назад", пока точка останова не будет достигнута.

На этом этапе мы можем проверить стек программ, чтобы узнать, какой код активен.




Так как маловероятно, что у функции wscpy_s(), предоставленной корпорацией Майкрософт, будет подобная ошибка кода, мы исследуем стек дальше. В стеке показано, что приветствие!main вызывает Приветствие! GetCppConGreeting. В нашем очень маленьком примере кода мы могли просто открыть код на этом этапе и, скорее всего, найти ошибку довольно легко. Но чтобы проиллюстрировать методы, которые можно использовать с более крупной, более сложной программой, мы добавим временную шкалу вызова функции.

# Временная шкала вызова функции

Нажмите **кнопку +Добавить временную шкалу** и введите `DisplayGreeting!GetCppConGreeting` строку поиска функции.

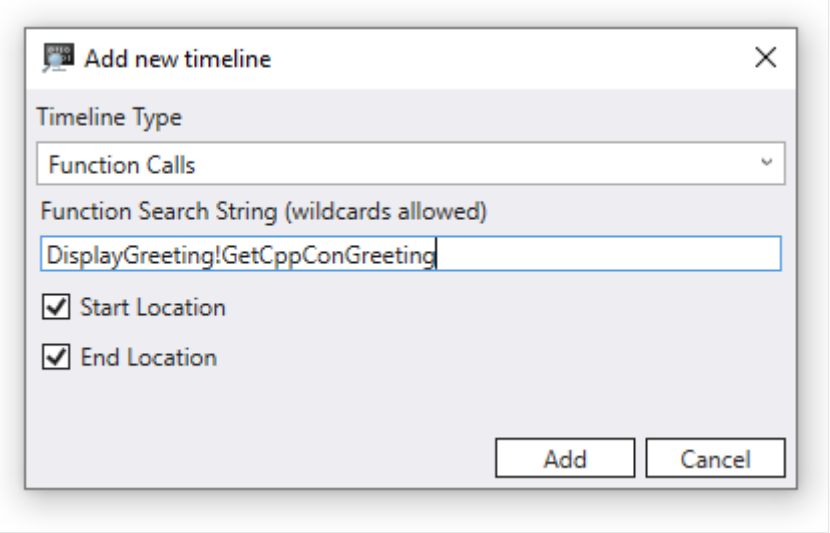
Флажки для указания начального и конечного местоположений указывают на начало и конец вызова функции в трассировке.

Мы можем использовать команду dx для отображения объекта вызова функции, чтобы просмотреть связанные поля TimeStart и TimeEnd, соответствующие начальному расположению и конечному расположению вызова функции.

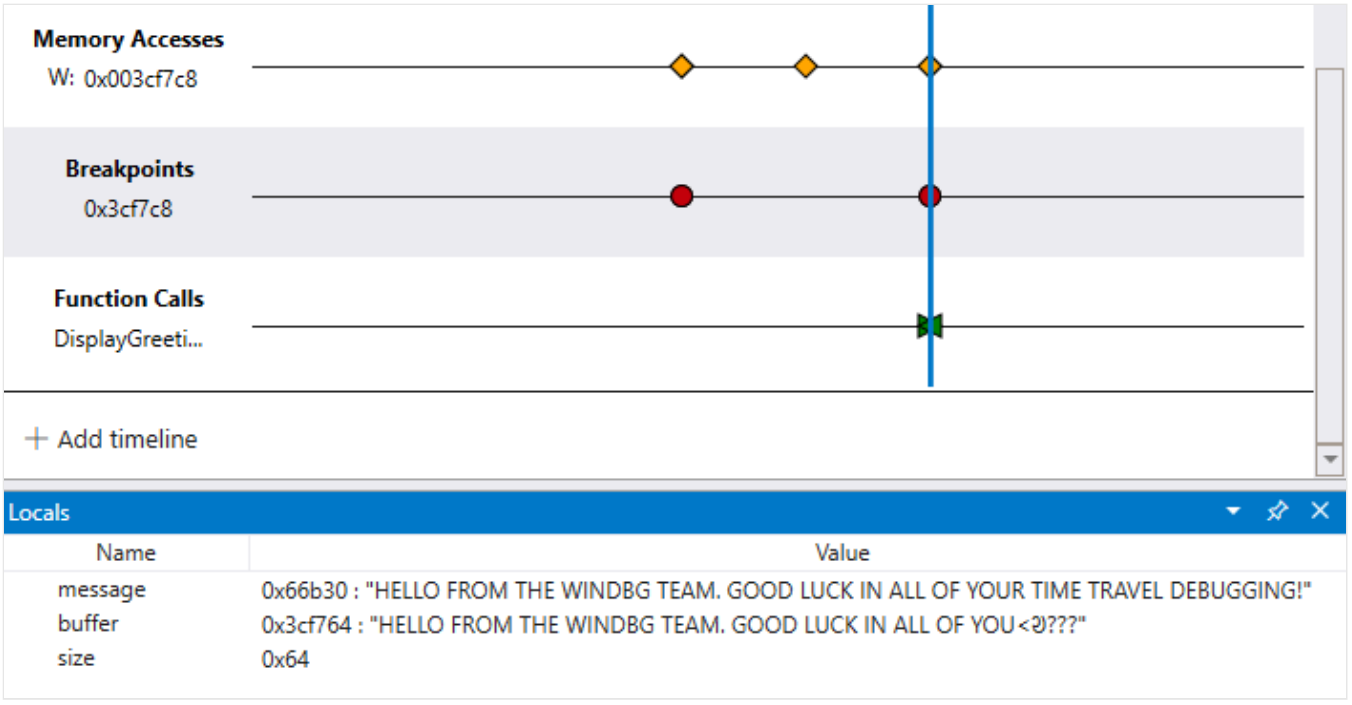
dbgcmd

```
dx @$cursession.TTD.Calls("DisplayGreeting!GetCppConGreeting")[0x0]
    EventType        : 0x0
    ThreadId         : 0x6600
    UniqueThreadId   : 0x2
    TimeStart        : 6D:BD [Time Travel]
    SystemTimeStart  : Thursday, October 31, 2019 23:36:05
    TimeEnd          : 6D:742 [Time Travel]
    SystemTimeEnd    : Thursday, October 31, 2019 23:36:05
    Function         : DisplayGreeting!GetCppConGreeting
    FunctionAddress  : 0x615a0
    ReturnAddress    : 0x61746
    Parameters
```

Необходимо установить флажок либо "Начало", либо "Конец", либо одновременно "Начало" и "Конец".

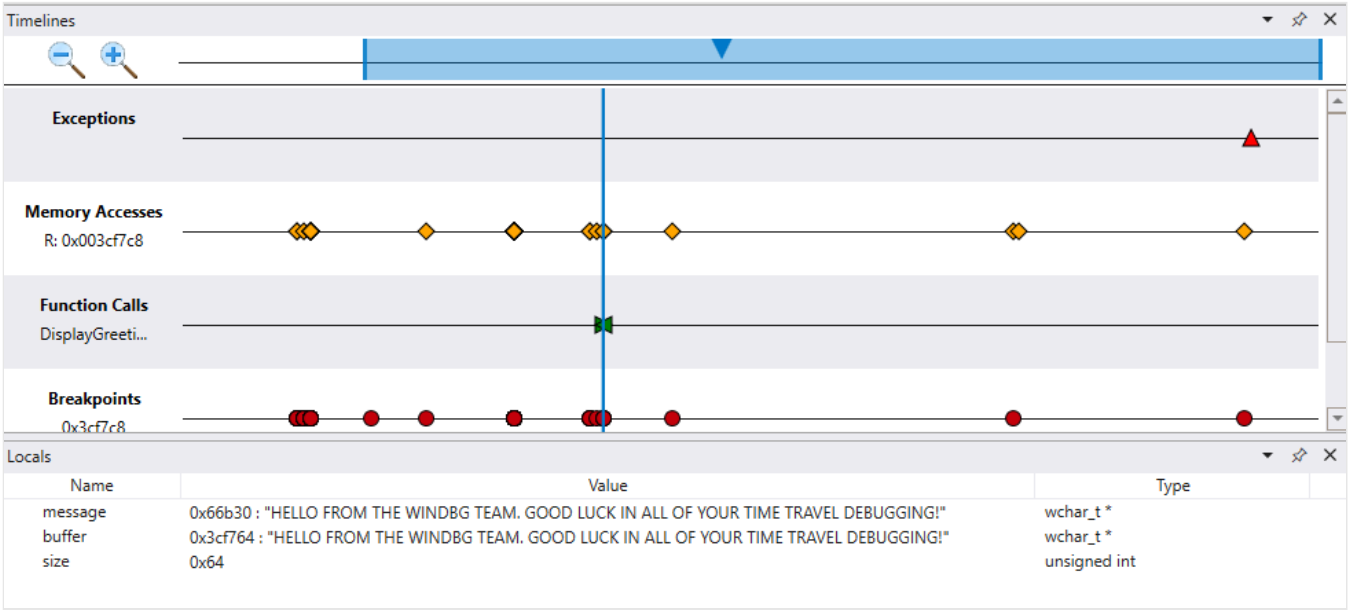



Так как наш код не является рекурсивным или реентерабельным, довольно легко определить на временной шкале момент вызова метода GetCppConGreeting. Вызов GetCppConGreeting также происходит в то же время, что и наша точка останова, а также событие доступа к памяти, которое мы определили. Так что похоже, что мы сузились в области кода, чтобы внимательно изучить первопричину сбоя приложения.




# Изучение выполнения кода путем просмотра нескольких временных шкал

Хотя наш пример кода мал, метод использования нескольких временных шкал позволяет визуально исследовать трассировку перемещения по времени. Вы можете просмотреть файл трассировки, чтобы задать вопросы, например, "когда происходит доступ к области памяти до того, как срабатывает точка останова?".

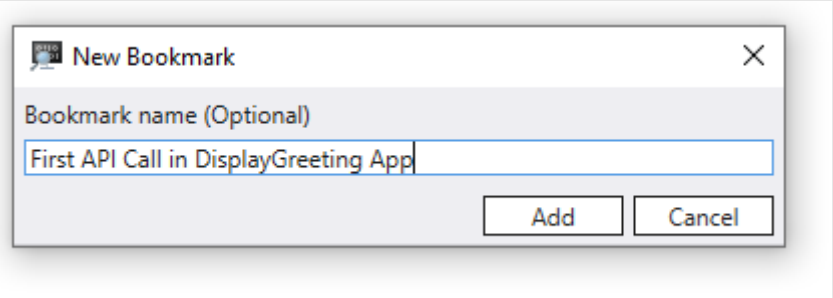



Возможность видеть дополнительные корреляции и находить неожиданные открытия может выделять средство временной шкалы от взаимодействия с трассировкой временных путешествий с помощью команд командной строки.

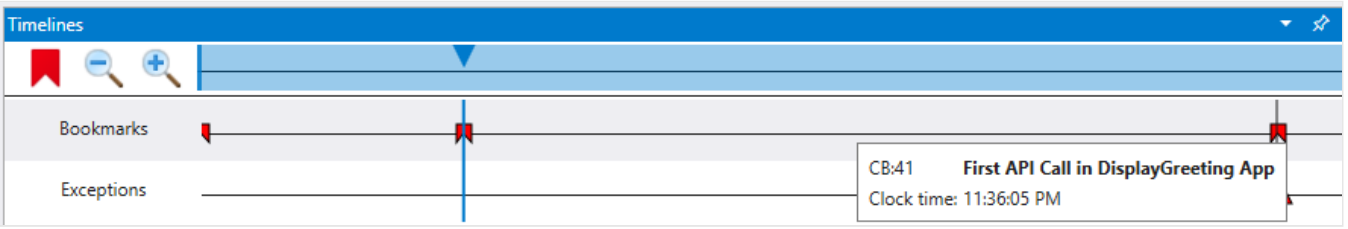
# Закладки временной шкалы

Закладка важных позиций поездок по времени в WinDbg вместо ручного копирования позиции в блокнот. Закладки упрощают просмотр разных позиций трассировки относительно других событий и их анимацию.

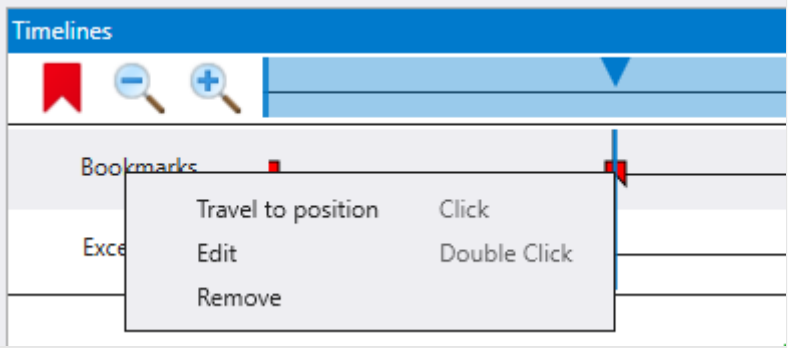
Вы можете указать описательное имя для закладок.




Доступ к закладкам через окно временной шкалы, доступное в *режиме просмотра* > *временной шкалы*. При наведении указателя мыши на закладку будет отображаться имя закладки.




Щелкните правой кнопкой мыши закладку, чтобы перейти к этой позиции, переименовать или удалить закладку.




> ⓘ **Примечание**
>
> В версии 1.2402.24001.0 отладчика функция закладки недоступна.

## См. также

Функции WinDbg

Пошаговое руководство по отладке временных путешествий

# Отладка перемещения по времени — работа с файлами трассировки

Статья • 01.04.2025



В этом разделе описывается, как работать с файлами, созданными и потребляемыми отладкой по времени.

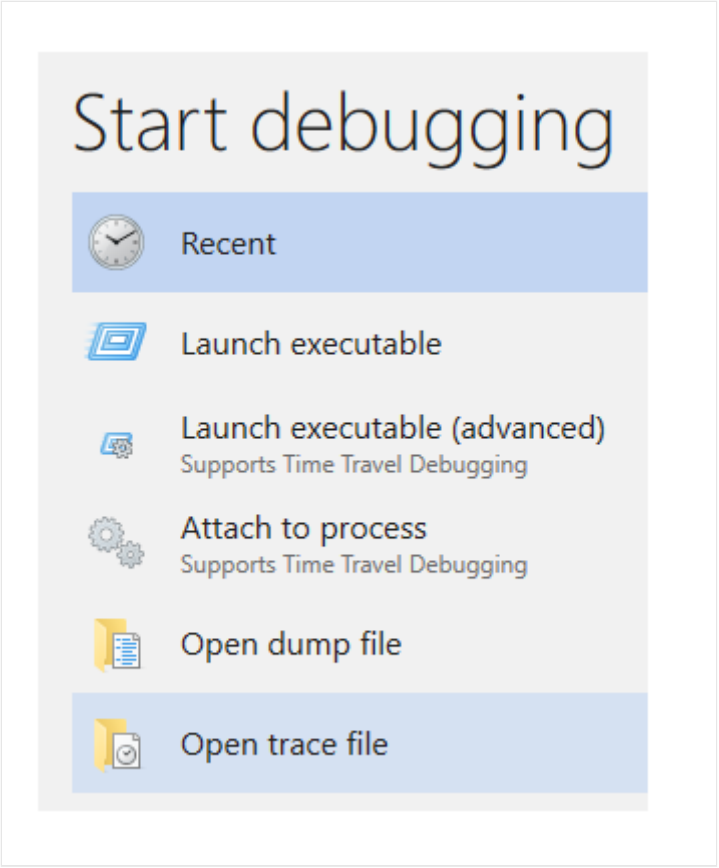
## Обзор файла трассировки

Отладка с использованием технологии путешествия во времени использует следующие файлы для анализа выполнения кода.

- Файл трассировки содержит запись выполнения кода и имеет расширение .RUN.
- Файл индекса обеспечивает быстрый доступ к информации в файле трассировки и имеет расширение .IDX.
- Ошибки записи и другие выходные данные записи записываются в файл журнала отладчика.

## Отследить .RUN файлы

Файлы трассировки .RUN можно открыть после записи, используя **Файл**>**Запустить отладку**>**Открыть файл трассировки**.



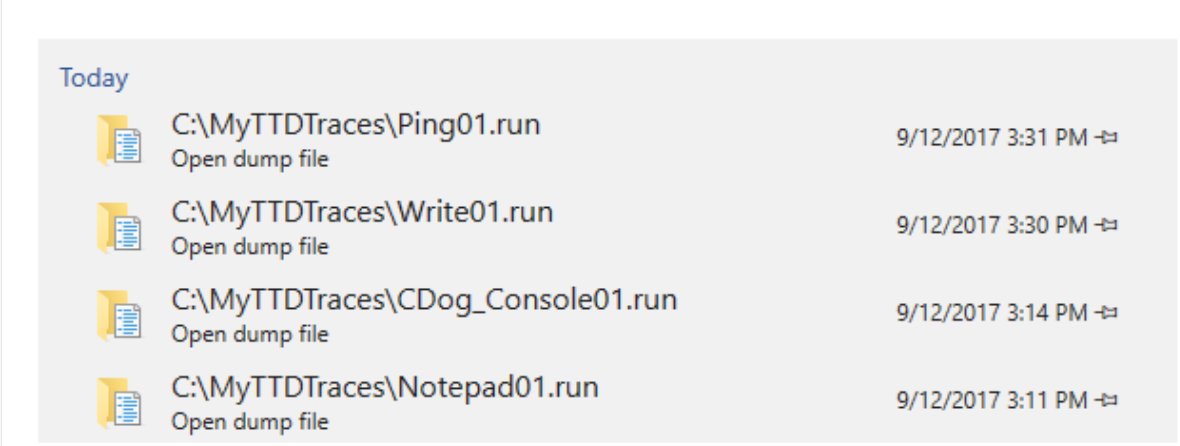

Все выходные файлы трассировки хранятся в папке документов пользователей по умолчанию. Например, для User1 файлы TTD будут храниться здесь:

Консоль

```
C:\Users\User1\Documents
```

При запуске записи можно изменить расположение файлов трассировки. Дополнительные сведения см. в Отладка перемещения по времени — запись.

Последний использованный список файлов позволяет быстро получить доступ к ранее используемым целевым файлам конфигурации. Также перечислены все недавно используемые файлы трассировки или файлы дампа.



# Файлы индекса .IDX

Индексный файл .IDX создается для ассоциированного трассировочного файла .RUN автоматически при открытии этого файла в WinDbg. Файл индекса можно создать вручную с помощью команды !index. Индекс обеспечивает более быстрый доступ к сведениям трассировки.

Файлы IDX также могут быть большими, как правило, в два раза больше .RUN-файла.

# Создание файла .IDX заново

Вы можете повторно создать файл .IDX из .RUN с помощью команды `!index`. Дополнительные сведения см. в статье "Отладка по пути времени" — !index (перемещение по времени).

dbgcmd

```
0:0:001> !index
Indexed 3/3 keyframes
Successfully created the index in 49ms.
```

# Общий доступ к файлам трассировки TTD .RUN

TTD работает только локально и не работает при удалённом подключении к другому компьютеру.

Файлы трассировки TTD можно предоставлять другим пользователям, копируя .RUN файл. Это может быть удобно для того, чтобы коллега помог вам выяснить проблему. Они не должны устанавливать аварийное приложение или выполнять любую другую связанную настройку, чтобы попытаться воспроизвести проблему. Они могут просто загрузить файл трассировки и отладить приложение, как если бы оно было установлено на своем компьютере.

Компьютер, на котором выполняется воспроизведение трассировки TTD, должен поддерживать все инструкции, используемые на компьютере записи, например инструкции AVX.

Вы можете переименовать файл, чтобы включить дополнительные сведения, такие как дата или номер ошибки.

Файл .IDX не требуется копировать, так как его можно воссоздать с помощью команды !index, как указано выше.

> **Совет**
>
> При совместной работе с другими пользователями передавайте любые релевантные данные трассировки, связанные с проблемой. Сотрудник может использовать `!tt x:y` команду, чтобы перейти к этой точной точке во времени выполнения кода. Диапазоны позиций времени можно включить в описания ошибок, чтобы отслеживать, где может возникнуть проблема.

# Ошибка — файл журнала

Ошибки записи и другие выходные данные записи записываются в файл журнала отладчика. Чтобы просмотреть файл журнала, выберите "**Просмотр**>**журналов**".

В этом примере показан текст журнала ошибок при попытке запустить и записать исполняемый файл с именем Foo.exe, который не находится в каталоге C:\Windows.

Консоль

```
2017-09-21:17:18:10:320 : Information : DbgXUI.dll : TTD: Output:
Microsoft (R) TTD 1.01.02
Release: 10.0.16366.1000
Copyright (C) Microsoft Corporation. All rights reserved.
Launching C:\Windows\Foo.exe
2017-09-21:17:18:10:320 : Error : DbgXUI.dll : TTD: Errors:
Error: Trace of C:\Windows\Foo.exe PID:0 did not complete successfully:
status:27
Error: Could not open 'Foo.exe'; file not found.
Error: Corrupted trace dumped to C:\Users\User1\Documents\Foo01.run.err.
```

# Размер файла трассировки

Файлы трассировки TTD могут быть очень большими, и важно убедиться, что у вас есть достаточно свободное место на диске. Если вы записываете приложение или процесс даже в течение нескольких минут, файл трассировки может увеличиться до нескольких гигабайтов. Размер файла трассировки зависит от ряда факторов, описанных ниже.

TTD не задает максимальный размер файлов трассировки, чтобы учесть сложные длительно работающие сценарии. Быстрая повторная симуляция проблемы

позволит сохранить файл трассировки минимального размера.

## Факторы размера файла трассировки

Невозможно предоставить и точную оценку размера файла трассировки, но существует несколько правил, которые помогут вам понять размеры файлов TTD.

Следующие факторы могут повлиять на размер файла трассировки:

- Количество инструкций по коду, выполняемых во всех потоках при записи запущенного приложения или процесса
- Продолжительность записи приложения или процесса (только так как это влияет на количество записанных инструкций кода)
- Размер данных памяти, используемых приложением или процессом

Количество выполняемых и записанных инструкций — это самый большой фактор, влияющий на размер файла трассировки. Трассировка обычно требует от 1 бита до 1 байта на каждую выполненную инструкцию. Вероятно, трассировка окажется в нижней части этого диапазона, когда запротоколированная программа выполняет меньше различных функций и обрабатывает меньший объем данных. Скорее всего, трассировка окажется на верхнем пределе этого диапазона, когда записанная программа выполняет больше различных функций или работает с более обширным набором данных.

## Правило размера файла трассировки

Файл трассировки увеличивается примерно на 5 МБ до 50 МБ в секунду при записи активного приложения или процесса в зависимости от факторов размера файла трассировки, определенных выше.

Файл трассировки не будет расти, когда приложение или процесс записи неактивен (например, при ожидании входных данных).

В настоящее время для файлов трассировки нет максимального размера файла. WinDbg может воспроизводить файлы трассировки, размер которых может составлять сотни гигабайтов.

## Размер файла индекса

Файл индекса автоматически создается WinDbg при первом открытии трассировки. Он содержит сведения, которые помогают отладчику более эффективно воспроизводить трассу и запрашивать информацию о памяти. Его размер обычно

составляет от 1 до 2 раз больше размера файла трассировки. Факторы, влияющие на его размер, аналогичны тем, которые влияют на размер файла трассировки.

Во-первых, размер файла индекса масштабируется относительно длины трассировки. Трассировка, содержащая большее количество записанных инструкций, обычно будет иметь более высокий индекс.

Во-вторых, размер индекса масштабируется относительно ширины доступа к памяти. Если записанная программа обращалась к большому количеству различных расположений памяти, индекс, как правило, будет больше, чем если программа обращалась к меньшему количеству различных расположений памяти или если обращения к ним были менее частыми.

Так как эти факторы похожи на факторы, влияющие на размер файла трассировки, размер файла индекса обычно масштабируется относительно размера файла индекса (поэтому мы оцениваем, что обычно это от 1x до 2x размер файла трассировки).

## Что делать, если у меня не будет места на диске?

Файлы трассировки и индекса TTD записываются на диск. В настоящее время для файла трассировки или индекса не существует ограничения максимального размера файла. Файл трассировки увеличивается до прекращения записи или превышения объема доступного места на диске.

*Во время записи:* TTD будет записывать последнюю страницу в файл трассировки и затем ждать, пока не сможет записывать снова. WinDbg продолжает отображать диалоговое окно записи, но не отображает сообщение об ошибке и предупреждении при истечении места на диске во время записи.

Недостаточно места на диске во время записи приводит к созданию файла трассировки с неполной записью выполнения кода. Неполный файл трассировки можно открыть в WinDbg, но он может не содержать фактическую проблему, если ошибка происходит после того, как на диске заканчивается место во время записи файла трассировки.

Обходное решение: Откройте Проводник и проверьте, равно ли свободное место на диске почти нулю. Кроме того, просмотрите файл трассировки (.RUN) в Проводник (по умолчанию в папке «Документы») и, если размер файла не увеличивается регулярно, возможно, запись находится на ожидании. Нажмите кнопку "Остановить и отладить" в WinDbg, освободите место или сохраните на другой диск, и снова начните запись.

*Во время индексирования:* отладчик может создать недопустимый файл индекса, что приводит к непредсказуемому поведению в отладчике, или движок отладчика может завершиться сбоем.

Способ обхода: Закройте отладчик и удалите любой файл индекса (.idx), если он существует для вашей трассировки. Освободить достаточно места на диске или переместить файл трассировки на другой диск с достаточным объемом свободного места. Снова откройте трассировку в отладчике и запустите индекс !index, чтобы создать новый, правильный индекс. Индексирование не изменяет исходный файл трассировки (.run), поэтому данные не будут потеряны.

## См. также

Отладка временных путешествий — обзор

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Отладка временных путешествий — устранение неполадок

Статья • 10.04.2025



В этом разделе описывается, как отлаживать трассировки путешествий во времени.

## Проблемы с попытками записи процесса

### Я получаю сообщение об ошибке: "WinDbg необходимо запустить с повышенными привилегиями для поддержки отладки с путешествием во времени."

Как указано в сообщении, запуск отладчика с повышенными привилегиями является обязательным. Чтобы запустить отладчик с повышенными привилегиями, щелкните правой кнопкой мыши **значок WinDbg** в меню "Пуск" и выберите "**Дополнительно**>**запустить от имени администратора**".

### Я установил WinDbg с учетной записью, которая не имеет прав администратора, и я получаю сообщение об ошибке, которое говорит: "WinDbg необходимо запустить с повышенными привилегиями для поддержки Time Travel Debugging".

Повторно установите WinDbg с помощью учетной записи, которая имеет права администратора и использует эту учетную запись при записи в отладчике.

### Не удается запустить и записать приложение UWP

Это не поддерживается в настоящее время, но вы можете подключиться и записать уже работающее приложение UWP.

### Не удается записать необычный тип процесса — выполнение в другом сеансе, контексте безопасности,

### учетных данных...

В настоящее время TTD записывает только обычные процессы, которые можно запустить из командной консоли или щелкнув исполняемый файл или ярлык в проводнике Windows.

## Не удается успешно записать приложение на компьютере

Если не удается записать ваше приложение, убедитесь, что вы можете записать простой процесс Windows. Например, "ping.exe" или "cmd.exe" — это простые процессы, которые обычно могут быть записаны.

## Я не могу успешно записать все на моем компьютере

Запись TTD — это инвазивная технология, которая может повлиять на другие инвазивные технологии, такие как платформы виртуализации приложений, продукты управления информацией, программное обеспечение безопасности или антивирусные продукты.

Дополнительную информацию об известных несовместимостях TTD можно найти в разделе "На что обратить внимание" в Обзор отладки путешествий во времени.

## Я отслеживаю приложение и запускаю AppVerifier одновременно, а производительность при повторной отправке трассировки замедляется.

Из-за того, как AppVerifier использует память для проверки приложения, опыт позже при повторном воспроизведении трассировки может быть заметно хуже, чем без AppVerifier. Чтобы повысить производительность, отключите AppVerifier при записи приложения. Если это невозможно, может потребоваться закрыть окно вызова в WinDbg, чтобы повысить производительность.

# Проблемы с индексными файлами .IDX

Отладка файла трассировки без файла индекса или с поврежденным или неполным файлом индекса возможна, но не рекомендуется. Файл индекса необходим, чтобы убедиться, что значения памяти, считываемые из отлаживаемого процесса, наиболее точны и для повышения эффективности всех других операций отладки.

Используйте команду `!index -status` для проверки состояния файла индекса .IDX, связанного с файлом трассировки .RUN.

Если вы можете попытаться восстановить файл индекса, выполнив команду `!index -force`.

## Создание индексного файла .IDX

Если вы подозреваете проблему с файлом индекса, или `!index -status` сообщает что-либо, кроме 'Файл индекса загружен', воссоздайте его. Для этого можно запустить `!index -force`. Если это не сработает:

1. Закройте отладчик.
2. Удалите существующий файл IDX, он будет иметь то же имя, что и файл трассировки .RUN и находиться в том же каталоге, что и файл .RUN.
3. Откройте трассировку .RUN-файл в WinDbg. Эта команда `!index` запустит процесс повторного создания индекса.
4. Используйте команду, чтобы убедиться, `!index -status` что индекс трассировки работает.

Убедитесь, что для файла индекса достаточно места в том же расположении, где находится файл трассировки. В зависимости от содержимого записи индексный файл может быть значительно больше, чем файл трассировки, как правило, в порядке в два раза больше.

# Проблемы с файлами трассировки .RUN

При возникновении проблем с трассировкой .RUN-файла вы можете получать сообщения об ошибках, например, такие.

dbgcmd

```
Replay and log are out of sync at fallback data. Packet type is incorrect "Packet
Type"
Replay and log are out of sync at opaque data. Log had already reached the end
Replay exit thread event does not match up with logged event
Logged debug write values are out of sync with replay
```

В большинстве случаев все сообщения об ошибках указывают на то, что . Файл трассировки RUN недоступен и должен быть повторно записан.

## Повторная запись приложения пользовательского режима

Если возникла определенная проблема с записью приложения в режиме пользователя, может потребоваться попробовать записать другое приложение на одном компьютере

или попробовать одно и то же приложение на другом компьютере. Вы можете попробовать и записать другое использование приложения, чтобы узнать, существует ли определенная проблема с записью определенных частей приложения.

## При отладке или создании индекса, я вижу сообщения о "событиях схода с рельс".

Возможно, вы увидите такие сообщения:

```
dbgcmd
```

```
Derailment event MissingDataDerailment(7) on UTID 2, position 2A550B:108 with PC
0x7FFE5EEB4448 Request address: 0x600020, size: 32
```

TTD работает путем запуска эмулятора внутри отладчика, который выполняет инструкции отлаживаемого процесса, чтобы реплицировать состояние этого процесса на каждой позиции записи. Сбои происходят, когда этот эмулятор наблюдает некоторое несоответствие между результирующей состоянием и информацией, найденной в файле трассировки. Ошибка, указанная выше, например, ссылается на инструкцию, найденную по адресу 0x7FFE5EEB4448 на позиции 2A550B:108 в трассировке, которая пыталась считать данные из памяти вблизи адреса 0x600020, который отсутствует в записи.

Сходы с рельсов часто вызваны ошибками в рекордере или иногда в эмуляторе, на некоторых записанных инструкциях, расположенных ранее в трассировке.

В большинстве случаев это сообщение об ошибке указывает на то, что файл трассировки .RUN будет иметь разрыв в потоке, который отклонился от курса, начиная с той точки, где это произошло, для неопределённого количества инструкций. Если событие, которое вы пытались исследовать, не произошло в течение этого пробела, трассировка может быть пригодна. Если во время этого разрыва произошло интересующее событие, трассировку нужно перезаписать.

# См. также

Отладка временных путешествий — обзор

Заметки о выпуске отладки путешествий во времени

# Команды навигации по времени

Статья • 22.11.2024



В этом разделе описываются команды навигации по времени.

## p- (шаг назад)

Команда *p-выполняет* предыдущую одну инструкцию или исходную строку. При возникновении вызовов или прерываний вложенных вызовов они рассматриваются как один шаг. Эту команду можно вызвать с помощью кнопки **"Шаг назад"** на **ленте "Главная** " в WinDbg.

## t- (трассировка назад)

Команда *t-* выполняет предыдущую одну инструкцию или исходную строку. При возникновении вызовов или прерываний подзадачей выполняется трассировка каждого шага. Эту команду можно вызвать с помощью кнопки **"Шаг назад** " на **ленте "Главная** " в WinDbg.

## g- (вернуться)

Команда *g-* начинает выполнение текущего процесса в обратном направлении. Выполнение будет остановлено в конце программы, при нажатии BreakAddress или при возникновении другого события, вызывающего остановку отладчика. Эту команду можно вызвать с помощью кнопки **"Назад"** на **ленте "Главная** " в WinDbg.

## Дополнительная информация

Команды навигации по пути во времени работают только с трассировками перемещения по времени. Дополнительные сведения о путешествии по времени см. в разделе "Отладка временных путешествий" — обзор.

## См. также

Отладка временных путешествий — обзор

Отладка перемещения по времени — воспроизведение трассировки

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Команды расширения для отладки по времени

18.07.2025



В этом разделе приводятся команды расширения отладчика по времени.

## В этом разделе

Развернуть таблицу

| Topic | Description |
|---|---|
| !tt (путешествие по времени) | Расширение отладчика !tt (путешествие по времени), позволяющее перемещаться вперед и назад во времени. |
| !positions | The !positions extension displays all the active threads, including their current positions in the time travel trace. |
| !index | The !index extension indexes time travel traces or displays index status information. |

## Additional Information

Эти расширения работают только с трассировками перемещения по времени. Дополнительные сведения о путешествии по времени см. в разделе "Отладка временных путешествий" — обзор.

## DLL

Команды расширения отладчика по времени реализуются в ttdext.dll. Библиотека dll команды перемещения по времени загружается автоматически в WinDbg. Для загрузки ttdext.dll вручную не требуется использовать команду загрузки.

## See Also

Отладка временных путешествий — обзор

# !tt (путешествие по времени)

Статья • 10.04.2025



Расширение отладчика !tt (путешествие по времени), позволяющее перемещаться вперед и назад во времени.

## Навигационные команды !tt

Используйте расширение !tt для перехода вперед или обратно во времени, перейдя к заданной позиции в трассировке.

dbgcmd

```
!tt [position]
```

### Параметры

**позиция**

Укажите позицию времени в любом из следующих форматов для перемещения в этот момент времени.

- Если {position} является десятичным числом от 0 до 100, он перемещается примерно на этот процент в трассировку. Рассмотрим пример.
  - !tt 0 — перемещение во времени к началу трассировки
  - !tt 50 — путешествие во времени на середину трассировки
  - !tt 100 — путешествие во времени до конца трассировки

- Если {position} является числом с плавающей запятой от 0 до 100, оно перемещается примерно на этот процент вдоль трассы. Рассмотрим пример.
  - !tt 0.00 — время перемещения к началу трассировки
  - !tt 50.1 — путешествие во времени чуть более чем наполовину по временной шкале
  - !tt 99.9 — путешествие во времени почти до конца следа

- Если {position} имеет значение #:#, где # является шестнадцатеричными числами, он перемещается в эту позицию. Если число после : опущено, значение по умолчанию равно нулю.

- !tt 1A0: — время поездки на позицию 1A0:0
- !tt 1A0:0 — путешествие во времени к позиции 1A0:0
- !tt 1A0:12F - Время путешествия по позиции 1A0:12F

> ⓘ **Примечание**
>
> Трассировки используют двухчастную позицию инструкции, которая ссылается на конкретную позицию в трассировке, например 12:0. или 15:7. Два элемента — шестнадцатеричные числа, определенные как описано здесь.
>
> xx:yy
>
> xx - первый элемент — это номер последовательности, соответствующий событию последовательности.
>
> y — второй элемент — это число шагов, которое соответствует примерно числу инструкций с момента события последовательности.

# Команды останова !tt

## Прерывание регистрации

`!tt br[-] <register> [<value>]`

Эта команда перейдет к предыдущей или следующей позиции, где указанный регистр изменил значения (или, если задано `<value>`, принимает указанное значение) в текущем потоке.

Примеры:

`!tt br rbx 0x12345678` — найдите следующую позицию в текущем потоке, который устанавливает rbx в 0x12345678.

`!tt br- ebx` — найти предыдущую позицию в текущем потоке, установившую rbx в его текущее значение.

dbgcmd

```
0:000> !tt br- ebx

Setting position: 2C8:0
(3b24.2d98): Break instruction exception - code 80000003 (first/second chance not
available)
```

```
Time Travel Position: 2C8:0
ntdll!LdrInitializeThunk:
00007ff9`999e3dd0 4053            push    rbx
```

- Команда register watchpoint поддерживает все поддерживаемые архитектуры Windows, такие как x64 и ARM64.
- Точка отслеживания ограничена только регистрами общего назначения и XMM-регистрами.
- Точка отслеживания регистра реализуется с помощью алгоритма, медленного по сравнению с обычной скоростью воспроизведения трассировки. Поиск можно отменить с помощью CTRL+C.

## Разрыв доступа

`!tt ba[-] <rwe> <address> <size> [<address> <size> …]`

Эта команда перейдет к предыдущей или следующей позиции, где один из указанных диапазонов памяти обращается к указанному способу: "R" предназначен для чтения памяти, "W" предназначен для записи памяти, а "E" предназначен для выполнения. Одновременно можно указать более одной опции, т. е. «RW». Это аналогично команде отладчика ba (break on Access).

Примеры. `!tt ba- rw 0x12345678 0x4000` Найдите предыдущую позицию, которая считывается из диапазона памяти. (0x12345678 – 0x12345678 + 0x4000). `!tt ba e 0x7fffe0001234 0x30000` — найдите следующую позицию, которая выполняется из указанного диапазона. Представьте себе, что адрес и диапазон представляют собой диапазон ntdll.dll. Эта команда найдет следующую позицию, в которой ntdll.dll вводится.

- Адреса и размеры должны быть предоставлены в виде чисел. Символы не поддерживаются.

## Прерывание в модуле

Используйте метод навигации с помощью функции прерывания для перемещения вперед или назад на уровне модуля.

`!tt bm[-] [<module> ...]`

Эта команда перемещается во времени на предыдущую или следующую позицию, чтобы выполнить один из указанных модулей.

Если модуль не указан, он будет переходить к предыдущей или следующей точке выполнения за пределами текущего выполняемого модуля.

При использовании этой команды для перемещения по трассировке, поиск в обратном направлении значительно медленнее, чем поиск вперед.

## Примеры вариантов использования

Ниже приведены два примера вариантов использования.

### Вариант использования 1. *Перемещение вперед или назад, пока вы не находитесь в другом модуле в текущем потоке. В этом случае TTD перемещается из dwmcore в модули ucrtbase, идущие вперед, а затем перемещается назад к ntdll.*

Пример трассировки начинается с исключения в модуле dwmcore.

dbgcmd

```
Setting position: 1C46AE:0
(a54.c98): Break instruction exception - code 80000003 (first/second chance not
available)
Time Travel Position: 1C46AE:0
dwmcore!wil_atomic_uint32_compare_exchange_relaxed+0xc [inlined in
dwmcore!wil_details_FeatureReporting_RecordUsageInCache+0x124]:
00007ffa`441e16a0 f0450fb102      lock cmpxchg dword ptr [r10],r8d
ds:00007ffa`444b0450=00000003
```

Выполните командлет !tt bm commmand. Поскольку модуль не указан, перейдите к следующей точке выполнения за пределами текущего выполняемого модуля — ucrtbase.

dbgcmd

```
0:001> !tt bm
Replaying - Currently at 1D1BEE:0 (  2.95%)
Setting position: 1C46AF:61
(a54.c98): Break instruction exception - code 80000003 (first/second chance not
available)
Time Travel Position: 1C46AF:61
ucrtbase!_finite:
00007ffa`4ac3c080 48b9ffffffffffffff7f mov rcx,7FFFFFFFFFFFFFFFh
```

Одношаговая обработка с этого места позволила TTD правильно найти границу модуля.

dbgcmd

```
0:001> t-
Time Travel Position: 1C46AF:60
dwmcore!CBaseExpression::IsExpressionValueValid+0x50:
00007ffa`44124b40 48ff15f9db2700  call    qword ptr [dwmcore!_imp__finite
```

```
(00007ffa`443a2740)] ds:00007ffa`443a2740={ucrtbase!_finite (00007ffa`4ac3c080)}

(a54.c98): Break instruction exception - code 80000003 (first/second chance not
available)
Time Travel Position: 1C46AE:0
dwmcore!wil_atomic_uint32_compare_exchange_relaxed+0xc [inlined in
dwmcore!wil_details_FeatureReporting_RecordUsageInCache+0x124]:
00007ffa`441e16a0 f0450fb102      lock cmpxchg dword ptr [r10],r8d
ds:00007ffa`444b0450=00000003
```

Выполните поиск назад до ближайшей границы модуля.

dbgcmd

```
0:001> !tt bm-
Setting position: 1C46AD:2B1
(a54.c98): Break instruction exception - code 80000003 (first/second chance not
available)
Time Travel Position: 1C46AD:2B1
ntdll!LdrpDispatchUserCallTarget+0x3b:
00007ffa`4d27f24b 48ffe0          jmp     rax
{dwmcore!CKeyframeAnimation::CalculateValueWorker (00007ffa`441264e0)}
```

Затем перейдите с границы модуля к следующему модулю.

dbgcmd

```
0:001> t
Time Travel Position: 1C46AD:2B2
dwmcore!CKeyframeAnimation::CalculateValueWorker:
00007ffa`441264e0 48895c2408      mov     qword ptr [rsp+8],rbx
ss:0000004e`5151f4d0=0000000000000000
```

## Вариант использования 2. *Перемещение вперед, пока не будет введен модуль. В этом примере ntdll.*

dbgcmd

```
0:001> !tt bm ntdll
Replaying - Currently at 1CBF15:0 (  1.66%)
Setting position: 1C46B0:97
(a54.c98): Break instruction exception - code 80000003 (first/second chance not
available)
Time Travel Position: 1C46B0:97
ntdll!LdrpDispatchUserCallTarget:
00007ffa`4d27f210 4c8b1db1c11000  mov     r11,qword ptr
[ntdll!LdrSystemDllInitBlock+0xb8 (00007ffa`4d38b3c8)]
ds:00007ffa`4d38b3c8=00007df503990000
```

Можно использовать `!tt bm- <module>` для поиска назад.

## Модель данных — NextModuleAccess и PrevModuleAccess

Используйте команду dx (выражение объектной модели отладчика) и объекты модели данных NextModuleAccess и PrevModuleAccess, чтобы получить доступ к тем же функциям, которые доступны в команде !tt.

Перейдите к следующей границе доступа к модулю.

dbgcmd

```
0:001> dx @$curprocess.TTD.NextModuleAccess()
Replaying - Currently at 1D1BEE:0 (  2.95%)
@$curprocess.TTD.NextModuleAccess() : [UTID 3] Execute [1C46AE:0] -> [1C46AF:61]
ucrtbase 0x7ffa4ac3c080
    Position         : 1C46AF:61
    OriginalPosition : 1C46AE:0
    UniqueThreadId   : 0x3
    AccessType       : Execute
    Address          : 0x7ffa4ac3c080
    ModuleName       : ucrtbase
```

Перемещайтесь назад к следующей предыдущей границе доступа.

dbgcmd

```
0:001> dx @$curprocess.TTD.PrevModuleAccess()
@$curprocess.TTD.PrevModuleAccess() : [UTID 3] Execute [1C46AE:0] -> [1C46AD:2B1]
ntdll 0x7ffa4d27f24b
    Position         : 1C46AD:2B1
    OriginalPosition : 1C46AE:0
    UniqueThreadId   : 0x3
    AccessType       : Execute
    Address          : 0x7ffa4d27f24b
    ModuleName       : ntdll
```

Перейдите к следующей границе доступа модуля *ntdll* .

dbgcmd

```
0:001> dx @$curprocess.TTD.NextModuleAccess("ntdll")
Replaying - Currently at 1CBF15:0 (  1.66%)
@$curprocess.TTD.NextModuleAccess("ntdll") : [UTID 3] Execute [1C46AE:0] ->
[1C46B0:97] ntdll 0x7ffa4d27f210
    Position         : 1C46B0:97
    OriginalPosition : 1C46AE:0
    UniqueThreadId   : 0x3
```

```
    AccessType       : Execute
    Address          : 0x7ffa4d27f210
    ModuleName       : ntdll
```

# Использование объектов модели отладчика для точек останова TTD

Доступ к функциональности прерывания при доступе к памяти можно получить с помощью команды dx (выражение объектной модели отладчика), окон моделей данных отладчиков, JavaScript и C++. Общие сведения о работе с запросами LINQ и объектами отладчика см. в разделе "Использование LINQ С объектами отладчика".

## TTD.PrevRegisterWrite

`dx @$curthread.TTD.PrevRegisterWrite("<reg>" [, <value>])`

Этот метод выполняет поиск предыдущей позиции, в которой указанный регистр изменил значения (или, если `<value>` задано, принимает указанное значение) в текущем потоке. Это аналогично приведенной `!tt br-` выше команде, за исключением того, что она не автоматически переходит к позиции. Вместо этого он возвращает такие сведения, как показано ниже.

dbgcmd

```
0:000> dx @$curthread.TTD.PrevRegisterWrite("rbx")

@$curthread.TTD.PrevRegisterWrite("rbx")                : [UTID 2] rbx [2C8:0]
0x0 -> [2C8:0] 0x0
    Register         : rbx
    Position         : 2C8:0 [Time Travel]
    Value            : 0x0
    OriginalPosition : 2C8:0 [Time Travel]
    OriginalValue    : 0x0
    UniqueThreadId   : 0x2
```

Аналогичным образом можно использовать @$curthread. TTD. NextRegisterWrite(...) для поиска следующего изменения регистра.

dbgcmd

```
0:000> dx @$curthread.TTD.NextRegisterWrite("rbx")

@$curthread.TTD.NextRegisterWrite("rbx")                : [UTID 2] rbx [2C8:1]
0x0 -> [2C8:2] 0x963127f5a0
```

```
    Register         : rbx
    Position         : 2C8:2 [Time Travel]
    Value            : 0x963127f5a0
    OriginalPosition : 2C8:1 [Time Travel]
    OriginalValue    : 0x0
    UniqueThreadId   : 0x2
```

Используйте ссылки *[Время перемещения]* в выходных данных, чтобы перейти к этой публикации в трассировке.

dbgcmd

```
0:000> dx -s @$create("Debugger.Models.TTD.Position", 712, 0).SeekTo()
(3b24.2d98): Break instruction exception - code 80000003 (first/second chance not
available)
Time Travel Position: 2C8:0
```

## TTD. PrevMemoryAccess

`dx @$curprocess.TTD.PrevMemoryAccess("<rwe>", <address>, <size> [, <address>, <size> …])`

Этот метод ищет предыдущую позицию, где доступ к одному из указанных диапазонов памяти осуществляется указанным способом. Это аналогично приведенной выше команде !tt ba-, за исключением того, что она не автоматически переходит к позиции. Вместо этого он возвращает такие сведения, как показано ниже.

dbgcmd

```
@$curprocess.TTD.PrevMemoryAccess("w", 0x01a16a939820, 0x10) : [UTID 3] Write
0x1a16a939828 4 bytes [1C46AE:0] -> [1C4575:0]
   Position         : 1C4575:0
   OriginalPosition : 1C46AE:0
   UniqueThreadId   : 0x3
   Address          : 0x1a16a939828
   Size             : 0x4
   AccessType       : Write
```

Аналогичным образом можно использовать `@$curprocess.TTD.NextMemoryAccess(…)` для поиска следующего доступа к памяти.

0:000> dx @$curprocess.TTD.NextMemoryAccess("r", 0x00007ff9`95420000, 0xFF)

dbgcmd

```
@$curprocess.TTD.NextMemoryAccess("r", 0x00007ff9`95420000, 0xFF)
: [UTID 2] Read 0x7ff995420000 2 bytes [2C8:0] -> [66C:10A2]
```

```
    Position         : 66C:10A2 [Time Travel]
    OriginalPosition : 2C8:0 [Time Travel]
    UniqueThreadId   : 0x2
    Address          : 0x7ff995420000
    Size             : 0x2
    AccessType       : Read
```

# !tt DLL

ttdext.dll

# Дополнительные сведения

Это расширение работает только с трассировками перемещения по времени. Дополнительные сведения о путешествии по времени см. в разделе "Отладка временных путешествий" — обзор.

# См. также

Отладка временных путешествий — обзор

# !индекс

Статья • 22.11.2024



**Расширение !index** индексирует трассировки перемещения по времени или отображает сведения о состоянии индекса.

dbgsyntax

```
!index [-status] [-force]
```

Используется `!index` для выполнения передачи индексирования по текущей трассировки.

dbgcmd

```
0:000> !index
Indexed 10/14 keyframes
Indexed 14/14 keyframes
Successfully created the index in 535ms.
```

Если текущая трассировка уже индексирована, команда !index ничего не делает.

dbgcmd

```
0:000> !index
Successfully created the index in 0ms.
```

## Параметры

**-статус**

Используется `!index -status` для отчета о состоянии индекса трассировки.

dbgcmd

```
0:000> !index -status
Index file loaded.
```

### -force

Используется `!index -force` для повторного индексирования трассировки, даже если на диске существует выгрузочный файл индекса.

dbgcmd

```
0:000> !index -force
Successfully created the index in 152ms.
```

## DLL-библиотеки

ttdext.dll

## Дополнительная информация

Это расширение работает только с трассировками перемещения по времени. Дополнительные сведения о путешествии по времени см. в разделе "Отладка временных путешествий" — обзор.

## См. также

Отладка временных путешествий — команды расширения

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Позиции

Статья • 01.04.2025



Расширение **!positions** отображает все активные потоки, включая их текущие позиции в трассировке перемещения по времени.

dbgcmd

```
!positions
```

## Параметры

нет

## Пример

В этих выходных данных показаны пять потоков. Поток 1660 — это текущий поток, обозначенный как > в левом столбце.

dbgcmd

```
0:000> !positions
>*Thread ID=0x1660 - Position: 724:0
  Thread ID=0x3E6C - Position: A8B:0
  Thread ID=0x30EC - Position: A8A:0
  Thread ID=0x1F40 - Position: A8E:0
  Thread ID=0x4170 - Position: C44:0
* indicates an actively running thread
```

## DLL

ttdext.dll

## Дополнительная информация

Это расширение работает только с трассировками перемещения по времени. Дополнительные сведения о путешествии по времени см. в разделе "Отладка временных путешествий" — обзор.

## См. также

Отладка методом перемещения во времени — команды для расширения

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Отладка перемещения по времени — пошаговое руководство по приложению

Статья • 18.11.2024



В этой лаборатории представлена отладка временных путешествий (TTD), используя небольшую примерную программу с недостатком кода. TTD используется для отладки, идентификации и первопричины проблемы. Хотя проблема в этой небольшой программе легко найти, общая процедура может использоваться в более сложном коде. Эта общая процедура может быть обобщена следующим образом.

1. Захват трассировки путешествия по времени неудачной программы.
2. Используйте команду dx (выражение объектной модели отладчика), чтобы найти событие исключения, хранящееся в записи.
3. Используйте команду !tt (путешествие по времени) для перемещения в положение события исключения в трассировке.
4. С этого момента один шаг трассировки выполняется обратно до тех пор, пока в области не появится код сбоя.
5. При использовании кода сбоя в области просмотрите локальные значения и разработайте гипотезу переменной, которая может содержать неверное значение.
6. Определите адрес памяти переменной с неправильным значением.
7. Задайте точку останова доступа к памяти (ba) в адресе подозрительной переменной с помощью команды ba (Break on Access).
8. Используйте g-, чтобы вернуться к последней точке доступа к памяти подозрительной переменной.
9. Узнайте, является ли это расположение или несколько инструкций ранее, является точкой ошибки кода. Если да, то все готово. Если неверное значение произошло из другой переменной, задайте еще один разрыв для точки останова доступа во второй переменной.
10. Используйте g-, чтобы вернуться к последней точке доступа к памяти во второй подозрительной переменной. Узнайте, содержит ли это расположение или несколько инструкций, прежде чем содержать ошибку кода. Если да, то все готово.

11. Повторите этот процесс, пока не будет указан код, задающий неверное значение, вызвавшее ошибку.

Хотя общие методы, описанные в этой процедуре, применяются к широкому набору проблем кода, существуют уникальные проблемы кода, требующие уникального подхода. Методы, показанные в пошаговом руководстве, должны служить для расширения набора средств отладки и проиллюстрируют некоторые возможности трассировки TTD.

# Цели лаборатории

После завершения этой лаборатории вы сможете использовать общую процедуру с трассировкой перемещения по времени для поиска проблем в коде.

# Организация занятия

Для выполнения лаборатории потребуется следующее оборудование.

- Ноутбук или настольный компьютер под управлением Windows 10 или Windows 11

Для выполнения лаборатории вам потребуется следующее программное обеспечение.

- The WinDbg. Сведения об установке WinDbg см. в статье "Установка WinDbg"
- Visual Studio для создания примера кода C++.

Лаборатория содержит следующие три раздела.

- Раздел 1. Создание примера кода
- Раздел 2. Запись трассировки примера DisplayGreeting
- Раздел 3. Анализ записи файла трассировки для выявления проблемы с кодом

# Раздел 1. Создание примера кода

*В разделе 1 вы создадите пример кода с помощью Visual Studio.*

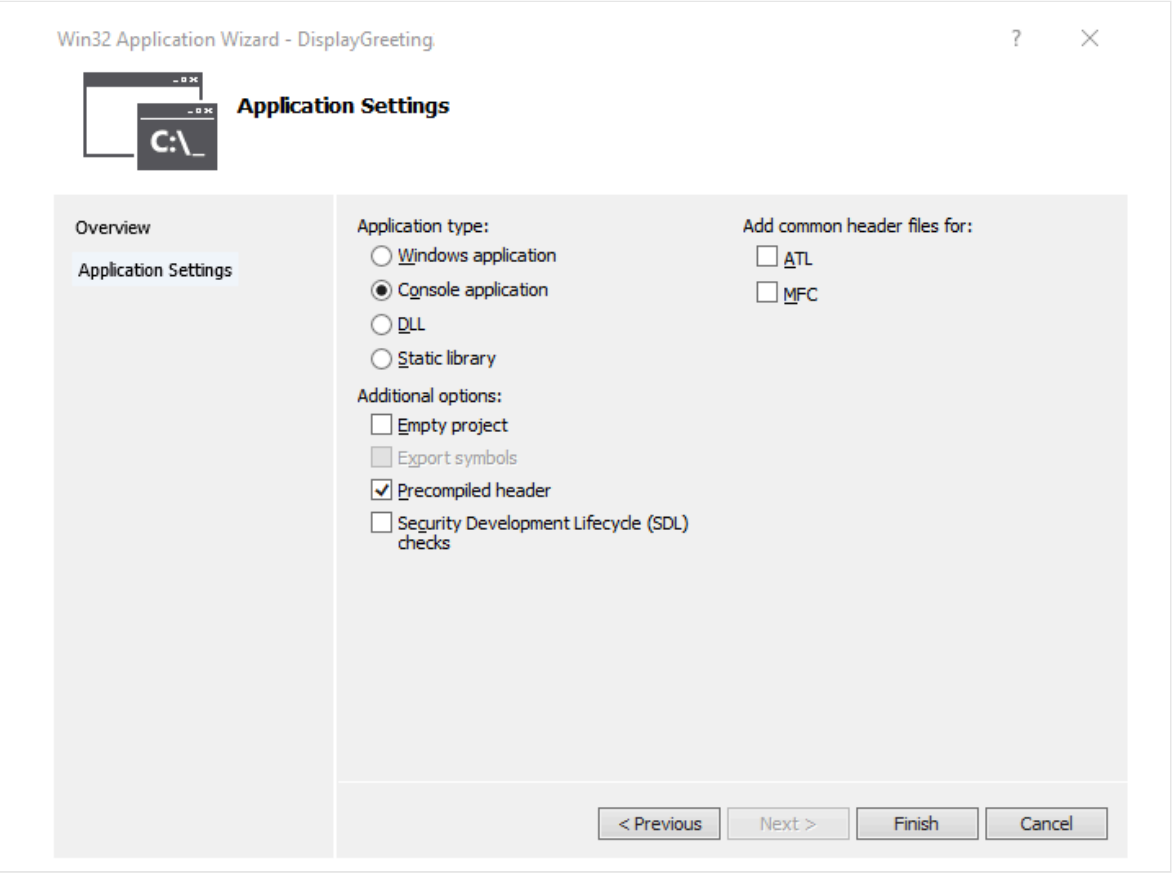
**Создание примера приложения в Visual Studio**

1. В Microsoft Visual Studio щелкните **файл>нового>проекта или решения...** и выберите шаблоны Visual **C++.**

   Выберите консольное приложение Win32.

Укажите имя *проекта DisplayGreeting* и нажмите кнопку "**OK**".

2. Снимите флажок проверки жизненного цикла разработки безопасности (SDL).




3. Нажмите кнопку "**Готово**".

4. Вставьте следующий текст в область DisplayGreeting.cpp в Visual Studio.

C++

```
// DisplayGreeting.cpp : Defines the entry point for the console
application.
//

#include "stdafx.h"
#include <array>
#include <stdio.h>
#include <string.h>

void GetCppConGreeting(wchar_t* buffer, size_t size)
{
    wchar_t const* const message = L"HELLO FROM THE WINDBG TEAM. GOOD
LUCK IN ALL OF YOUR TIME TRAVEL DEBUGGING!";

    wcscpy_s(buffer, size, message);
}

int main()
{
```

```
    std::array <wchar_t, 50> greeting{};
    GetCppConGreeting(greeting.data(), sizeof(greeting));

    wprintf(L"%ls\n", greeting.data());

    return 0;
}
```

5. В Visual Studio щелкните **свойства Project**>DisplayGreeting. Затем щелкните **C/C++** и **создание** кода.

   Задайте следующие свойства.

   Развернуть таблицу

| **Параметр** | **Значение** |
|---|---|
| Проверка безопасности | Отключение проверки безопасности (/GS-) |
| Базовые проверки среды выполнения | По умолчанию |

> **Примечание**
>
> Хотя эти параметры не рекомендуется, можно представить сценарий, в котором кто-то советует использовать эти параметры для ускорения написания кода или упрощения определенных сред тестирования.

6. В Visual Studio щелкните "**Сборка решения сборки**>".

   Если все идет хорошо, окна сборки должны отображать сообщение, указывающее, что сборка выполнена успешно.

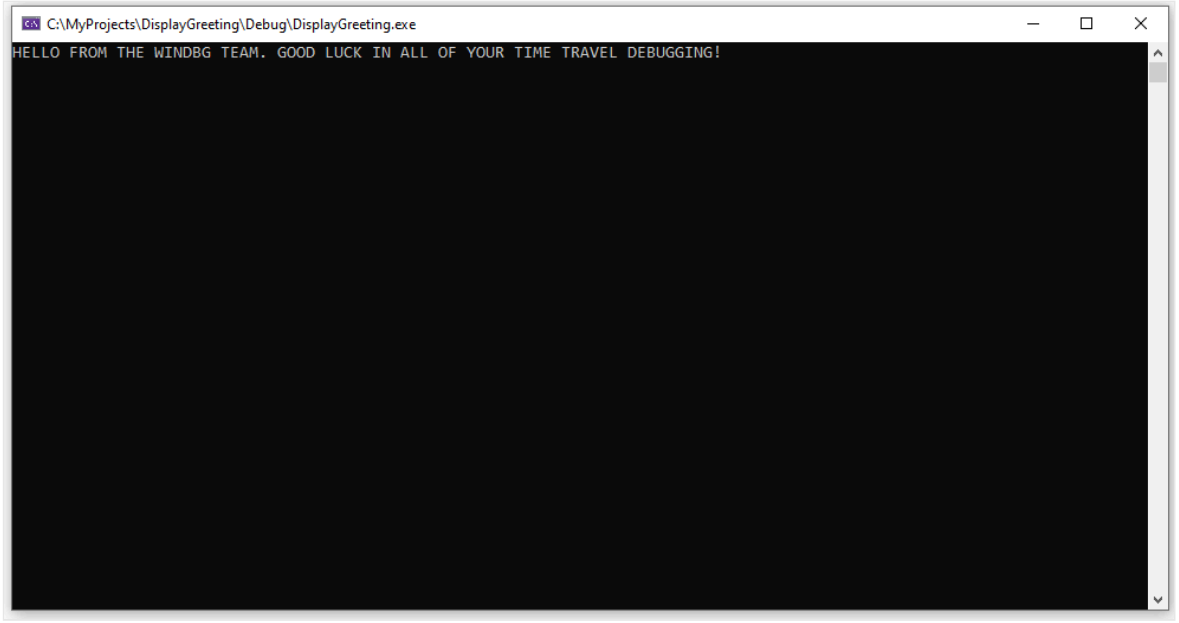
7. **Поиск созданных примеров файлов приложения**

   В Обозреватель решений щелкните правой кнопкой мыши проект DisplayGreeting *и выберите* "***Открыть папку*** *" в проводнике*.

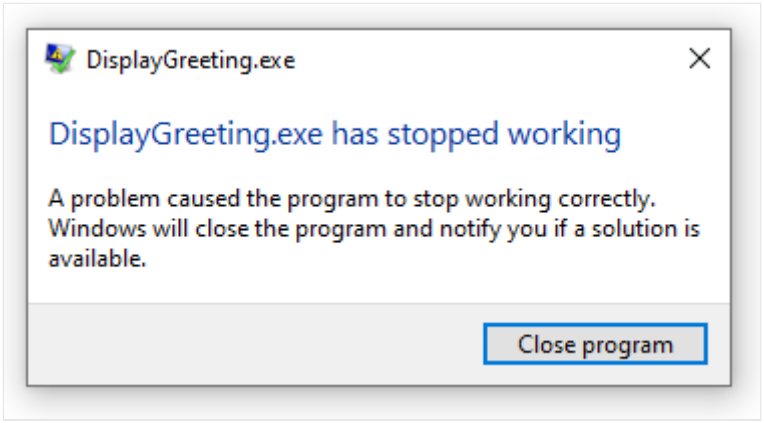
   Перейдите в папку отладки, содержащую соответствующий файл exe и pdb символов для примера. Например, вы перейдете в папку *C:\Projects\DisplayGreeting\Debug*, если это папка, в которой хранятся проекты.

8. **Запуск примера приложения с ошибкой кода**

   Дважды щелкните файл exe, чтобы запустить пример приложения.




Если откроется это диалоговое окно, нажмите кнопку **"Закрыть программу"**




В следующем разделе пошагового руководства мы запишите выполнение примера приложения, чтобы узнать, можно ли определить, почему это исключение происходит.

# Раздел 2. Запись трассировки примера DisplayGreeting

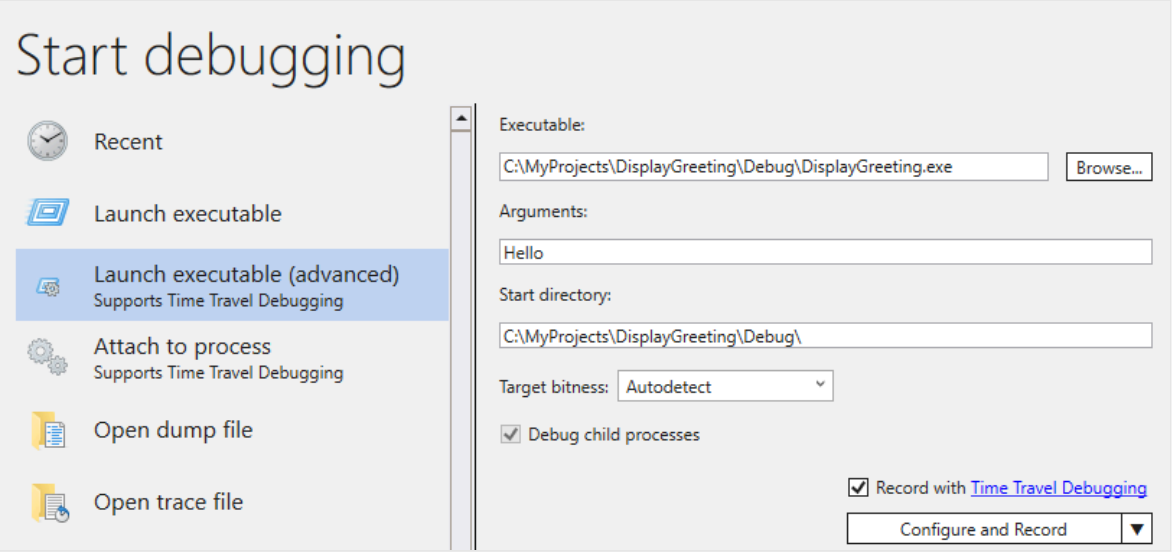
*В разделе 2 вы запишите трассировку неправильного примера приложения DisplayGreeting*

Чтобы запустить пример приложения и записать трассировку TTD, выполните следующие действия. Общие сведения о записях трассировок TTD см. в разделе "Отладка перемещения по времени" — запись трассировки

1. Запустите WinDbg от имени администратора, чтобы иметь возможность записывать трассировки перемещения по времени.

2. В WinDbg выберите **"Запуск запуска>файла" для запуска исполняемого файла>(дополнительно).**

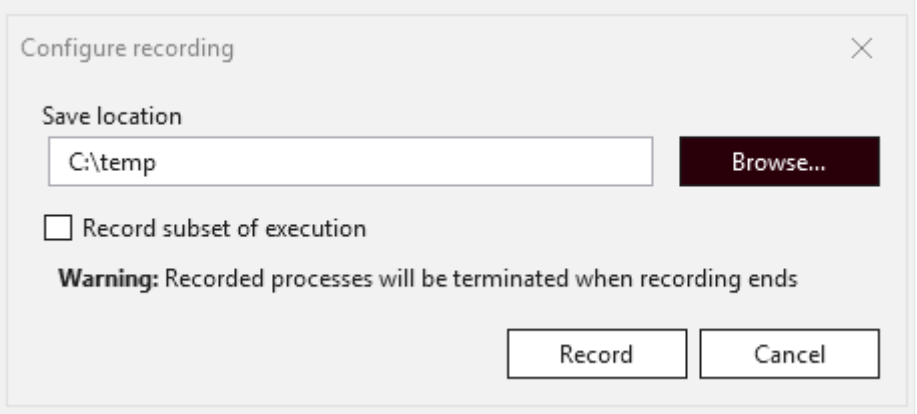
3. Введите путь к исполняемому файлу в пользовательском режиме, который вы хотите записать, или нажмите кнопку **"Обзор** ", чтобы перейти к исполняемому файлу. Сведения о работе с исполняемым меню запуска в WinDbg см. в разделе WinDbg — запуск сеанса пользовательского режима.




4. **Установите флажок "Запись с отладчиком** по времени", чтобы записать трассировку при запуске исполняемого файла.

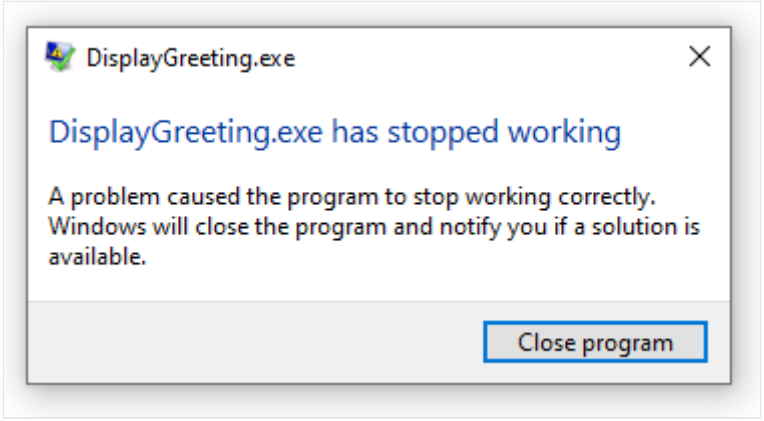
5. Нажмите кнопку **"Настроить" и "Запись", чтобы начать запись** .

6. Когда появится диалоговое окно "Настройка записи", нажмите кнопку " **Запись** ", чтобы запустить исполняемый файл и начать запись.

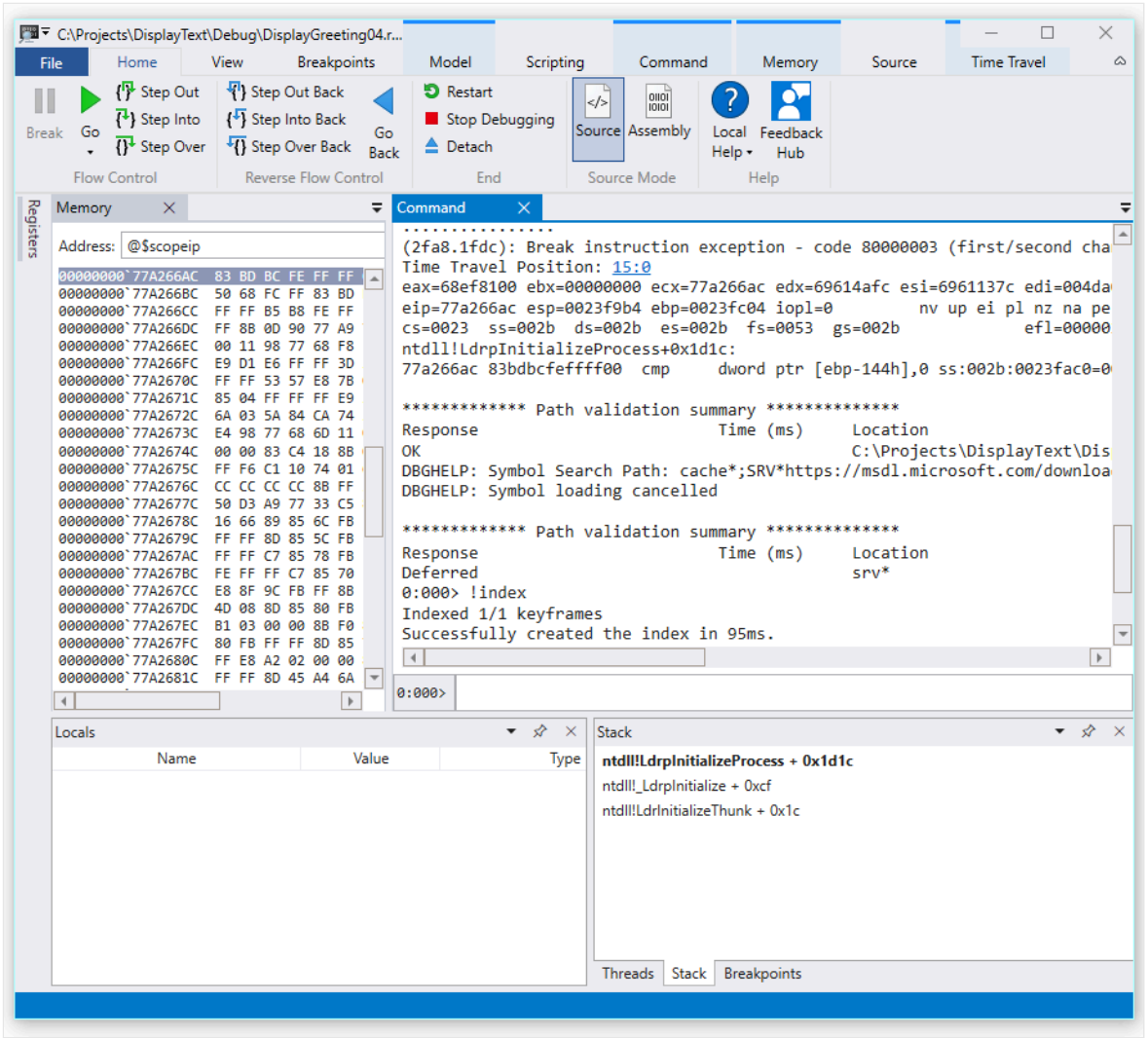



7. Откроется диалоговое окно записи, показывающее, что трассировка записывается. Вскоре после этого приложение завершает работу.

8. **Нажмите кнопку "Закрыть программу**", чтобы закрыть диалоговое окно "DisplayGreeting перестало работать".




9. После сбоя программы файл трассировки будет закрыт и записан на диск.




10. Отладчик автоматически откроет файл трассировки и индексировать его. Индексирование — это процесс, который обеспечивает эффективную отладку файла трассировки. Этот процесс индексирования займет больше времени для больших файлов трассировки.

dbgcmd

```
(5120.2540): Break instruction exception - code 80000003 (first/second
chance not available)
```

```
Time Travel Position: D:0 [Unindexed] Index
!index
Indexed 10/22 keyframes
Indexed 20/22 keyframes
Indexed 22/22 keyframes
Successfully created the index in 755ms.
```

> ⓘ **Примечание**
>
> Ключевой кадр — это расположение трассировки, используемое для индексирования. Ключевые кадры создаются автоматически. Более крупные трассировки будут содержать дополнительные ключевые кадры.

11. На этом этапе вы находитесь в начале файла трассировки и готовы к перемещению вперед и назад во времени.

    Теперь, когда вы записали трассировку TTD, вы можете воспроизвести трассировку обратно или работать с файлом трассировки, например совместно с ним. Дополнительные сведения о работе с файлами трассировки см. в статье "Отладка перемещения по времени" — работа с файлами трассировки

В следующем разделе этой лаборатории мы проанализируем файл трассировки, чтобы найти проблему с нашим кодом.

# Раздел 3. Анализ записи файла трассировки для выявления проблемы с кодом

*В разделе 3 вы проанализируете запись файла трассировки, чтобы определить проблему кода.*

**Настройка среды WinDbg**

1. Добавьте расположение локального символа в путь к символу и перезагрузите их, введя следующие команды.

```
dbgcmd
```

```
.sympath+ C:\MyProjects\DisplayGreeting\Debug
.reload
```

2. Добавьте расположение локального кода в исходный путь, введя следующую команду.

```
dbgcmd
```

```
.srcpath+ C:\MyProjects\DisplayGreeting\DisplayGreeting
```

3. Чтобы просмотреть состояние стека и локальных переменных, на ленте WinDbg выберите "Вид" и "Локальные**" и** "Вид" и "**Стек**". Упорядочение окон для их просмотра, исходного кода и окон команд одновременно.

4. На ленте WinDbg выберите "**Исходный**" и "**Файл с открытым исходным кодом**". Найдите файл DisplayGreeting.cpp и откройте его.

**Проверка исключения**

1. Когда файл трассировки был загружен, он отображает сведения о возникновении исключения.

```
dbgcmd
```

```
2fa8.1fdc): Break instruction exception - code 80000003 (first/second
chance not available)
Time Travel Position: 15:0
eax=68ef8100 ebx=00000000 ecx=77a266ac edx=69614afc esi=6961137c
edi=004da000
eip=77a266ac esp=0023f9b4 ebp=0023fc04 iopl=0         nv up ei pl nz na
pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000206
ntdll!LdrpInitializeProcess+0x1d1c:
77a266ac 83bdbcfeffff00  cmp     dword ptr [ebp-144h],0
ss:002b:0023fac0=00000000
```

2. Используйте команду dx для перечисления всех событий записи. Событие исключения отображается в событиях.

```
dbgcmd
```

```
0:000> dx -r1 @$curprocess.TTD.Events
...
[0x2c]           : Module Loaded at position: 9967:0
[0x2d]           : Exception at 9BDC:0
[0x2e]           : Thread terminated at 9C43:0
...
```

> ⓘ **Примечание**

В этом пошаговом руководстве три периода используются для указания того, что лишние выходные данные были удалены.

3. Щелкните событие исключения, чтобы отобразить сведения об этом событии TTD.

```
dbgcmd

0:000> dx -r1 @$curprocess.TTD.Events[17]
@$curprocess.TTD.Events[17]                 : Exception at 68:0
    Type             : Exception
    Position         : 68:0 [Time Travel]
    Exception        : Exception of type Hardware at PC: 0X540020
```

4. Щелкните поле "Исключение", чтобы продолжить детализацию данных исключений.

```
dbgcmd

0:000> dx -r1 @$curprocess.TTD.Events[17].Exception
@$curprocess.TTD.Events[17].Exception                : Exception of
type Hardware at PC: 0X540020
    Position         : 68:0 [Time Travel]
    Type             : Hardware
    ProgramCounter   : 0x540020
    Code             : 0xc0000005
    Flags            : 0x0
    RecordAddress    : 0x0
```

Данные исключения указывают на то, что это ошибка оборудования, вызываемая ЦП. Он также предоставляет код исключения 0xc0000005, указывающий, что это нарушение доступа. Обычно это означает, что мы пытались записать в память, к которому у нас нет доступа.

5. Щелкните ссылку [Время перемещения] в событии исключения, чтобы перейти к этой позиции в трассировке.

```
dbgcmd

0:000> dx @$curprocess.TTD.Events[17].Exception.Position.SeekTo()
Setting position: 68:0

@$curprocess.TTD.Events[17].Exception.Position.SeekTo()
(16c8.1f28): Break instruction exception - code 80000003 (first/second
chance not available)
Time Travel Position: 68:0
eax=00000000 ebx=00cf8000 ecx=99da9203 edx=69cf1a6c esi=00191046
```

```
edi=00191046
eip=00540020 esp=00effe4c ebp=00520055 iopl=0         nv up ei pl zr na
pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000246
00540020 ??
```

Обратите внимание, что в этом выходных данных стек и базовый указатель указывают на два очень разных адреса.

dbgcmd

```
esp=00effe4c ebp=00520055
```

Это может указывать на повреждение стека, возможно, возвращенную функцию, а затем поврежденную стек. Чтобы проверить это, необходимо вернуться к состоянию ЦП и узнать, можно ли определить, когда произошла ошибка стека.

**Проверьте локальные переменные и задайте точку останова кода**

В точке сбоя трассировки обычно выполняется несколько шагов после истинной причины обработки ошибок. С путешествием по времени мы можем вернуть инструкцию за раз, чтобы найти истинную первопричину.

1. **На ленте "Главная**" используйте **команду "Шаг назад"** для выполнения трех инструкций. По мере этого продолжайте изучать окна стека и памяти.

   В командном окне будут отображаться позиции перемещения по времени и регистры при шаге назад три инструкции.

   dbgcmd

   ```
   0:000> t-
   Time Travel Position: 67:40
   eax=00000000 ebx=00cf8000 ecx=99da9203 edx=69cf1a6c esi=00191046
   edi=00191046
   eip=00540020 esp=00effe4c ebp=00520055 iopl=0         nv up ei pl zr na
   pe nc
   cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
   efl=00000246
   00540020 ??              ???

   0:000> t-
   Time Travel Position: 67:3F
   eax=00000000 ebx=00cf8000 ecx=99da9203 edx=69cf1a6c esi=00191046
   edi=00191046
   eip=0019193d esp=00effe48 ebp=00520055 iopl=0         nv up ei pl zr na
   ```

```
pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000246
DisplayGreeting!main+0x4d:
0019193d c3

0:000> t-
Time Travel Position: 67:39
eax=0000004c ebx=00cf8000 ecx=99da9203 edx=69cf1a6c esi=00191046
edi=00191046
eip=00191935 esp=00effd94 ebp=00effe44 iopl=0         nv up ei pl nz ac
po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000212
DisplayGreeting!main+0x45:
```

> ⓘ **Примечание**
>
> В этом пошаговом руководстве выходные данные команды отображают команды, которые можно использовать вместо параметров меню пользовательского интерфейса, чтобы разрешить пользователям с предпочтениями использования командной строки использовать команды командной строки.

2. На этом этапе трассировки наш стек и базовый указатель имеют значения, которые имеют больше смысла, поэтому, по-видимому, мы приближаемся к точке в коде, где произошла коррупция.

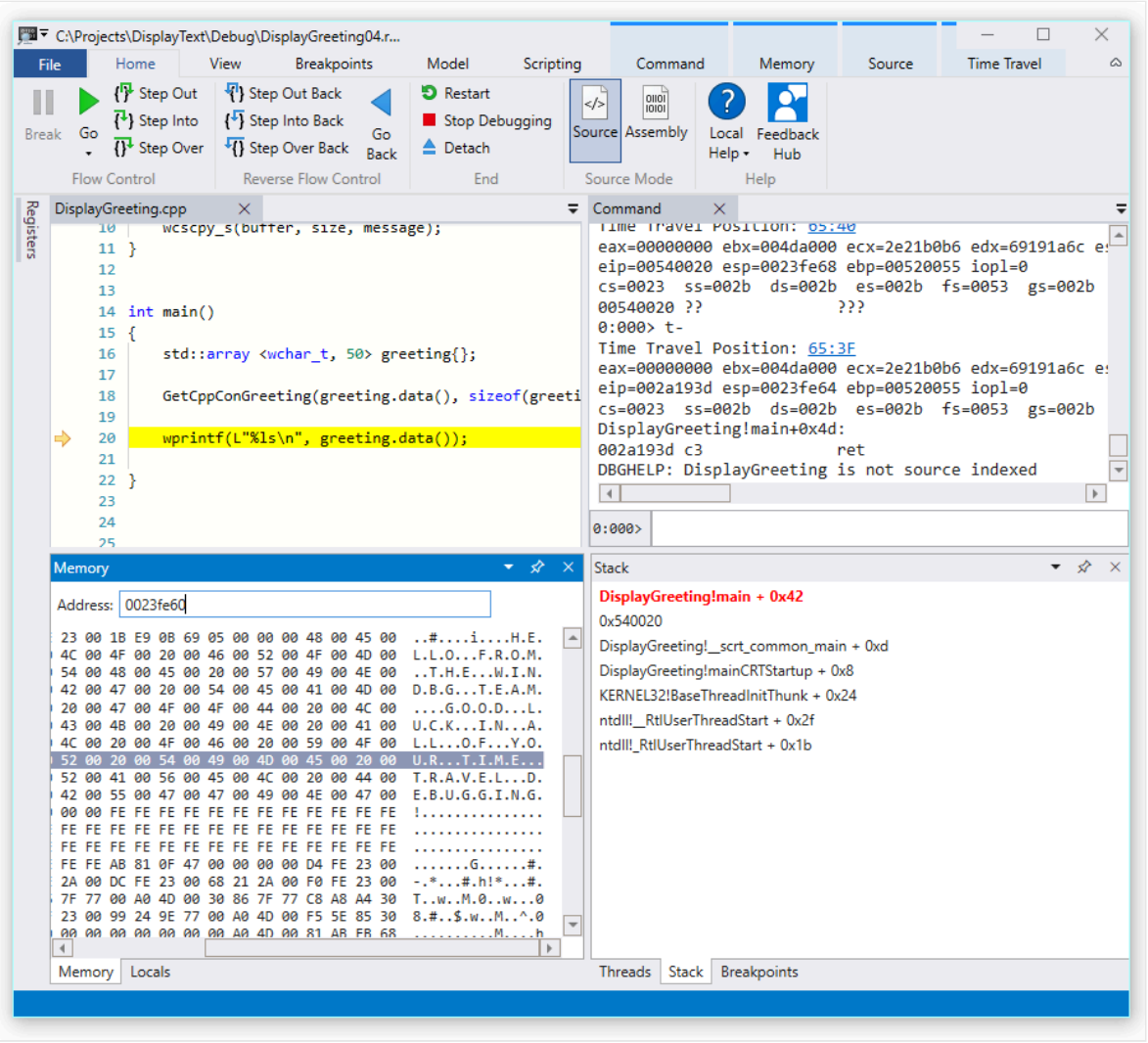
   dbgcmd

   ```
   esp=00effd94 ebp=00effe44
   ```

   Кроме того, в окне локальных параметров содержатся значения из целевого приложения, а окно исходного кода выделяет строку кода, которая готова выполняться на этом этапе трассировки.

3. Для дальнейшего изучения можно открыть окно памяти, чтобы просмотреть содержимое рядом с адресом *памяти базового указателя 0x00effe44*.

4. Чтобы отобразить связанные символы ASCII, на ленте памяти выберите **текст и ASCII**.




5. Вместо базового указателя, указывающего на инструкцию, указывающую на текст сообщения. Так что что-то не так здесь, это может быть близко к моменту времени, что мы повреждены стек. Для дальнейшего изучения мы установим точку останова.

> ⓘ **Примечание**
>
> В этом очень маленьком примере было бы довольно легко просто выглядеть в коде, но если есть сотни строк кода и десятки вложенных шаблонов, описанные здесь, можно использовать для уменьшения времени, необходимого для поиска проблемы.

### Точки останова и TTD

Использование точек останова — это распространенный подход для приостановки выполнения кода на каком-то событии, интересующем вас. TTD позволяет задать точку останова и вернуться назад до тех пор, пока эта точка останова не будет достигнута после записи трассировки. Возможность проверить состояние процесса после того, как возникла проблема, чтобы определить оптимальное расположение

точки останова, позволяет дополнительным рабочим процессам отладки, уникальным для TTD.

### Точки останова доступа к памяти

Точки останова можно задать при доступе к расположению памяти. **Используйте команду ba** (break on access) со следующим синтаксисом.

```
dbgcmd
ba <access> <size> <address> {options}
```

Развернуть таблицу

| Вариант | Описание |
|---|---|
| e | выполнение (при получении инструкции из адреса ЦП) |
| r | чтение и запись (при чтении или записи ЦП в адрес) |
| w | запись (когда ЦП записывает в адрес) |

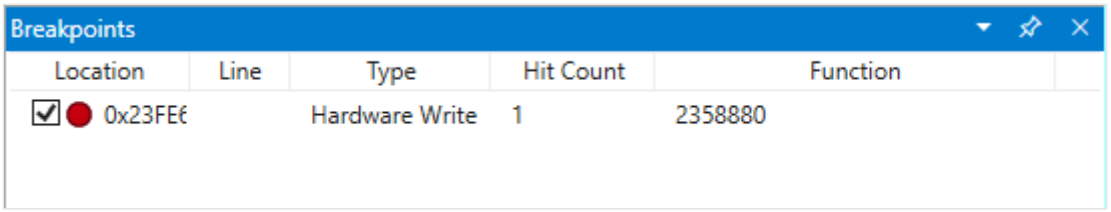
Обратите внимание, что вы можете задать только четыре точки останова данных в любой момент времени, и вам нужно убедиться, что вы правильно выравниваете данные или не активируете точку останова (слова должны заканчиваться адресами, делимыми на 2, dwords должны быть делимыми на 4, а квадроты на 0 или 8).

### Настройка точки останова доступа к памяти для базового указателя

1. На этом этапе трассировки мы хотели бы задать точку останова для доступа к памяти записи к базовому указателю - ebp, который в нашем примере равен 000effe444. Для этого используйте **команду ba** с помощью адреса, который мы хотим отслеживать. Мы хотим отслеживать записи для четырех байтов, поэтому мы указываем w4.

   ```
   dbgcmd
   0:000> ba w4 00effe44
   ```

2. Выберите **"Вид** ", а затем **точки останова, чтобы убедиться, что они заданы** как предполагаемые.

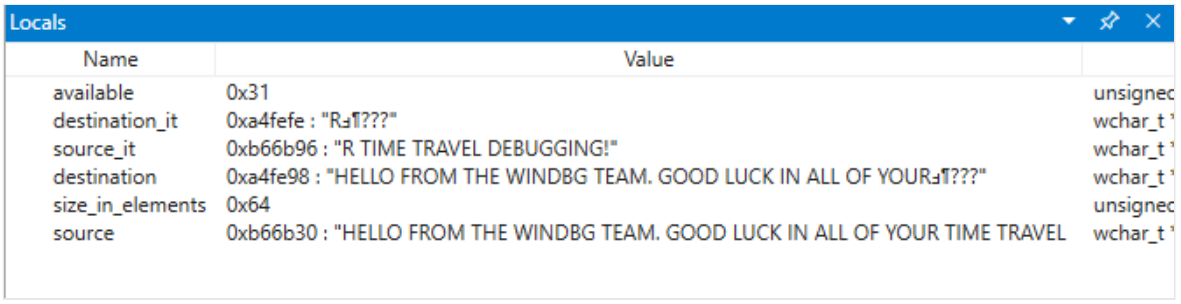



3. В меню "Главная" выберите "**Вернуться** назад", пока точка останова не будет достигнута.

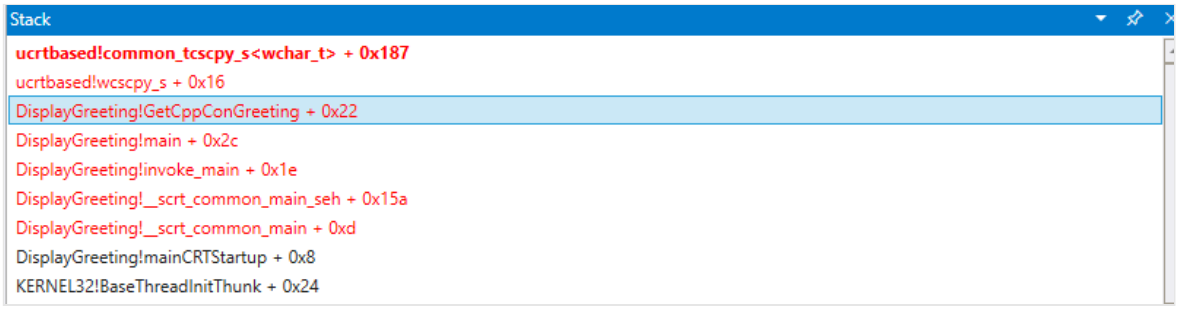
dbgcmd

```
0:000> g-
Breakpoint 0 hit
Time Travel Position: 5B:92
eax=0000000f ebx=003db000 ecx=00000000 edx=00cc1a6c esi=00d41046
edi=0053fde8
eip=00d4174a esp=0053fcf8 ebp=0053fde8 iopl=0         nv up ei pl nz ac
pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000216
DisplayGreeting!DisplayGreeting+0x3a:
00d4174a c745e000000000  mov     dword ptr [ebp-20h],0
ss:002b:0053fdc8=cccccccc
```

4. Выберите "**Вид**" и "**Локальные**". В окне локальных параметров видно, что *целевая* переменная имеет только часть сообщения, в то время как *источник* содержит весь текст. Эта информация поддерживает идею повреждения стека.




5. На этом этапе мы можем проверить стек программ, чтобы узнать, какой код активен. На ленте "Вид" выберите "**Стек**".



Так как маловероятно, что у предоставленной корпорацией Майкрософт функции wscpy_s() будет ошибка кода, как в этом случае, мы ищем дополнительные сведения в стеке. В стеке показано, что приветствие!main вызывает Приветствие! GetCppConGreeting. В нашем очень маленьком примере кода мы могли просто открыть код на этом этапе и, скорее всего, найти ошибку довольно легко. Но чтобы проиллюстрировать методы, которые можно использовать с более крупной, более сложной программой, мы установим новую точку останова для дальнейшего изучения.

**Настройка точки останова доступа для функции GetCppConGreeting**

1. Используйте окно точек останова, чтобы очистить существующую точку останова, щелкнув правой кнопкой мыши существующую точку останова и выбрав "**Удалить**".

2. Определите адрес DisplayGreeting! Функция GetCppConGreeting с помощью **команды DX** .

   dbgcmd

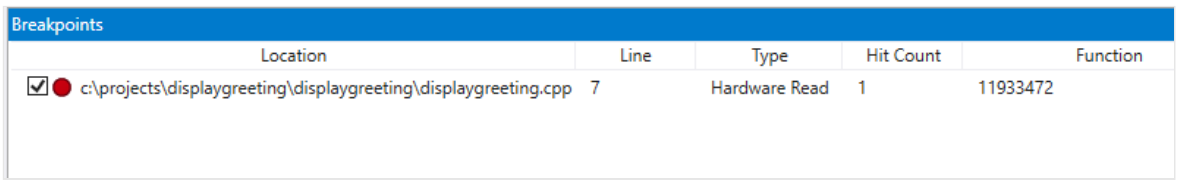
   ```
   0:000> dx &DisplayGreeting!GetCppConGreeting
   &DisplayGreeting!GetCppConGreeting                 : 0xb61720 [Type:
   void (__cdecl*)(wchar_t *,unsigned int)]
       [Type: void __cdecl(wchar_t *,unsigned int)]
   ```

3. **Используйте команду ba**, чтобы задать точку останова для доступа к памяти. Так как функция будет только считываться из памяти для выполнения, нам нужно задать точку останова r - чтение.

   dbgcmd

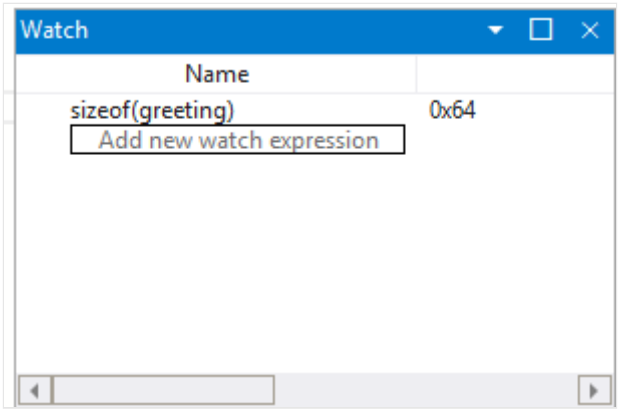
   ```
   0:000> ba r4 b61720
   ```

4. Убедитесь, что точка останова "Аппаратное чтение" активна в окне точек останова.

5. Так как мы задаемся вопросом о размере строки приветствия, мы зададим окно просмотра, чтобы отобразить значение sizeof(приветствие). На ленте "Вид" выберите "**Смотреть**" и укажите *sizeof(приветствие)*. Если значение не

находится в области, откроется окно наблюдения. *Не удается привязать имя "приветствие"*.




6. В меню "Путешествие по времени" используйте **время для запуска** или использования `!tt 0` команды, чтобы перейти к началу трассировки.

```
dbgcmd
```

```
0:000> !tt 0
Setting position to the beginning of the trace
Setting position: 15:0
(1e5c.710): Break instruction exception - code 80000003 (first/second
chance not available)
Time Travel Position: 15:0
eax=68e28100 ebx=00000000 ecx=77a266ac edx=69e34afc esi=69e3137c
edi=00fa2000
eip=77a266ac esp=00ddf3b8 ebp=00ddf608 iopl=0         nv up ei pl nz na
pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000206
ntdll!LdrpInitializeProcess+0x1d1c:
77a266ac 83bdbcfeffff00  cmp     dword ptr [ebp-144h],0
ss:002b:00ddf4c4=00000000
```

7. В меню "Главная" выберите "**Перейти** " или используйте `g` команду, чтобы перейти вперед в коде до тех пор, пока точка останова не будет достигнута.

```
dbgcmd
```

```
0:000> g
Breakpoint 2 hit
Time Travel Position: 4B:1AD
eax=00ddf800 ebx=00fa2000 ecx=00ddf800 edx=00b61046 esi=00b61046
edi=00b61046
eip=00b61721 esp=00ddf7a4 ebp=00ddf864 iopl=0         nv up ei pl nz na
po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000202
```

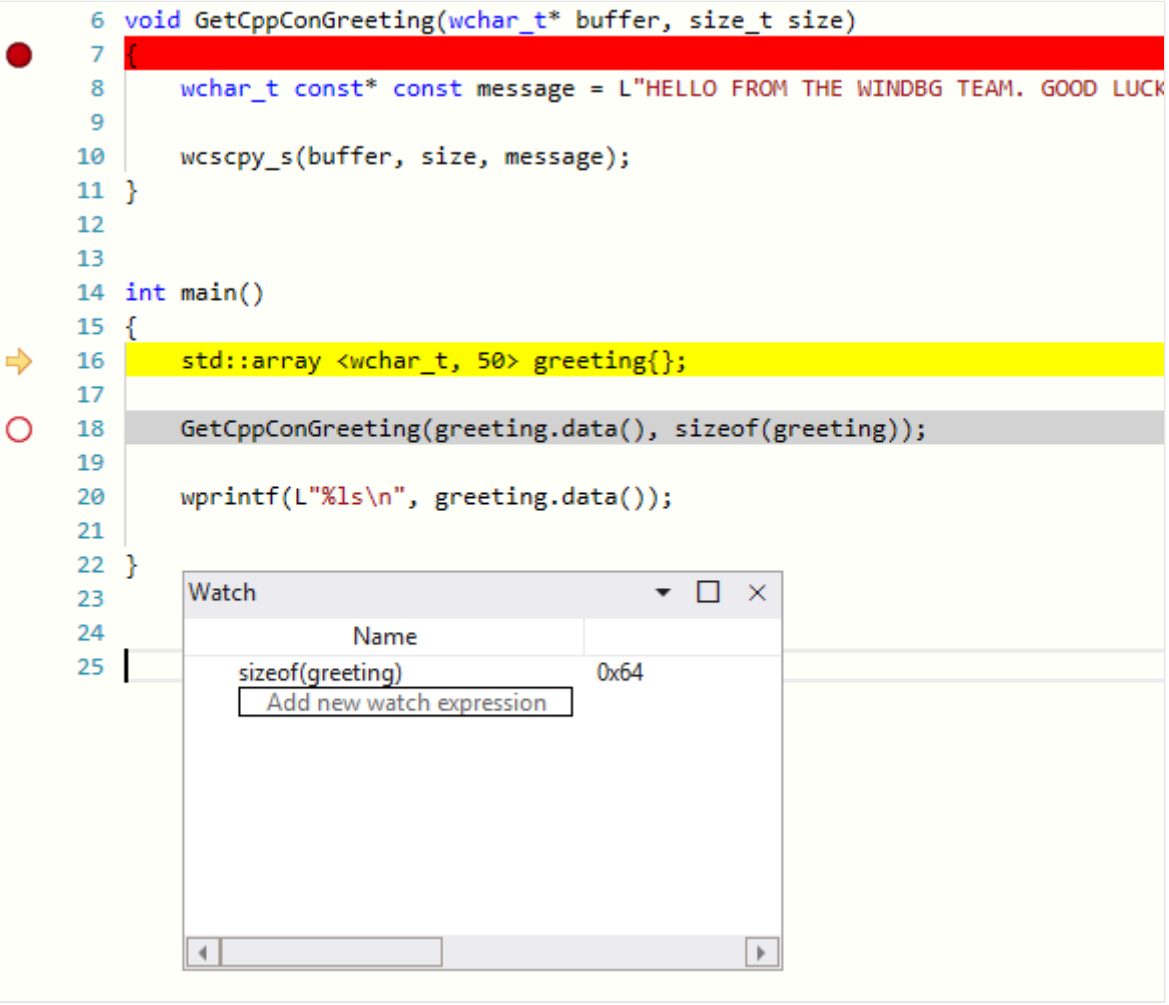
```
DisplayGreeting!GetCppConGreeting+0x1:
00b61721 8bec            mov     ebp,esp
```

8. В меню "Главная" выберите "**Шаг назад**" или используйте `g-u` команду, чтобы вернуть один шаг.

dbgcmd

```
0:000> g-u
Time Travel Position: 4B:1AA
eax=00ddf800 ebx=00fa2000 ecx=00ddf800 edx=00b61046 esi=00b61046
edi=00b61046
eip=00b61917 esp=00ddf7ac ebp=00ddf864 iopl=0         nv up ei pl nz na
po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000202
DisplayGreeting!main+0x27:
00b61917 e8def7ffff      call    DisplayGreeting!ILT+245(?
GetCppConGreetingYAXPA_WIZ) (00b610fa)
```

9. Похоже, мы нашли первопричину. Массив *приветствий* , объявленный нами, составляет 50 символов в длину, а размер (приветствие), который мы передаваем в GetCppConGreeting, 0x64, 100.



Как мы рассмотрим проблему размера, мы также замечаем, что сообщение имеет длину 75 символов и равно 76 при включении конца строкового символа.

```
dbgcmd
```

```
HELLO FROM THE WINDBG TEAM. GOOD LUCK IN ALL OF YOUR TIME TRAVEL
DEBUGGING!
```

10. Одним из способов исправления кода будет расширение размера массива символов до 100.

```
C++
```

```
std::array <wchar_t, 100> greeting{};
```

Кроме того, необходимо изменить sizeof(greeting) на size(greeting) в этой строке кода.

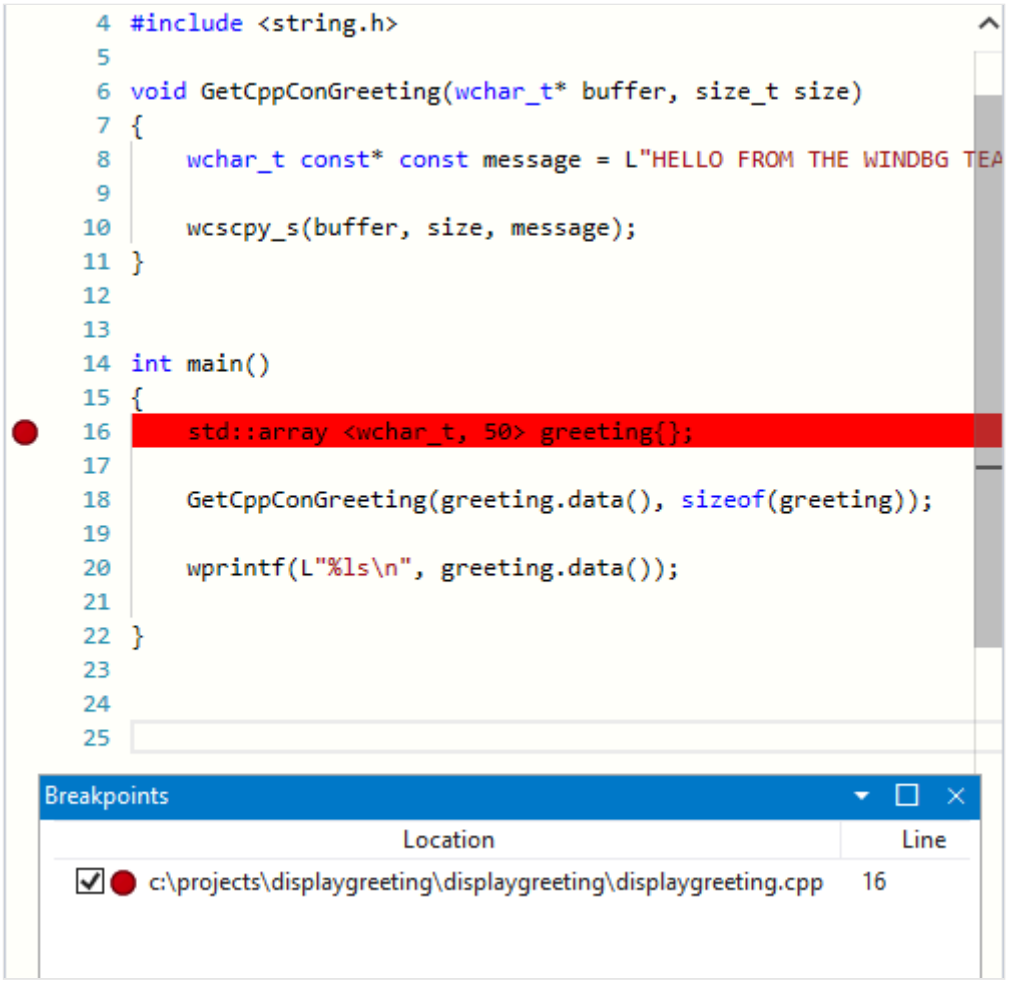
```
C++
```

```
 GetCppConGreeting(greeting.data(), size(greeting));
```

11. Чтобы проверить эти исправления, мы можем перекомпилировать код и убедиться, что он выполняется без ошибок.

**Настройка точки останова с помощью исходного окна**

1. Альтернативным способом выполнения этого исследования будет установка точки останова, щелкнув любую строку кода. Например, щелкнув справа от строки определения std:array в окне источника, установите точку останова.




2. В меню "Путешествие по времени" используйте **команду "Время" для запуска** команды, чтобы перейти к началу трассировки.

```
dbgcmd
```

```
0:000> !tt 0
Setting position to the beginning of the trace
Setting position: 15:0
(1e5c.710): Break instruction exception - code 80000003 (first/second
chance not available)
Time Travel Position: 15:0
eax=68e28100 ebx=00000000 ecx=77a266ac edx=69e34afc esi=69e3137c
edi=00fa2000
eip=77a266ac esp=00ddf3b8 ebp=00ddf608 iopl=0         nv up ei pl nz na
pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000206
ntdll!LdrpInitializeProcess+0x1d1c:
77a266ac 83bdbcfeffff00  cmp     dword ptr [ebp-144h],0
ss:002b:00ddf4c4=00000000
```

3. На домашней ленте нажмите кнопку "**Перейти** ", чтобы вернуться, пока точка останова не будет достигнута.

```
dbgcmd
```

```
Breakpoint 0 hit
Time Travel Position: 5B:AF
eax=0000000f ebx=00c20000 ecx=00000000 edx=00000000 esi=013a1046
edi=00effa60
eip=013a17c1 esp=00eff970 ebp=00effa60 iopl=0         nv up ei pl nz na
po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000202
DisplayGreeting!DisplayGreeting+0x41:
013a17c1 8bf4            mov     esi,esp
```

**Установка точки останова для точки останова доступа для переменной *приветствия***

Другим альтернативным способом выполнения этого исследования будет установка точки останова на подозрительных переменных и изучение того, какой код изменяет их. Например, чтобы задать точку останова в переменной приветствия в методе GetCppConGreeting, используйте эту процедуру.

В этом пошаговом руководстве предполагается, что вы все еще находитесь в точке останова из предыдущего раздела.

1. В **представлении** , а затем **в локальном режиме**. В окне *локальных параметров приветствие* доступно в текущем контексте, поэтому мы сможем определить расположение памяти.

2. **Используйте команду DX** для проверки массива *приветствия*.

   dbgcmd

   ```
   0:000> dx &greeting
   &greeting                 : ddf800 : { size=50 } [Type:
   std::array<wchar_t,50> *]
      [<Raw View>]     [Type: std::array<wchar_t,50>]
      [0]              : 3 [Type: wchar_t]
      [1]              : 0 [Type: wchar_t]
   ```

   В этой трассировке *приветствие* находится в памяти в ddf800.

3. Используйте окно точек останова, чтобы очистить любую существующую точку останова, щелкнув правой кнопкой мыши существующую точку останова и выбрав "**Удалить**".

4. Задайте точку останова с **помощью команды ba** с помощью адреса памяти, который мы хотим отслеживать для доступа на запись.

   ```
   ```

dbgcmd

```
ba w4 ddf800
```

5. В меню "Путешествие по времени" используйте **команду "Время" для запуска** команды, чтобы перейти к началу трассировки.

dbgcmd

```
0:000> !tt 0
Setting position to the beginning of the trace
Setting position: 15:0
(1e5c.710): Break instruction exception - code 80000003 (first/second
chance not available)
Time Travel Position: 15:0
eax=68e28100 ebx=00000000 ecx=77a266ac edx=69e34afc esi=69e3137c
edi=00fa2000
eip=77a266ac esp=00ddf3b8 ebp=00ddf608 iopl=0         nv up ei pl nz na
pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000206
ntdll!LdrpInitializeProcess+0x1d1c:
77a266ac 83bdbcfeffff00  cmp     dword ptr [ebp-144h],0
ss:002b:00ddf4c4=00000000
```

6. В меню "Главная" выберите **перейти** к первой точке доступа к памяти массива приветствий.

dbgcmd

```
0:000> g-
Breakpoint 0 hit
Time Travel Position: 5B:9C
eax=cccccccc ebx=002b1000 ecx=00000000 edx=68d51a6c esi=013a1046
edi=001bf7d8
eip=013a1735 esp=001bf6b8 ebp=001bf7d8 iopl=0         nv up ei pl nz na
po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000202
DisplayGreeting!GetCppConGreeting+0x25:
013a1735 c745ec04000000  mov     dword ptr [ebp-14h],4
ss:002b:001bf7c4=cccccccc
```

Кроме того, мы могли бы переместиться в конец трассировки и работали в обратном направлении с помощью кода, чтобы найти последнюю точку в трассировке, в которую была записана папка памяти массива.

**Используйте TTD. Объекты памяти для просмотра доступа к памяти**

Другой способ определить, к каким точкам в памяти трассировки был доступ, — использовать TTD. Объекты памяти и команда DX.

1. **Используйте команду DX** для проверки массива *приветствия*.

   dbgcmd

   ```
   0:000> dx &greeting
   &greeting                 : 0xddf800 [Type: std::array<wchar_t,50> *]
      [+0x000] _Elems           : "괍棶櫆瞝???" [Type: wchar_t [50]]
   ```

   В этой трассировке *приветствие* находится в памяти в ddf800.

2. **Используйте команду DX**, чтобы просмотреть четыре байта в памяти, начиная с этого адреса с доступом на чтение записи.

   dbgcmd

   ```
   0:000> dx -r1 @$cursession.TTD.Memory(0xddf800,0xddf804, "rw")
   @$cursession.TTD.Memory(0x1bf7d0,0x1bf7d4, "rw")
       [0x0]
       [0x1]
       [0x2]
       [0x3]
       [0x4]
       [0x5]
       [0x6]
       [0x7]
       [0x8]
       [0x9]
       [0xa]
       ...
   ```

3. Щелкните любой из вхождения, чтобы отобразить дополнительные сведения об этом вхождения доступа к памяти.

   dbgcmd

   ```
   0:000> dx -r1 @$cursession.TTD.Memory(0xddf800,0xddf804, "rw")[5]
   @$cursession.TTD.Memory(0xddf800,0xddf804, "rw")[5]
       EventType        : MemoryAccess
       ThreadId         : 0x710
       UniqueThreadId   : 0x2
       TimeStart        : 27:3C1 [Time Travel]
       TimeEnd          : 27:3C1 [Time Travel]
       AccessType       : Write
       IP               : 0x6900432f
       Address          : 0xddf800
       Size             : 0x4
   ```

```
    Value            : 0xddf818
    OverwrittenValue : 0x0
    SystemTimeStart  : Monday, November 18, 2024 23:01:43.400
    SystemTimeEnd    : Monday, November 18, 2024 23:01:43.400
```

4. Щелкните [Время перемещения] для timeStart, чтобы разместить трассировку в момент времени.

dbgcmd

```
0:000> dx @$cursession.TTD.Memory(0xddf800,0xddf804, "rw")
[5].TimeStart.SeekTo()
@$cursession.TTD.Memory(0xddf800,0xddf804, "rw")[5].TimeStart.SeekTo()
(1e5c.710): Break instruction exception - code 80000003 (first/second
chance not available)
Time Travel Position: 27:3C1
eax=00ddf81c ebx=00fa2000 ecx=00ddf818 edx=ffffffff esi=00000000
edi=00b61046
eip=6900432f esp=00ddf804 ebp=00ddf810 iopl=0         nv up ei pl nz ac
po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00000212
ucrtbased!_register_onexit_function+0xf:
6900432f 51              push    ecx
```

5. Если мы заинтересованы в последнем вхождения доступа к памяти чтения и записи в трассировке, мы можем щелкнуть последний элемент в списке или добавить его. Функция Last() до конца команды DX.

dbgcmd

```
0:000> dx -r1 @$cursession.TTD.Memory(0xddf800,0xddf804, "rw").Last()
@$cursession.TTD.Memory(0xddf800,0xddf804, "rw").Last()
    EventType        : MemoryAccess
    ThreadId         : 0x710
    UniqueThreadId   : 0x2
    TimeStart        : 53:100E [Time Travel]
    TimeEnd          : 53:100E [Time Travel]
    AccessType       : Read
    IP               : 0x690338e4
    Address          : 0xddf802
    Size             : 0x2
    Value            : 0x45
    SystemTimeStart  : Monday, November 18, 2024 23:01:43.859
    SystemTimeEnd    : Monday, November 18, 2024 23:01:43.859
```

6. Затем мы можем щелкнуть [Время путешествия], чтобы перейти к этой позиции в трассировке и узнать больше о выполнении кода на этом этапе, используя методы, описанные ранее в этой лаборатории.

Дополнительные сведения о TTD. Объекты памяти см. в разделе TTD. Объект памяти.

## Итоги

В этом очень маленьком примере проблема могла быть определена путем просмотра нескольких строк кода, но в более крупных программах методы, представленные здесь, можно использовать для уменьшения времени, необходимого для поиска проблемы.

После записи трассировки можно совместно использовать шаги трассировки и повтора, и проблема будет воспроизведена на любом компьютере.

## См. также

Отладка временных путешествий — обзор

Отладка перемещения по времени — запись

Отладка перемещения по времени — воспроизведение трассировки

Отладка перемещения по времени — работа с файлами трассировки

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Общие сведения о объектах отладки временных путешествий

Статья • 20.11.2024



В этом разделе описывается, как использовать модель данных для запроса трассировок перемещения по времени. Это может быть полезным инструментом для ответов на такие вопросы, как эти коды, захваченные в трассировке времени.

- Какие исключения находятся в трассировке?
- В какой момент времени трассировки загружается определенный модуль кода?
- Когда были созданы или завершены потоки в трассировке?
- Что такое самые длительные потоки в трассировке?

Существуют расширения TTD, добавляющие данные в *объекты модели сеансов* и *обработки* данных. К объектам модели данных TTD можно получить доступ с помощью команды dx (выражение объектной модели отладчика), окон моделей WinDbg, JavaScript и C++. Расширения TTD автоматически загружаются при отладке трассировки перемещения по времени.

## Объекты обработки

Основные объекты, добавленные в *объекты Process* , можно найти в *пространстве имен TTD* вне любого *объекта Process* . Например, `@$curprocess.TTD`.

dbgcmd

```
0:000> dx @$curprocess.TTD
@$curprocess.TTD
    Index
    Threads
    Events
    DebugOutput
    Lifetime         : [2C8:0, 16EC:98A]
    DefaultMemoryPolicy : InFragmentAggressive
    SetPosition      [Sets the debugger to point to the given position on
this process.]
    GatherMemoryUse  [0]
    RecordClients
```

```
    PrevMemoryAccess [(accessMask, address, size [, address, size, ...]) -
Find the previous matching memory access before current position.]
    NextMemoryAccess [(accessMask, address, size [, address, size, ...]) -
Find the next matching memory access after current position.]
    Recorder
```

Общие сведения о работе с запросами LINQ и объектами отладчика см. в разделе "Использование LINQ С объектами отладчика".

## Свойства

Развернуть таблицу

| Объект | Description |
|---|---|
| Время существования | Объект диапазона TTD, описывающий время существования всей трассировки. |
| Потоки | Содержит коллекцию объектов потоков TTD, по одному для каждого потока в течение всего времени существования трассировки. |
| События | Содержит коллекцию объектов событий TTD, по одному для каждого события в трассировке. |

## Методы

Развернуть таблицу

| Метод | Description |
|---|---|
| SetPosition() | Принимает целое число от 0 до 100 или строк в форме N:N в качестве входных данных и переходит к трассировке в это расположение. Дополнительные сведения см. в разделе !tt . |

# Объекты сеанса

Основные объекты, добавленные в объекты сеанса, *можно найти в пространстве имен TTD вне любого объекта Session*. Например: `@$cursession.TTD`.

dbgcmd

```
0:000> dx @$cursession.TTD
@$cursession.TTD
    Calls            [Returns call information from the trace for the
specified set of methods: TTD.Calls("module!method1", "module!method2", ...)
```

```
For example: dx @$cursession.TTD.Calls("user32!SendMessageA")]
    Memory          [Returns memory access information for specified
address range: TTD.Memory(startAddress, endAddress [, "rwec"])]
    MemoryForPositionRange [Returns memory access information for specified
address range and position range: TTD.MemoryForPositionRange(startAddress,
endAddress [, "rwec"], minPosition, maxPosition)]
    PinObjectPosition [Pins an object to the given time position:
TTD.PinObjectPosition(obj, pos)]
    AsyncQueryEnabled : false
    RichQueryTypesEnabled : true
    DefaultParameterCount : 0x4
    Data            : Normalized data sources based on the contents of the
time travel trace
    Utility         : Methods that can be useful when analyzing time travel
traces
    Analyzers       : Methods that perform code analysis on the time travel
trace
    Bookmarks       : Bookmark collection
    Checkers        : Checkers (scripts for detection of common issues
recorded in a time travel trace)
```

> **Примечание**
>
> Существуют некоторые объекты и методы, добавленные TTDAnalyze, которые используются для внутренних функций расширения. Не все пространства имен документируются, а текущие пространства имен будут развиваться со временем.

## Методы

Развернуть таблицу

| Метод | Description |
|---|---|
| Data.Heap() | Коллекция объектов кучи, выделенных во время трассировки. Обратите внимание, что это функция, которая выполняет вычисления, поэтому требуется некоторое время для выполнения. |
| Вызовы() | Возвращает коллекцию вызовов объектов , соответствующих входной строке. Входная строка может содержать подстановочные знаки. Обратите внимание, что это функция, которая выполняет вычисления, поэтому требуется некоторое время для выполнения. |
| Memory() | Это метод, который принимает beginAddress, endAddress и параметры dataAccessMask и возвращает коллекцию объектов памяти. Обратите внимание, что это функция, которая выполняет вычисления, поэтому требуется некоторое время для выполнения. |

# Сортировка выходных данных запроса

Используйте метод OrderBy() для сортировки строк, возвращаемых из запроса по одному или нескольким столбцам. В этом примере выполняется сортировка по TimeStart в порядке возрастания.

dbgcmd

```
0:000> dx -r2 @$cursession.TTD.Calls("kernelbase!GetLastError").OrderBy(c =>
c.TimeStart)
@$cursession.TTD.Calls("kernelbase!GetLastError").OrderBy(c => c.TimeStart)
    [0x0]
        EventType        : 0x0
        ThreadId         : 0x2d98
        UniqueThreadId   : 0x2
        TimeStart        : 718:7D7 [Time Travel]
        TimeEnd          : 718:7DA [Time Travel]
        Function         : KERNELBASE!GetLastError
        FunctionAddress  : 0x7ff996cf20f0
        ReturnAddress    : 0x7ff99855ac5a
        ReturnValue      : 0x0 [Type: unsigned long]
        Parameters
        SystemTimeStart  : Friday, January 12, 2024 21:18:40.862
        SystemTimeEnd    : Friday, January 12, 2024 21:18:40.862
    [0x1]
        EventType        : 0x0
        ThreadId         : 0x2d98
        UniqueThreadId   : 0x2
        TimeStart        : 72D:1B3 [Time Travel]
        TimeEnd          : 72D:1B6 [Time Travel]
        Function         : KERNELBASE!GetLastError
        FunctionAddress  : 0x7ff996cf20f0
        ReturnAddress    : 0x7ff9961538df
        ReturnValue      : 0x57 [Type: unsigned long]
        Parameters
        SystemTimeStart  : Friday, January 12, 2024 21:18:40.862
        SystemTimeEnd    : Friday, January 12, 2024 21:18:40.862
...
```

Для отображения дополнительной глубины объектов модели данных используется параметр уровня рекурсии -r2. Дополнительные сведения о параметрах команды dx см. в разделе dx (Выражение объектной модели отладчика отладчика).

В этом примере выполняется сортировка по TimeStart в порядке убывания.

dbgcmd

```
0:000> dx -r2
@$cursession.TTD.Calls("kernelbase!GetLastError").OrderByDescending(c =>
c.TimeStart)
```

```
@$cursession.TTD.Calls("kernelbase!GetLastError").OrderByDescending(c =>
c.TimeStart)
    [0x1896]
        EventType        : Call
        ThreadId         : 0x3a10
        UniqueThreadId   : 0x2
        TimeStart        : 464224:34 [Time Travel]
        TimeEnd          : 464224:37 [Time Travel]
        Function         : UnknownOrMissingSymbols
        FunctionAddress  : 0x7561ccc0
        ReturnAddress    : 0x7594781c
        ReturnValue      : 0x0
        Parameters
    [0x18a0]
        EventType        : Call
        ThreadId         : 0x3a10
        UniqueThreadId   : 0x2
        TimeStart        : 464223:21 [Time Travel]
        TimeEnd          : 464223:24 [Time Travel]
        Function         : UnknownOrMissingSymbols
        FunctionAddress  : 0x7561ccc0
        ReturnAddress    : 0x7594781c
        ReturnValue      : 0x0
        Parameters
```

# Указание элементов в запросе

Чтобы выбрать определенный элемент, можно добавить к запросу различные квалификаторы. Например, в запросе отображается первый вызов, содержащий "kernelbase! GetLastError".

dbgcmd

```
0:000> dx @$cursession.TTD.Calls("kernelbase!GetLastError").First()
@$cursession.TTD.Calls("kernelbase!GetLastError").First()
    EventType        : 0x0
    ThreadId         : 0x2d98
    UniqueThreadId   : 0x2
    TimeStart        : 718:7D7 [Time Travel]
    TimeEnd          : 718:7DA [Time Travel]
    Function         : KERNELBASE!GetLastError
    FunctionAddress  : 0x7ff996cf20f0
    ReturnAddress    : 0x7ff99855ac5a
    ReturnValue      : 0x0 [Type: unsigned long]
    Parameters
    SystemTimeStart  : Friday, January 12, 2024 21:18:40.862
    SystemTimeEnd    : Friday, January 12, 2024 21:18:40.862
```

# Фильтрация в запросе

Используйте метод Select(), чтобы выбрать столбцы для просмотра и изменения отображаемого имени столбца.

В этом примере возвращаются строки, в которых ReturnValue не равен нулю, и выбирается для отображения столбцов TimeStart и ReturnValue с пользовательскими именами времени и ошибки.

dbgcmd

```
0:000> dx -r2 @$cursession.TTD.Calls("kernelbase!GetLastError").Where(c =>
c.ReturnValue != 0).Select(c => new { Time = c.TimeStart, Error =
c.ReturnValue })
@$cursession.TTD.Calls("kernelbase!GetLastError").Where(c => c.ReturnValue
!= 0).Select(c => new { Time = c.TimeStart, Error = c.ReturnValue })
    [0x1]
        Time             : 72D:1B3 [Time Travel]
        Error            : 0x57 [Type: unsigned long]
    [0x2]
        Time             : 72D:1FC [Time Travel]
        Error            : 0x2af9 [Type: unsigned long]
    [0x3]
        Time             : 72D:26E [Time Travel]
        Error            : 0x2af9 [Type: unsigned long]
```

# Группировка

Используйте метод GroupBy() для группировки данных, возвращаемых запросом для выполнения анализа с помощью структурированных результатов, в этом примере группируются расположения времени по номеру ошибки.

dbgcmd

```
0:000> dx -r2 @$cursession.TTD.Calls("kernelbase!GetLastError").Where(c =>
c.ReturnValue != 0).Select(c => new { Time = c.TimeStart, Error =
c.ReturnValue }).GroupBy(x => x.Error)
@$s = @$cursession.TTD.Calls("kernelbase!GetLastError").Where(c =>
c.ReturnValue != 0).Select(c => new { Time = c.TimeStart, Error =
c.ReturnValue }).GroupBy(x => x.Error)
    [0x36b7]
        [0x0]
        [0x1]
        [0x2]
        [0x3]
        [...]
    [0x3f0]
        [0x0]
```

```
        [0x1]
        [0x2]
        [0x3]
...
```

# Назначение результата запроса переменной

Используйте этот синтаксис для назначения результата запроса переменной `dx`

`@$var = <expression>`

В этом примере результаты запроса назначаются myResults

dbgcmd

```
dx -r2 @$myResults =
@$cursession.TTD.Calls("kernelbase!GetLastError").Where(c => c.ReturnValue
!= 0).Select(c => new { Time = c.TimeStart, Error = c.ReturnValue })
```

Используйте команду DX, чтобы отобразить только что созданную переменную с помощью параметра сетки -g. Дополнительные сведения о параметрах команды dx см. в разделе dx (Выражение объектной модели отладчика).

dbgcmd

```
0:000> dx -g @$myResults
=======================================
=           = (+) Time     = (+) Error =
=======================================
= [0x13]    - 3C64A:834    - 0x36b7    =
= [0x1c]    - 3B3E7:D6     - 0x3f0     =
= [0x1d]    - 3C666:857    - 0x36b7    =
= [0x20]    - 3C67E:12D    - 0x2       =
= [0x21]    - 3C6F1:127    - 0x2       =
= [0x23]    - 3A547:D6     - 0x3f0     =
= [0x24]    - 3A59B:D0     - 0x3f0     =
```

# Примеры

## Запрос исключений

Этот запрос LINQ использует TTD. Объект события для отображения всех исключений в трассировке.

dbgcmd

```
0:000> dx @$curprocess.TTD.Events.Where(t => t.Type == "Exception").Select(e
=> e.Exception)
@$curprocess.TTD.Events.Where(t => t.Type == "Exception").Select(e =>
e.Exception)
    [0x0]            : Exception 0x000006BA of type Software at PC:
0X777F51D0
    [0x1]            : Exception 0x000006BA of type Software at PC:
0X777F51D0
    [0x2]            : Exception 0xE06D7363 of type CPlusPlus at PC:
0X777F51D0
```

## Запрос для определенных вызовов API

Используйте TTD. Вызывает объект для запроса конкретных вызовов API. В этом примере произошла ошибка при вызове *пользователя 32! MessageBoxW*, API Windows для отображения поля сообщения. Мы перечисляем все вызовы MessageBoxW, упорядочиваем его по времени начала функции, а затем выберите последний вызов.

dbgcmd

```
0:000> dx @$cursession.TTD.Calls("user32!MessageBoxW").OrderBy(c =>
c.TimeStart).Last()
@$cursession.TTD.Calls("user32!MessageBoxW").OrderBy(c =>
c.TimeStart).Last()
    EventType        : Call
    ThreadId         : 0x3a10
    UniqueThreadId   : 0x2
    TimeStart        : 458310:539 [Time Travel]
    TimeEnd          : 45C648:61 [Time Travel]
    Function         : UnknownOrMissingSymbols
    FunctionAddress  : 0x750823a0
    ReturnAddress    : 0x40cb93
    ReturnValue      : 0x10a7000000000001
    Parameters
```

## Запрос события загрузки определенного модуля

Во-первых, используйте команду lm (список загруженных модулей) для отображения загруженных модулей.

dbgcmd

```
0:000> lm
start    end        module name
```

```
012b0000 012cf000   CDog_Console   (deferred)
11570000 1158c000   VCRUNTIME140D   (deferred)
11860000 119d1000   ucrtbased   (deferred)
119e0000 11b63000   TTDRecordCPU   (deferred)
11b70000 11cb1000   TTDWriter   (deferred)
73770000 73803000   apphelp    (deferred)
73ea0000 74062000   KERNELBASE   (deferred)
75900000 759d0000   KERNEL32   (deferred)
77070000 771fe000   ntdll      (private pdb symbols)
```

Затем используйте следующую команду dx, чтобы узнать, какую позицию в трассировке загружен определенный модуль, например ntdll.

dbgcmd

```
dx @$curprocess.TTD.Events.Where(t => t.Type == "ModuleLoaded").Where(t =>
t.Module.Name.Contains("ntdll.dll"))
@$curprocess.TTD.Events.Where(t => t.Type == "ModuleLoaded").Where(t =>
t.Module.Name.Contains("ntdll.dll"))
    [0x0]            : Module Loaded at position: A:0
```

Этот запрос LINQ отображает события загрузки определенного модуля.

dbgcmd

```
0:000> dx @$curprocess.TTD.Events.Where(t => t.Type ==
"ModuleUnloaded").Where(t => t.Module.Name.Contains("ntdll.dll"))
@$curprocess.TTD.Events.Where(t => t.Type == "ModuleUnloaded").Where(t =>
t.Module.Name.Contains("ntdll.dll"))
    [0x0]            : Module Unloaded at position: FFFFFFFFFFFFFFFE:0
```

Адрес FFFFFFFFFFFFFFFFE:0 указывает конец трассировки.

## Запрос всех проверок ошибок в трассировке

Используйте эту команду для сортировки по всем проверкам ошибок в трассировке по количеству ошибок.

dbgcmd

```
0:000> dx -g @$cursession.TTD.Calls("kernelbase!GetLastError").Where( x=>
x.ReturnValue != 0).GroupBy(x => x.ReturnValue).Select(x => new {
ErrorNumber = x.First().ReturnValue, ErrorCount =
x.Count()}).OrderByDescending(p => p.ErrorCount),d
================================================
=                 = (+) ErrorNumber = ErrorCount =
================================================
= [1008]          - 1008            - 8668       =
```

```
= [14007]          - 14007           - 4304       =
= [2]              - 2               - 1710       =
= [6]              - 6               - 1151       =
= [1400]           - 1400            - 385        =
= [87]             - 87              - 383        =
```

## Запрос позиции времени в трассировке при создании потоков

Используйте эту команду dx, чтобы отобразить все события трассировки в формате сетки (-g).

dbgcmd

```
0:000> dx -g @$curprocess.TTD.Events
================================================================================
================================================================================
========================================
=                                                          = (+) Type
= (+) Position          = (+) Module
= (+) Thread             =
================================================================================
================================================================================
========================================
= [0x0] : Module Loaded at position: 2:0                   - ModuleLoaded
- 2:0                   - Module C:\Users\USER1\Documents\Visual Studio
2015\Proje... -                         =
= [0x1] : Module Loaded at position: 3:0                   - ModuleLoaded
- 3:0                   - Module C:\WINDOWS\SYSTEM32\VCRUNTIME140D.dll at
address 0... -                          =
= [0x2] : Module Loaded at position: 4:0                   - ModuleLoaded
- 4:0                   - Module C:\WINDOWS\SYSTEM32\ucrtbased.dll at
address 0X118... -                        =
= [0x3] : Module Loaded at position: 5:0                   - ModuleLoaded
- 5:0                   - Module
C:\Users\USER1\AppData\Local\Dbg\UI\Fast.20170907... -
=
= [0x4] : Module Loaded at position: 6:0                   - ModuleLoaded
- 6:0                   - Module
C:\Users\USER1\AppData\Local\Dbg\UI\Fast.20170907... -
=
= [0x5] : Module Loaded at position: 7:0                   - ModuleLoaded
- 7:0                   - Module C:\WINDOWS\SYSTEM32\apphelp.dll at address
0X73770... -                          =
= [0x6] : Module Loaded at position: 8:0                   - ModuleLoaded
- 8:0                   - Module C:\WINDOWS\System32\KERNELBASE.dll at
address 0X73... -                       =
= [0x7] : Module Loaded at position: 9:0                   - ModuleLoaded
- 9:0                   - Module C:\WINDOWS\System32\KERNEL32.DLL at address
0X7590... -                         =
= [0x8] : Module Loaded at position: A:0                   - ModuleLoaded
```

```
- A:0                   - Module C:\WINDOWS\SYSTEM32\ntdll.dll at address
0X7707000... -                          =
= [0x9] : Thread created at D:0                             - ThreadCreated
- D:0                   -
- UID: 2, TID: 0x4C2C    =
= [0xa] : Thread terminated at 64:0                         -
ThreadTerminated    - 64:0                  -
- UID: 2, TID: 0x4C2C    =
= [0xb] : Thread created at 69:0                            - ThreadCreated
- 69:0                  -
- UID: 3, TID: 0x4CFC    =
= [0xc] : Thread created at 6A:0                            - ThreadCreated
- 6A:0                  -
- UID: 4, TID: 0x27B0    =
= [0xd] : Thread terminated at 89:0                         -
ThreadTerminated    - 89:0                  -
- UID: 4, TID: 0x27B0    =
= [0xe] : Thread terminated at 8A:0                         -
ThreadTerminated    - 8A:0                  -
- UID: 3, TID: 0x4CFC    =
= [0xf] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0  - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module C:\Users\USER1\Documents\Visual Studio
2015\Proje... -                         =
= [0x10] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0 - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module
C:\Users\USER1\AppData\Local\Dbg\UI\Fast.20170907... -
=
= [0x11] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0 - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module C:\WINDOWS\SYSTEM32\VCRUNTIME140D.dll at
address 0... -                          =
= [0x12] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0 - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module
C:\Users\USER1\AppData\Local\Dbg\UI\Fast.20170907... -
=
= [0x13] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0 - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module C:\WINDOWS\SYSTEM32\ucrtbased.dll at
address 0X118... -                          =
= [0x14] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0 - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module C:\WINDOWS\SYSTEM32\apphelp.dll at address
0X73770... -                         =
= [0x15] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0 - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module C:\WINDOWS\System32\KERNELBASE.dll at
address 0X73... -                         =
= [0x16] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0 - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module C:\WINDOWS\System32\KERNEL32.DLL at address
0X7590... -                         =
= [0x17] : Module Unloaded at position: FFFFFFFFFFFFFFFE:0 - ModuleUnloaded
- FFFFFFFFFFFFFFFE:0    - Module C:\WINDOWS\SYSTEM32\ntdll.dll at address
0X7707000... -                          =
================================================================================
================================================================================
========================================
```

Выберите любой из столбцов с знаком +, чтобы отсортировать выходные данные.

Используйте этот запрос LINQ для отображения в формате сетки, положение времени в трассировке при создании потоков (Type == ThreadCreated).

dbgcmd

```
dx -g @$curprocess.TTD.Events.Where(t => t.Type == "ThreadCreated").Select(t
=> t.Thread)
================================================================================
==============================
=                             = (+) UniqueId = (+) Id    = (+) Lifetime
= (+) ActiveTime  =
================================================================================
==============================
= [0x0] : UID: 2, TID: 0x4C2C - 0x2          - 0x4c2c    - [0:0,
FFFFFFFFFFFFFFFE:0]    - [D:0, 64:0]     =
= [0x1] : UID: 3, TID: 0x4CFC - 0x3          - 0x4cfc    - [0:0, 8A:0]
- [69:0, 8A:0]    =
= [0x2] : UID: 4, TID: 0x27B0 - 0x4          - 0x27b0    - [0:0, 89:0]
- [6A:0, 89:0]    =
================================================================================
==============================
```

Используйте этот запрос LINQ для отображения в формате сетки позиций времени трассировки при завершении потоков (Type == ThreadTerminated).

dbgcmd

```
0:000> dx -g @$curprocess.TTD.Events.Where(t => t.Type ==
"ThreadTerminated").Select(t => t.Thread)
================================================================================
==============================
=                             = (+) UniqueId = (+) Id    = (+) Lifetime
= (+) ActiveTime  =
================================================================================
==============================
= [0x0] : UID: 2, TID: 0x4C2C - 0x2          - 0x4c2c    - [0:0,
FFFFFFFFFFFFFFFE:0]    - [D:0, 64:0]     =
= [0x1] : UID: 4, TID: 0x27B0 - 0x4          - 0x27b0    - [0:0, 89:0]
- [6A:0, 89:0]    =
= [0x2] : UID: 3, TID: 0x4CFC - 0x3          - 0x4cfc    - [0:0, 8A:0]
- [69:0, 8A:0]    =
================================================================================
==============================
```

## Сортировка выходных данных для определения наиболее длительных потоков

Используйте этот запрос LINQ для отображения в формате сетки, приблизительного длительного выполнения потоков в трассировке.

dbgcmd

```
0:000> dx -g @$curprocess.TTD.Events.Where(e => e.Type ==
"ThreadTerminated").Select(e => new { Thread = e.Thread, ActiveTimeLength =
e.Thread.ActiveTime.MaxPosition.Sequence -
e.Thread.ActiveTime.MinPosition.Sequence }).OrderByDescending(t =>
t.ActiveTimeLength)
=========================================================
=          = (+) Thread              = ActiveTimeLength =
=========================================================
= [0x0]    - UID: 2, TID: 0x1750     - 0x364030         =
= [0x1]    - UID: 3, TID: 0x420C     - 0x360fd4         =
= [0x2]    - UID: 7, TID: 0x352C     - 0x35da46         =
= [0x3]    - UID: 9, TID: 0x39F4     - 0x34a5b5         =
= [0x4]    - UID: 11, TID: 0x4288    - 0x326199         =
= [0x5]    - UID: 13, TID: 0x21C8    - 0x2fa8d8         =
= [0x6]    - UID: 14, TID: 0x2188    - 0x2a03e3         =
= [0x7]    - UID: 15, TID: 0x40E8    - 0x29e7d0         =
= [0x8]    - UID: 16, TID: 0x124     - 0x299677         =
= [0x9]    - UID: 4, TID: 0x2D74     - 0x250f43         =
= [0xa]    - UID: 5, TID: 0x2DC8     - 0x24f921         =
= [0xb]    - UID: 6, TID: 0x3B1C     - 0x24ec8e         =
= [0xc]    - UID: 10, TID: 0x3808    - 0xf916f          =
= [0xd]    - UID: 12, TID: 0x26B8    - 0x1ed3a          =
= [0xe]    - UID: 17, TID: 0x37D8    - 0xc65            =
= [0xf]    - UID: 8, TID: 0x45F8     - 0x1a2            =
=========================================================
```

## Запрос доступа на чтение к диапазону памяти

Используйте TTD. Объект памяти для запроса для запроса доступа на чтение к диапазону памяти.

Блок среды потока (TEB) — это структура, содержащая все сведения о состоянии потока, включая результат, возвращенный GetLastError(). Эту структуру данных можно запросить, выполнив для `dx @$teb` текущего потока. Одним из членов TEB является переменная LastErrorValue, размером 4 байта. Мы можем ссылаться на член LastErrorValue в TEB с помощью этого синтаксиса. `dx &@$teb->LastErrorValue`.

В примере запроса показано, как найти каждую операцию чтения в этом диапазоне в памяти, выбрать все операции чтения, которые происходят до создания диалогового окна, а затем сортировать результат, чтобы найти последнюю операцию чтения.

dbgcmd

```
0:000> dx @$cursession.TTD.Memory(&@$teb->LastErrorValue, &@$teb-
>LastErrorValue + 0x4, "r")
@$cursession.TTD.Memory(&@$teb->LastErrorValue, &@$teb->LastErrorValue +
0x4, "r")
    [0x0]
    [0x1]
    [0x2]
    [0x3]
```

Если в нашей трассировке произошло событие диалогового окна, можно выполнить запрос, чтобы найти каждую операцию чтения, выполненную в этом диапазоне в памяти, выберите все операции чтения, которые происходят до создания диалогового окна, а затем сортировать результат, чтобы найти последнюю операцию чтения. Затем время перемещается к этой точке времени путем вызова SeekTo() в результирующем положении времени.

dbgcmd

```
:000> dx @$cursession.TTD.Memory(&@$teb->LastErrorValue, &@$teb-
>LastErrorValue + 0x4, "r").Where(m => m.TimeStart < @$dialog).OrderBy(m =>
m.TimeStart).Last().TimeEnd.SeekTo()
Setting position: 458300:37
ModLoad: 6cee0000 6cf5b000   C:\WINDOWS\system32\uxtheme.dll
ModLoad: 75250000 752e6000   C:\WINDOWS\System32\OLEAUT32.dll
ModLoad: 76320000 7645d000   C:\WINDOWS\System32\MSCTF.dll
ModLoad: 76cc0000 76cce000   C:\WINDOWS\System32\MSASN1.dll
```

# GitHub TTD Query Lab

Руководство по отладке кода C++ с помощью записи отладки временных путешествий с помощью запросов для поиска сведений о выполнении проблемного кода см. в статье WinDbg-Samples — отладка и запросы во времени.

Весь код, используемый в лаборатории, доступен здесь: https://github.com/Microsoft/WinDbg-Samples/tree/master/TTDQueries/app-sample

# Устранение неполадок запросов TTD

## "UnknownOrMissingSymbols" в качестве имен функций

Для предоставления имен функций, значений параметров и т. д. для расширения модели данных требуются полные сведения о символах. Если полные сведения о символах недоступны, отладчик использует "UnknownOrMissingSymbols" в качестве имени функции.

- Если у вас есть закрытые символы, вы получите имя функции и правильный список параметров.
- Если у вас есть открытые символы, вы получите имя функции и набор параметров по умолчанию — четыре неподписанных 64-разрядных ints.
- Если у вас нет сведений о символах для запрашиваемого модуля, в качестве имени используется "UnknownOrMissingSymbols".

## Запросы TTD для вызовов

Может быть несколько причин, по которым запрос не возвращает никаких вызовов библиотеки DLL.

- Синтаксис вызова не совсем правильный. Попробуйте проверить синтаксис вызова с помощью x (проверить символы). `x <call>` Если имя модуля, возвращаемое x, находится в верхнем регистре, используйте это.
- Библиотека DLL еще не загружена и загружается позже в трассировке. Чтобы обойти это путешествие до точки во времени после загрузки библиотеки DLL и повторного выполнения запроса.
- Вызов встраиваются, которые подсистема запросов не может отслеживать.
- Шаблон запроса использует подстановочные знаки, которые возвращают слишком много функций. Попробуйте сделать шаблон запроса более конкретным, чтобы количество сопоставленных функций было достаточно небольшим.

# См. также

Использование LINQ С объектами отладчика

dx (выражение объектной модели отладчика)

Отладка временных путешествий — обзор

Отладка перемещения по времени — автоматизация JavaScript

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Объекты вызовов TTD

18.07.2025

## Description

*TTD Calls* objects are used to give information about function calls that occur over the course of a trace.

## Parameters

Развернуть таблицу

| Property | Description |
|---|---|
| Function!SymbolName | Один или несколько, содержащихся в двойных кавычках, разделенных запятой. Например, dx @$cursession. TTD. Вызовы("module!symbol1", "module!symbol2", ...) |

## Properties

Развернуть таблицу

| Property | Description |
|---|---|
| EventType | Тип события. Это "Вызов" для всех объектов вызовов TTD. |
| ThreadId | Идентификатор потока ОС, который сделал запрос. |
| UniqueThreadId | Уникальный идентификатор потока в трассировке. Обычные идентификаторы потоков могут повторно использоваться в течение всего времени существования процесса, но UniqueThreadIds не может. |
| Function | Символическое имя функции. |
| FunctionAddress | Адрес функции в памяти. |
| ReturnValue | Возвращаемое значение функции. Если функция имеет тип void, это свойство не будет присутствовать. |

## Children

Развернуть таблицу

| Object | Description |
|---|---|
| Parameters[] | Массив, содержащий параметры, переданные функции. Количество элементов зависит от сигнатуры типа функции. |
| TimeStart | A position object that describes the position at the start of the call. |
| TimeEnd | A position object that describes the position at the end of the call. |

# Remarks

Отладка перемещения по времени использует сведения о символах, предоставленные в PDF-файлах, для определения количества параметров функции и их типов, возвращаемого типа значения и соглашения о вызовах. В случае, если сведения о символах недоступны или символы были ограничены общедоступными сведениями о символах, по-прежнему можно выполнять запросы. Подсистема запросов по пути во времени будет делать некоторые предположения в этом сценарии:

- Существует четыре 64-разрядных целочисленных параметра без знака для функции.
- Возвращаемое значение — 64-разрядное целое число без знака
- Для имени функции задана фиксированная строка : "UnknownOrMissingSymbols"

Эти предположения позволяют выполнять запросы в отсутствие надлежащих сведений о символах. Однако для получения наилучших результатов по возможности используются полные символы PDB.

Обратите внимание, что функция "Вызовы" выполняет вычисления и в зависимости от размера трассировки может занять некоторое время. Загрузка ЦП будет резко возрастать во время вычислений и наблюдения за использованием ЦП в диспетчере задач, дает указание на то, что вычисления выполняются. Результаты запроса кэшируются в памяти, поэтому последующие запросы к ранее запрашиваемые вызовы значительно быстрее.

# Example Usage

В этом примере показан объект вызовов для ucrtbase!initterm.

dbgcmd

```
0:000> dx -r2 @$cursession.TTD.Calls("ucrtbase!initterm")
@$cursession.TTD.Calls("ucrtbase!initterm")
    [0x0]
        EventType        : Call
```

```
        ThreadId         : 0x2074
        UniqueThreadId   : 0x2
        TimeStart        : 1E:5D0
        TimeEnd          : 2D:E
        Function         : ucrtbase!_initterm
        FunctionAddress  : 0x7ffb345825d0
        ReturnAddress    : 0x7ff6a521677e
        Parameters
        SystemTimeStart  : Friday, January 12, 2024 21:18:40.799
        SystemTimeEnd    : Friday, January 12, 2024 21:18:44.26
```

## See Also

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

dx (выражение объектной модели отладчика)

# Объекты кучи TTD

Статья • 22.11.2024

## Description

*Объекты кучи TTD используются для предоставления сведений о вызовах кучи* , происходящих в ходе трассировки.

## Свойства

Каждый объект кучи будет иметь эти свойства.

Развернуть таблицу

| Свойство | Description |
|---|---|
| Действие | Описывает действие, которое произошло. Возможные значения: Alloc, ReAlloc, Free, Create, Protect, Lock, Unlock, Destroy. |
| Куча | Дескриптор кучи Win32. |

### Условные свойства

В зависимости от объекта кучи он может иметь некоторые свойства ниже.

Развернуть таблицу

| Свойство | Description |
|---|---|
| Адрес | Адрес выделенного объекта. |
| PreviousAddress | Адрес выделенного объекта, прежде чем он был перераспределирован. Если адрес не совпадает с предыдущимaddress, то перемещение привело к перемещению памяти. |
| Размер | Размер и/или запрошенный размер выделенного объекта. |
| BaseAddress | Адрес выделенного объекта в куче. Он может представлять адрес, который будет освобожден (Бесплатный) или адрес объекта до его перераспределенности (ReAlloc).) |
| Флаги | Значение зависит от API. |

| Свойство | Description |
| --- | --- |
| Результат | Результат вызова API кучи. Ненулевая означает успешность и ноль означает неудачу. |
| ReserveSize | Объем памяти для резервирования кучи. |
| Фиксация | Начальный размер фиксации для кучи. |
| MakeReadOnly | Значение, отличное от нуля, указывает запрос, чтобы сделать кучу доступной только для чтения; Нулевое значение указывает, что куча должна быть чтением и записью. |

# Children

Развернуть таблицу

| Объект | Description |
| --- | --- |
| TimeStart | Объект положения, описывающий позицию в начале выделения. |
| TimeEnd | Объект положения, описывающий позицию в конце выделения. |

# Пример использования

Используйте эту команду dx (выражение объектной модели отладчика) для отображения памяти кучи в сетке с помощью параметра -g.

dbgcmd

```
0:0:000> dx -g @$cursession.TTD.Data.Heap()
================================================================================
================================================================================
=                          = Action     = Heap          = Address       =
Size      = Flags  = (+) TimeStart = (+) TimeEnd = Result = PreviousAddress
=
================================================================================
================================================================================
= [0x0] : [object Object]  - Alloc      - 0xaf0000      - 0xb0cfd0      -
0x4c      - 0x0    - FAB:17B1      - FAD:40      -        -
=
= [0x1] : [object Object]  - Alloc      - 0xaf0000      - 0xb07210      -
0x34      - 0x8    - FB1:9         - FB3:74      -        -
=
= [0x2] : [object Object]  - Alloc      - 0xaf0000      - 0xb256d8      -
0x3c      - 0x8    - E525:174      - E526:E1     -        -
=
```

Выходные данные можно описать как "нормализованные данные", так как существует выбранный набор API, представляющих операции кучи. Данные, извлеченные из соответствующих параметров, представлены в единообразном режиме.

Щелкнув TimeStart или TimeEnd, вы перейдете к этой точке трассировки.

Щелкните поле параметров рядом с определенной записью, чтобы отобразить доступные сведения о параметрах.

dbgcmd

```
dx -r1 @$cursession.TTD.Data.Heap()[2].@"Parameters"
@$cursession.TTD.Data.Heap()[2].@"Parameters"
    [0x0]            : 0x16c7d780000
    [0x1]            : 0x280000
    [0x2]            : 0x20
    [0x3]            : 0x0
...
```

Увеличьте значение рекурсии -r, чтобы отобразить дополнительные сведения.

## См. также

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

dx (выражение объектной модели отладчика)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Объекты памяти TTD

18.07.2025

## Описание

*Память TTD* — это метод, который принимает beginAddress, endAddress и dataAccessMask и возвращает коллекцию объектов памяти, содержащих сведения о доступе к памяти.

## Параметры

Развернуть таблицу

| Недвижимость | Описание |
|---|---|
| начальный адрес | Начальный адрес объекта памяти начинается с 0x. |
| endAddress | Конечный адрес объекта памяти, который начинается с 0x. |
| маска доступа к данным | Маска доступа к данным, содержащаяся в двойных кавычках. Это может быть r для чтения, w для записи, e для выполнения и c для изменения. |

## Дети

Развернуть таблицу

| Объект | Описание |
|---|---|
| Тип события | Тип события. Это "MemoryAccess" для всех объектов TTD.Memory. |
| ThreadId | Идентификатор потока ОС, который сделал запрос. |
| UniqueThreadId | Уникальный идентификатор потока в трассировке. Обычные идентификаторы потоков могут повторно использоваться в течение всего времени существования процесса, но UniqueThreadIds не может. |
| TimeStart | Объект положения, описывающий позицию при создании доступа к памяти. |
| TimeEnd | Объект положения, описывающий позицию при создании доступа к памяти. Это всегда будет совпадать со временем запуска объектов TTD.Memory. |
| Тип доступа | Тип доступа — чтение, запись или выполнение. |
| IP-адрес | Указатель кода инструкции, который инициировал доступ к памяти. |

| Объект | Описание |
| --- | --- |
| Адрес | Адрес, который будет прочитан, записан или выполнен, будет находиться в диапазоне [beginAddress, endAddress) и взят из параметров для .Memory(). Обратите внимание, что интервал открыт наполовину. То есть ни один из возвращаемых событий не будет иметь адрес, соответствующий endAddress, но могут быть события, соответствующие endAddress – 1. |
| Размер | Размер операции чтения и записи и выполнения в байтах. Обычно это будет 8 байтов или меньше. В случае выполнения кода это число байтов в инструкции, которая была выполнена. |
| Ценность | Значение, которое было прочитано, записано или выполнено. В случае выполнения он содержит байты кода для инструкции. Обратите внимание, что байты инструкций перечислены в порядке MSB по дизассемблеру, но будут храниться в значении в порядке LSB. |

# Замечания

В TTD разрешены следующие типы доступа. Запросы памяти:

- r — чтение
- w — запись
- rw — чтение и запись
- e — выполнение
- rwe — чтение / запись / выполнение
- ec — выполнить/изменить

Обратите внимание, что это функция, которая выполняет вычисления, поэтому требуется некоторое время для выполнения.

# Пример использования

В этом примере показана сетка всех позиций в трассировке, где произошел доступ на чтение к четырем байтам памяти, начиная с адреса 0x00a4fca0. Щелкните любую запись, чтобы провести подробный анализ каждого случая доступа к памяти.

dbgcmd

```
dx -g @$cursession.TTD.Memory(0x00a4fca0,0x00a4fca4, "r")
```

| | (+) EventType | (+) ThreadId | (+) UniqueThreadId | (+) TimeStart | (+) TimeEnd | (+) AccessType | (+) IP | (+) Address | (+) Size | (+) Value |
|---|---|---|---|---|---|---|---|---|---|---|
| [0x0] | MemoryAccess | 0x3168 | 0x2 | 15:16 | 15:16 | Read | 0x77a02196 | 0xa4fca0 | 0x4 | 0xa4fcf8 |
| [0x1] | MemoryAccess | 0x3168 | 0x2 | 15:18 | 15:18 | Read | 0x77a02198 | 0xa4fca0 | 0x4 | 0x77a26768 |
| [0x2] | MemoryAccess | 0x3168 | 0x2 | 15:36 | 15:36 | Read | 0x779c4305 | 0xa4fca0 | 0x4 | 0x779e0c21 |
| [0x3] | MemoryAccess | 0x3168 | 0x2 | 1E:26 | 1E:26 | Read | 0x779e0d07 | 0xa4fca0 | 0x4 | 0xa4fcf8 |
| [0x4] | MemoryAccess | 0x3168 | 0x2 | 2B:183 | 2B:183 | Read | 0x690b9c21 | 0xa4fca0 | 0x4 | 0x0 |
| [0x5] | MemoryAccess | 0x3168 | 0x2 | 3F:6 | 3F:6 | Read | 0x7562500a | 0xa4fca0 | 0x4 | 0x1f000 |
| [0x6] | MemoryAccess | 0x3168 | 0x2 | 43:3 | 43:3 | Read | 0x779dfa9b | 0xa4fca0 | 0x4 | 0x117a9480 |
| [0x7] | MemoryAccess | 0x3168 | 0x2 | 45:4 | 45:4 | Read | 0x779bc75b | 0xa4fca0 | 0x4 | 0x799123f0 |
| [0x8] | MemoryAccess | 0x3168 | 0x2 | 47:3 | 47:3 | Read | 0x779df73b | 0xa4fca0 | 0x4 | 0x117a9480 |
| [0x9] | MemoryAccess | 0x3168 | 0x2 | 47:A | 47:A | Read | 0x779df74b | 0xa4fca0 | 0x4 | 0x117a9480 |
| [0xa] | MemoryAccess | 0x3168 | 0x2 | 4B:2F | 4B:2F | Read | 0xb0a688 | 0xa4fca0 | 0x4 | 0xa4fd00 |
| [0xb] | MemoryAccess | 0x3168 | 0x2 | 4B:118 | 4B:118 | Read | 0xb02e96 | 0xa4fca0 | 0x4 | 0x0 |
| [0xc] | MemoryAccess | 0x3168 | 0x2 | 4B:11C | 4B:11C | Read | 0xb02ea1 | 0xa4fca0 | 0x4 | 0x0 |
| [0xd] | MemoryAccess | 0x3168 | 0x2 | 4B:928 | 4B:928 | Read | 0x690c2bec | 0xa4fca0 | 0x4 | 0x69191a60 |
| [0xe] | MemoryAccess | 0x3168 | 0x2 | 4B:939 | 4B:939 | Read | 0x690ca901 | 0xa4fca0 | 0x4 | 0x69108d19 |
| [0xf] | MemoryAccess | 0x3168 | 0x2 | 4B:941 | 4B:941 | Read | 0x779c3418 | 0xa4fca0 | 0x4 | 0x69191a80 |
| [0x10] | MemoryAccess | 0x3168 | 0x2 | 5D:113 | 5D:113 | Read | 0x690795e5 | 0xa4fca0 | 0x4 | 0x0 |
| [0x11] | MemoryAccess | 0x3168 | 0x2 | 5F:18 | 5F:18 | Read | 0x690795f0 | 0xa4fca0 | 0x4 | 0x0 |
| [0x12] | MemoryAccess | 0x3168 | 0x2 | 64:6D | 64:6D | Read | 0x690ca8fc | 0xa4fca0 | 0x4 | 0x69191a60 |
| [0x13] | MemoryAccess | 0x3168 | 0x2 | 7A:CC | 7A:CC | Read | 0x77a340d9 | 0xa4fca0 | 0x4 | 0xf0 |
| [0x14] | MemoryAccess | 0x3168 | 0x2 | B9:CC | B9:CC | Read | 0x77a340d9 | 0xa4fca0 | 0x4 | 0xf0 |

Вы можете щелкнуть поля TimeStart в любом из событий в сетке, чтобы отобразить сведения об этом событии.

dbgcmd

```
0:000> dx -r1 @$cursession.TTD.Memory(0x00a4fca0,0x00a4fca4, "r")[16].TimeStart
@$cursession.TTD.Memory(0x00a4fca0,0x00a4fca4, "r")[16].TimeStart
: 5D:113 [Time Travel]
    Sequence         : 0x5d
    Steps            : 0x113
```

Чтобы перейти к позиции в трассировке, в которой произошло событие, щелкните [Time Travel].

dbgcmd

```
0:000> dx @$cursession.TTD.Memory(0x00a4fca0,0x00a4fca4, "r")
[16].TimeStart.SeekTo()
@$cursession.TTD.Memory(0x00a4fca0,0x00a4fca4, "r")[16].TimeStart.SeekTo()
(27b8.3168): Break instruction exception - code 80000003 (first/second chance not
available)
Time Travel Position: 5D:113

eax=0000004c ebx=00dd0000 ecx=00a4f89c edx=00a4f85c esi=00a4f89c edi=00b61046
eip=690795e5 esp=00a4f808 ebp=00a4f818 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206
690795e5 ffb604040000    push    dword ptr [esi+404h] ds:002b:00a4fca0=00000000
```

В этом примере перечислены все позиции трассировки, где были выполнены операции чтения и записи для четырёх байтов памяти, начиная с 0x1bf7d0. Щелкните любую запись, чтобы подробно изучить каждое использование доступа к памяти.

dbgcmd

```
0:000> dx @$cursession.TTD.Memory(0x1bf7d0,0x1bf7d4, "rw")
@$cursession.TTD.Memory(0x1bf7d0,0x1bf7d4, "rw")
    [0x0]
```

```
    [0x1]
    [0x2]
    [0x3]
     ...
```

В этом примере указаны все позиции в трассировке, где были выполнены или изменены четыре байта памяти, начиная с адреса 0x13a1710. Щелкните на любой случай, чтобы получить дополнительные сведения о каждом случае доступа к памяти.

dbgcmd

```
0:000> dx -r1 @$cursession.TTD.Memory(0x13a1710,0x13a1714, "ec")[0]
@$cursession.TTD.Memory(0x13a1710,0x13a1714, "ec")[0]
    EventType        : MemoryAccess
    ThreadId         : 0x1278
    UniqueThreadId   : 0x2
    TimeStart        : 5B:4D [Time Travel]
    TimeEnd          : 5B:4D [Time Travel]
    AccessType       : Execute
    IP               : 0x13a1710
    Address          : 0x13a1710
    Size             : 0x1
    Value            : 0x55
```

## См. также

Отладка с путешествием во времени — введение в объекты этой отладки

Отладка временных путешествий — обзор

# Объекты событий TTD

18.07.2025

## Description

*TTD Event* objects are used to give information about important events that happened during a time travel trace.

## Properties

Развернуть таблицу

| Property | Description |
|---|---|
| Type | Описывает тип события, которое произошло. Возможные значения: ThreadCreated, ThreadTerminated, ModuleLoaded, ModuleUnloaded, Exception |

## Children

Развернуть таблицу

| Object | Description |
|---|---|
| Position | A position object that describes the position the event occurred. |
| Module* | A module object containing information about the module that was loaded or unloaded. |
| Thread* | A thread object containing information about the thread that was created or terminated. |
| Exception* | An exception object containing information about the exception that was hit. |

* — существование этих дочерних объектов зависит от типа события

## Example Usage

dbgcmd

```
0:000> dx -r2 @$curprocess.TTD.Events.Where(t => t.Type == "Exception").Select(e
=> e.Exception)
@$curprocess.TTD.Events.Where(t => t.Type == "Exception").Select(e => e.Exception)
    [0x0]            : Exception of type CPlusPlus at PC: 0X777663B0
        Position         : 13B7:0 [Time Travel]
```

```
        Type             : CPlusPlus
        ProgramCounter   : 0x777663b0
        Code             : 0xe06d7363
        Flags            : 0x1
        RecordAddress    : 0x0
    [0x1]            : Exception of type Hardware at PC: 0XF1260D0
        Position         : BC0F:0 [Time Travel]
        Type             : Hardware
        ProgramCounter   : 0xf1260d0
        Code             : 0x80000003
        Flags            : 0x0
        RecordAddress    : 0x0
```

# See Also

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

dx (выражение объектной модели отладчика)

# Объекты исключений TTD

18.07.2025

## Description

*TTD Exception* objects are used to provide information about event exceptions that happened during a trace session.

## Properties

Развернуть таблицу

| Property | Description |
|---|---|
| Type | Описывает тип исключения. Возможные значения: Software и Hardware. |
| ProgramCounter | Инструкция, в которой было создано исключение. |
| Code | Код исключения. |
| Flags | Флаги исключения. |
| RecordAddress | Где в памяти можно найти запись исключения. |

## Children

Развернуть таблицу

| Object | Description |
|---|---|
| Position | A position object that describes the position the exception occurred. |

## Example Usage

dbgcmd

```
0:003> dx -r1 @$curprocess.TTD.Events.Where(t => t.Type == "Exception")
[0].Exception
@$curprocess.TTD.Events.Where(t => t.Type == "Exception")[0].Exception
: Exception 0x80010012 of type Software at PC: 0X7FF9F6DC8670
    Position         : 36A7:0 [Time Travel]
    Type             : Software
```

```
    ProgramCounter   : 0x7ff9f6dc8670
    Code             : 0x80010012
    Flags            : 0x1
    RecordAddress    : 0x0
```

## See Also

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

dx (выражение объектной модели отладчика)

# Объекты позиции TTD

01.07.2025

## Описание

*Объекты позиций* используются для описания позиции в трассировке перемещения по времени. Объект позиции обычно описывается двумя шестнадцатеричными числами, разделенными двоеточием. Первым из шестнадцатеричных чисел является *последовательность* , а второй — *шаги*.

Позиция FFFFFFFFFFFFFFFE:0 указывает конец трассировки.

## Свойства

Развернуть таблицу

| Свойство | Описание |
|---|---|
| Процент | Процент отслеживания (до ближайшей последовательности).<br>Примечание. Точный процент может отличаться от того, что запрашивается в команде !tt из-за округления до ближайшей последовательности. |
| Последовательность | Точка последовательности, относящаяся к позиции. |
| Шаги | Количество шагов из точки последовательности в этом потоке, чтобы получить эту позицию. |

## Методы

Развернуть таблицу

| Метод | Описание |
|---|---|
| SeekTo() | Время перемещается в эту позицию трассировки. |
| ToSystemTime() | Возвращает приблизительное время часов стены для позиции (UTC). |

## Пример использования

```
dbgcmd
```

```
0:003> dx -r1 @$create("Debugger.Models.TTD.Position", 14006, 0)
@$create("Debugger.Models.TTD.Position", 14006, 0)                : 36B6:0 [Time
Travel]
    Sequence         : 0x36b6
    Steps            : 0x0
    SeekTo           [Method which seeks to time position]
    ToSystemTime     [Method which obtains the approximate system time at a given
position]
```

# См. также

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

dx (выражение объектной модели отладчика)

# Объекты диапазона TTD

Статья • 22.11.2024

## Description

*Объекты диапазона* TTD используются для предоставления сведений о диапазоне времени трассировки. Обычно они используются для описания времени существования объекта потока TTD во время сеанса TTD.

## Children

Развернуть таблицу

| Объект | Description |
|---|---|
| MinPosition | Объект положения, описывающий наиболее раннюю позицию, относящуюся к диапазону. |
| MaxPosition | Объект положения, описывающий последнюю позицию, относящуюся к диапазону. |

## Пример использования

В этом примере объекты MinPosition и MaxPosition отображаются для времени существования и ActiveTime, связанных с потоком.

dbgcmd

```
0:003> dx -r1 @$curprocess.TTD.Threads[5]
@$curprocess.TTD.Threads[5]                 : UID: 7, TID: 0x2580
    UniqueId         : 0x7
    Id               : 0x2580
    Lifetime         : [BAF:0, FFFFFFFFFFFFFFFE:0]
    ActiveTime       : [BB2:0, C6A:0]
    GatherMemoryUse  [Gather inputs, outputs and memory used by a range of
execution within a thread]

0:003> dx -r1 @$curprocess.TTD.Threads[5].Lifetime
@$curprocess.TTD.Threads[5].Lifetime                : [BAF:0,
FFFFFFFFFFFFFFFE:0]
    MinPosition      : BAF:0 [Time Travel]
    MaxPosition      : FFFFFFFFFFFFFFFE:0 [Time Travel]

0:003> dx -r1 @$curprocess.TTD.Threads[5].ActiveTime
```

```
@$curprocess.TTD.Threads[5].ActiveTime                 : [BB2:0, C6A:0]
    MinPosition      : BB2:0 [Time Travel]
    MaxPosition      : C6A:0 [Time Travel]
```

## См. также

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

dx (выражение объектной модели отладчика)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Объекты потоков TTD

Статья • 22.11.2024

## Description

*Объекты потока* TTD используются для предоставления сведений о потоках и их времени существования во время трассировки времени.

## Свойства

Развернуть таблицу

| Свойство | Description |
|---|---|
| UniqueId | Уникальный идентификатор потока в трассировке. |
| Артикул | TID потока. |

## Children

Развернуть таблицу

| Объект | Description |
|---|---|
| Продолжительность жизни | Объект диапазона TTD, описывающий время существования потока. |
| ActiveTime | Объект диапазона TTD, описывающий время активности потока. |

## Пример использования

Используйте эту команду dx, чтобы отобразить все потоки в массиве.

dbgcmd

```
0:0:000> dx -g @$curprocess.TTD.Threads
================================================================================
====================================
=                             = (+) UniqueId = (+) Id    = (+) Lifetime
= (+) ActiveTime        =
```

```
================================================================================
====================================
= [0x0] : UID: 2, TID: 0x2428 - 0x2          - 0x2428    - [0:0, 6F0C4:0]
- [50C63:0, 6F0C4:0]    =
= [0x1] : UID: 3, TID: 0x3520 - 0x3          - 0x3520    - [0:0,
FFFFFFFFFFFFFFFE:0]    - [5115A:0, 56B07:0]    =
= [0x2] : UID: 4, TID: 0x18E8 - 0x4          - 0x18e8    - [0:0,
FFFFFFFFFFFFFFFE:0]    - [52F65:0, 56B1E:0]    =
= [0x3] : UID: 5, TID: 0x5690 - 0x5          - 0x5690    - [0:0,
FFFFFFFFFFFFFFFE:0]    - [5300D:0, 5D4FA:0]    =
= [0x4] : UID: 6, TID: 0x46FC - 0x6          - 0x46fc    - [0:0,
FFFFFFFFFFFFFFFE:0]    - [53782:0, 5433B:0]    =
= [0x5] : UID: 7, TID: 0x58D0 - 0x7          - 0x58d0    - [0:0,
FFFFFFFFFFFFFFFE:0]    - [542FE:0, 543B9:0]    =
= [0x6] : UID: 8, TID: 0x950  - 0x8          - 0x950     - [0:0,
FFFFFFFFFFFFFFFE:0]    - [543C4:0, 544B8:0]    =
= [0x7] : UID: 9, TID: 0x4514 - 0x9          - 0x4514    - [0:0, 6D61B:0]
- [5DBBD:0, 6D61B:0]    =
================================================================================
====================================
```

Эта команда dx используется для отображения сведений о первом потоке в массиве.

dbgcmd

```
0:0:000 dx -r2 @$curprocess.TTD.Threads[0]
@$curprocess.TTD.Threads[0]                 : UID: 2, TID: 0x2428
    UniqueId         : 0x2
    Id               : 0x2428
    Lifetime         : [0:0, 6F0C4:0]
        MinPosition      : Min Position [Time Travel]
        MaxPosition      : 6F0C4:0 [Time Travel]
    ActiveTime       : [50C63:0, 6F0C4:0]
        MinPosition      : 50C63:0 [Time Travel]
        MaxPosition      : 6F0C4:0 [Time Travel]
```

Ссылки [Время перемещения] предоставляют ссылку на SeekTo() конкретную запись в трассировке при активном потоке.

dbgcmd

```
0:0:000> dx
@$curprocess.TTD.Threads[0].@"ActiveTime".@"MinPosition".SeekTo()
Setting position: 50C63:0
@$curprocess.TTD.Threads[0].@"ActiveTime".@"MinPosition".SeekTo()
(40b4.2428): Break instruction exception - code 80000003 (first/second
chance not available)
Time Travel Position: 50C63:0
```

```
ntdll!NtTestAlert+0x14:
00007ffb`e3e289d4 c3              ret
```

## См. также

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

dx (выражение объектной модели отладчика)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Объекты модуля TTD

18.07.2025

## Description

*TTD Module* objects are used to give information about modules that were loaded and unloaded during a trace session.

## Properties

Развернуть таблицу

| Property | Description |
|---|---|
| Name | Имя и путь модуля. |
| Address | Адрес, в котором был загружен модуль. |
| Size | Размер модуля в байтах. |
| Checksum | Контрольная сумма модуля. |
| Timestamp | Метка времени модуля. |

## Example Usage

dbgcmd

```
0:005> dx -r1 @$cursession.Processes[24848].Modules[10]
@$cursession.Processes[24848].Modules[10]                 : C:\Program
Files\WindowsApps\Microsoft.VCLibs.140.00_14.0.30704.0_x64__8wekyb3d8bbwe\VCRUNTIM
E140_APP.dll [Force Symbol Reload]
    BaseAddress      : 0x7ff9adae0000
    Name             : C:\Program
Files\WindowsApps\Microsoft.VCLibs.140.00_14.0.30704.0_x64__8wekyb3d8bbwe\VCRUNTIM
E140_APP.dll
    Size             : 0x1b000
    SymbolType       : Deferred
    Attributes
    Contents
    Symbols          : [SymbolModule]VCRUNTIME140_APP
```

## See Also

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

dx (выражение объектной модели отладчика)

# Объекты коллекции TTD

Статья • 22.11.2024

## Description

Объекты модели коллекции позволяют совместно манипулировать объектами модели отладчика, поддерживающими объекты коллекции.

## Children

Развернуть таблицу

| Объект | Description |
|---|---|
| MinPosition | Объект положения, описывающий наиболее раннюю позицию, относящуюся к диапазону. |

### Методы объекта коллекции TTD

**Содержит (OtherString)** -Method, который возвращает, содержит ли строка заданную подстроку.

**EndsWith(OtherString)** -Method, который возвращает, заканчивается ли строка заданной строкой.

**IndexOf(OtherString)** -Method, который возвращает индекс первого вхождения подстроки в данной строке. Если такое вхождение отсутствует, возвращается значение -1.

**Метод LastIndexOf(OtherString),** который возвращает индекс последнего вхождения подстроки в данной строке. Если такое вхождение отсутствует, возвращается значение -1.

**Длина** — свойство, возвращающее длину строки.

**PadLeft(TotalWidth)** — метод, который справа выравнивает строку по заданной ширине, вставляя пробелы слева от строки.

**PadRight(TotalWidth)** — метод, который слева выравнивает строку до указанной ширины, вставляя пробелы справа от строки.

**Remove(StartPos, [Length])** — метод, который удаляет все символы, начиная с указанной позиции из строки. Если указана необязательная длина, удаляются только многие символы после удаления начальной позиции.

**Replace(SearchString, ReplaceString)** — метод, который заменяет каждое вхождение указанной строки поиска на строку замены.

**StartsWith(OtherString)** — метод, который возвращает, начинается ли строка с заданной строкой.

**Подстрока(StartPos, [Длина])** — метод, который извлекает подстроку из заданной строки. Подстрока начинается с указанной позиции символа и продолжается до конца строки или для необязательно указанной длины.

**ToLower()** — возвращает копию этой строки, преобразованной в нижний регистр.

**ToUpper()** — возвращает копию этой строки, преобразованной в верхний регистр.

# Пример использования

Этот запрос LINQ отображает события загрузки определенного модуля, содержащего ntdll.dll.

dbgcmd

```
0:000> dx @$curprocess.TTD.Events.Where(t => t.Type ==
"ModuleUnloaded").Where(t => t.Module.Name.Contains("ntdll.dll"))
```

# См. также

Использование LINQ С объектами отладчика

dx (выражение объектной модели отладчика)

Отладка временных путешествий — введение в объекты отладки временных путешествий

Отладка временных путешествий — обзор

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Отладка перемещения по времени — автоматизация JavaScript

Статья • 19.11.2024



Автоматизация JavaScript позволяет работать с трассировками TTD несколькими способами, например автоматизацией команд или запросами для поиска данных событий из файла трассировки.

Общие сведения о работе с JavaScript см. в скриптах отладчика JavaScript. Также существуют примеры скриптов отладчика JavaScript.

## Автоматизация команд JavaScript TTD

Одним из способов использования JavaScript для автоматизации TTD является отправка команд для автоматизации работы с файлами трассировки времени.

### Перемещение в файл трассировки

В этом JavaScript показано, как перейти к началу трассировки перемещения по времени с помощью команды !tt .

JavaScript

```
var dbgControl = host.namespace.Debugger.Utility.Control;
dbgControl.ExecuteCommand("!tt 0",false);
host.diagnostics.debugLog(">>> Sent command to move to the start of the TTD
file \n");
```

Мы можем сделать это в функцию ResetTrace и сохранить ее как MyTraceUtils.js с помощью пользовательского интерфейса JavaScript в WinDbg.

JavaScript

```
// My Trace Utils
// WinDbg TTD JavaScript MyTraceUtilsCmd Sample

"use strict";
```

```
function MyTraceUtilsCmd()
{
    var dbgControl = host.namespace.Debugger.Utility.Control;
    dbgControl.ExecuteCommand("!tt 0",false);
    host.diagnostics.debugLog(">>> Sent command to move to the start of the
TTD file \n");
}
```

После загрузки файла TTD в WinDbg вызовите функцию ResetTraceCmd() с помощью команды DX в окне команды отладчика.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.ResetTraceCmd()
>>> Sent command to move to the start of the TTD file
Debugger.State.Scripts.MyTraceUtils.Contents.ResetTraceCmd()
```

# Ограничения отправки команд

Но для всех, кроме самых простых ситуаций, подход отправки команд имеет недостатки. Он зависит от использования текстовых выходных данных. И анализ выходных данных приводит к тому, что код является хрупким и трудным для поддержания. Лучший подход — использовать объекты TTD напрямую.

В следующем примере показано, как использовать объекты непосредственно для выполнения той же задачи с помощью объектов напрямую.

JavaScript

```
// My Trace Utils
// WinDbg TTD JavaScript ResetTrace Sample

"use strict";

function ResetTrace()
{
    host.currentProcess.TTD.SetPosition(0);
    host.diagnostics.debugLog(">>> Set position to the start of the TTD file
\n");
}
```

При выполнении этого кода показано, что мы можем перейти к началу файла трассировки.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.ResetTrace()
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: F:0
>>> Set position to the start of the TTD file
```

В этом примере функция ResetTraceEnd устанавливается в конец трассировки, а текущее и новое положение отображается с помощью объекта currentThread.TTD Position.

JavaScript

```
// WinDbg TTD JavaScript Sample to Reset Trace using objects directly
// and display current and new position

function ResetTraceEnd()
{
   var PositionOutputStart = host.currentThread.TTD.Position;
   host.diagnostics.debugLog(">>> Current position in trace file:  "+
PositionOutputStart +"\n");
   host.currentProcess.TTD.SetPosition(100);
   var PositionOutputNew = host.currentThread.TTD.Position;
   host.diagnostics.debugLog(">>> New position in trace file:  "+
PositionOutputNew +"\n");
}
```

При выполнении этого кода отображается текущая и новая позиция.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.ResetTraceEnd()
>>> Current position in trace file:  F:0
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: D3:1
>>> New position in trace file:  D3:1
```

В этом развернутом примере значения начальной и конечной позиции сравниваются, если позиция в трассировке изменилась.

JavaScript

```
// WinDbg TTD JavaScript ResetTraceEx Sample

"use strict";

function ResetTraceEx()
```

```
{
    const PositionOutputStart = host.currentThread.TTD.Position;
    host.diagnostics.debugLog(">>> Current position in trace file:  "+
PositionOutputStart +"\n");

    host.currentProcess.TTD.SetPosition(0);

    const PositionOutputNew = host.currentThread.TTD.Position;
    host.diagnostics.debugLog(">>> New position in trace file:  "+
PositionOutputNew +"\n");

    if (parseInt(PositionOutputStart,16) != parseInt(PositionOutputNew,16))
    {
        host.diagnostics.debugLog(">>> Set position to the start of the TTD
file  \n");
    }
    else
    {
        host.diagnostics.debugLog(">>> Position was already set to the start
of the TTD file \n");
    }
}
```

В этом примере отображается сообщение о том, что все готово в начале файла трассировки.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.ResetTraceEx()
>>> Current position in trace file:  F:0
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: F:0
>>> New position in trace file:  F:0
>>> Position was already set to the start of the TTD file
```

Чтобы протестировать скрипт, используйте команду !tt для перехода по полпути в файле трассировки.

dbgcmd

```
0:000> !tt 50
Setting position to 50% into the trace
Setting position: 71:0
```

Теперь при выполнении скрипта отображается соответствующее сообщение, указывающее, что позиция была задана в начале трассировки TTD.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.ResetTraceEx()
>>> Current position in trace file:  71:0
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: F:0
>>> New position in trace file:  F:0
>>> Set position to the start of the TTD file
```

# Индексирование файла трассировки перемещения по времени

Если только файл трассировки копируется на другой компьютер, он должен быть переиндексирован. Дополнительные сведения см. в статье "Отладка перемещения по времени" — работа с файлами трассировки.

В этом коде показан пример функции IndexTrace, которая показывает, сколько времени требуется для повторного индексирования файла трассировки.

JavaScript

```
function IndexTrace()
{
    var timeS = (new Date()).getTime();
    var output = host.currentProcess.TTD.Index.ForceBuildIndex();
    var timeE = (new Date()).getTime();
    host.diagnostics.debugLog("\n>>> Trace was indexed in " + (timeE -
timeS) + " ms\n");
}
```

Ниже приведены выходные данные из небольшого файла трассировки.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.IndexTrace()

>>> Trace was indexed in 2 ms
```

## Добавление инструкции try catch

Чтобы проверить, возникают ли ошибки при запуске индексирования, заключите код индексирования в инструкцию try catch.

```
```

JavaScript

```
function IndexTraceTry()
{
    var timeS = (new Date()).getTime();
    try
    {
         var IndexOutput =  host.currentProcess.TTD.Index.ForceBuildIndex();
         host.diagnostics.debugLog("\n>>> Index Return Value: " +
IndexOutput + "\n");
         var timeE = (new Date()).getTime();
         host.diagnostics.debugLog("\n>>> Trace was successfully indexed in
" + (timeE - timeS) + " ms\n");
     }

    catch(err)
    {
         host.diagnostics.debugLog("\n>>> Index Failed! \n");
         host.diagnostics.debugLog("\n>>> Index Return Value: " +
IndexOutput + "\n");
         host.diagnostics.debugLog("\n>>> Returned error: " + err.name +
"\n");
    }
}
```

Ниже приведены выходные данные скрипта, если индексирование выполнено успешно.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.IndexTraceTry()

>>> Index Return Value: Loaded

>>> Trace was successfully indexed in 1 ms
```

Если трассировка не может быть индексирована, например если трассировка не загружена в отладчик, выполняется код цикла catch.

dbgcmd

```
0:007> dx Debugger.State.Scripts.MyTraceUtils.Contents.IndexTraceTry()

>>> Index Failed!

>>> Index Return Value: undefined

>>> Returned error: TypeError
```

# Запросы объектов JavaScript TTD

Более расширенное использование JavaScript и TTD — запрос объектов перемещения по времени для поиска определенных вызовов или событий, произошедших в трассировке. Дополнительные сведения о объектах TTD см. в следующих статье:

Общие сведения о объектах отладки временных путешествий

Собственные объекты отладчика в расширениях JavaScript — сведения об объекте отладчика

Команда dx отображает сведения из модели данных отладчика и поддерживает запросы с помощью синтаксиса LINQ. Dx очень полезно запрашивать объекты в режиме реального времени. Это позволяет создать прототип требуемого запроса, который затем можно автоматизировать с помощью JavaScript. Команда dx предоставляет завершение вкладки, которое может быть полезно при изучении объектной модели. Общие сведения о работе с запросами LINQ и объектами отладчика см. в разделе "Использование LINQ С объектами отладчика".

Эта команда dx подсчитывает все вызовы определенного API в этом примере *GetLastError*.

dbgcmd

```
0:000> dx @$cursession.TTD.Calls("kernelbase!GetLastError").Count()

@$cursession.TTD.Calls("kernelbase! GetLastError").Count() : 0x12
```

Эта команда выглядит во всей трассировке перемещения по времени, чтобы увидеть, когда был вызван GetLastError.

dbgcmd

```
0:000> dx @$cursession.TTD.Calls("kernelbase!GetLastError").Where(c =>
c.ReturnValue != 0)

@$cursession.TTD.Calls("kernelbase!GetLastError").Where(c => c.ReturnValue
!= 0)
    [0x0]
    [0x1]
    [0x2]
    [0x3]
```

## Сравнение строк для TTD. Вызовы объекта для поиска вызовов

В этом примере команды показано, как использовать сравнения строк для поиска конкретных вызовов. В этом примере запрос ищет строку OLE в параметре *lpFileName* функции CreateFileW.

dbgcmd

```
dx -r2 @$cursession.TTD.Calls("kernelbase!CreateFileW").Where(x =>
x.Parameters.lpFileName.ToDisplayString("su").Contains("OLE"))
```

Добавьте . Оператор Select для печати timestart и значения *параметра lpFileName* .

dbgcmd

```
dx -r2 @$cursession.TTD.Calls("kernelbase!CreateFileW").Where(x =>
x.Parameters.lpFileName.ToDisplayString("su").Contains("OLE")).Select(x =>
new { TimeStart = x.TimeStart, lpFileName = x.Parameters.lpFileName })
```

Это создает выходные данные, если TTD. Объект вызовов найден, содержащий целевую информацию.

dbgcmd

```
    [0x0]
        TimeStart        : 6E37:590
        lpFileName       : 0x346a78be90 : "C:\WINDOWS\SYSTEM32\OLEACCRC.DLL"
[Type: wchar_t *]
```

# Отображение количества вызовов в трассировке

После использования команды dx для изучения объектов, с которыми вы хотите работать, вы можете автоматизировать их использование с JavaScript. В этом простом примере TTD. Объект "Вызовы" используется для подсчета вызовов к *базе ядра! GetLastError*.

JavaScript

```
function CountLastErrorCalls()
{
    var LastErrorCalls =
```

```
host.currentSession.TTD.Calls("kernelbase!GetLastError");
    host.diagnostics.debugLog(">>> GetLastError calls in this TTD recording:
" +  LastErrorCalls.Count() +" \n");
}
```

Сохраните скрипт в файле TTDUtils.js и вызовите его с помощью команды dx, чтобы отобразить количество ядровой базы! GetLastError в файле трассировки.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.CountLastErrorCalls()
>>> GetLastError calls in this TTD recording: 18
```

# Отображение кадров в стеке

Для отображения кадров в стеке используется массив.

JavaScript

```
function DisplayStack()
{
// Create an array of stack frames in the current thread
const Frames = Array.from(host.currentThread.Stack.Frames);
host.diagnostics.debugLog(">>> Printing stack \n");
// Print out all of the frame entries in the array
for(const [Idx, Frame] of Frames.entries())
    {
        host.diagnostics.debugLog(">>> Stack Entry -> " + Idx + ":  "+ Frame
+ " \n");
    }
}
```

В этом примере трассировки отображается одна запись стека.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.DisplayStack()
>>> Printing stack
>>> Stack Entry -> 0:  ntdll!LdrInitializeThunk + 0x21
```

# Поиск события и отображение стека

В этом коде находятся все события исключений, а цикл используется для перемещения к каждому из них. Затем currentThread.ID объектов потока TTD

используется для отображения идентификатора потока и currentThread.Stack для отображения всех кадров в стеке.

JavaScript

```
function HardwareExceptionDisplayStack()
{
var exceptionEvents = host.currentProcess.TTD.Events.Where(t => t.Type ==
"Exception");
    for (var curEvent of exceptionEvents)
    {
        // Move to the current event position
        curEvent.Position.SeekTo();
        host.diagnostics.debugLog(">>> The Thread ID (TID) is : " +
host.currentThread.Id + "\n");
        // Create an array of stack frames in the current thread
        const Frames = Array.from(host.currentThread.Stack.Frames);
        host.diagnostics.debugLog(">>> Printing stack \n");
        // Print out all of the frame entries in the array
        for(const [Idx, Frame] of Frames.entries()) {
            host.diagnostics.debugLog(">>> Stack Entry -> " + Idx + ":  "+
Frame + " \n");
        }
    host.diagnostics.debugLog("\n");
    }
}
```

Выходные данные показывают расположение события исключения, TID и кадров стека.

dbgcmd

```
0:000> dx
Debugger.State.Scripts.MyTraceUtils.Contents.HardwareExceptionDisplayStack()
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: 91:0
>>> The Thread ID (TID) is : 5260
>>> Printing stack
>>> Stack Entry -> 0:  0x540020
>>> Stack Entry -> 1:  0x4d0049
>>> Stack Entry -> 2:  DisplayGreeting!__CheckForDebuggerJustMyCode + 0x16d
>>> Stack Entry -> 3:  DisplayGreeting!mainCRTStartup + 0x8
>>> Stack Entry -> 4:  KERNEL32!BaseThreadInitThunk + 0x19
>>> Stack Entry -> 5:  ntdll!__RtlUserThreadStart + 0x2f
>>> Stack Entry -> 6:  ntdll!_RtlUserThreadStart + 0x1b
```

# Поиск события и отправка двух команд

При необходимости можно объединить запросы к объектам TTD и отправлять команды. Этот пример находит каждое событие в трассировке TTD типа ThreadCreated, перемещается в эту позицию и отправляет команды ~Thread Status и !runaway для отображения состояния потока.

JavaScript

```
function ThreadCreateThreadStatus()
{
var threadEvents = host.currentProcess.TTD.Events.Where(t => t.Type ==
"ThreadCreated");
    for (var curEvent of threadEvents)
    {
        // Move to the current event position
       curEvent.Position.SeekTo();
        // Display Information about threads
       host.namespace.Debugger.Utility.Control.ExecuteCommand("~", false);
       host.namespace.Debugger.Utility.Control.ExecuteCommand("!runaway 7",
false);
    }
}
```

При выполнении кода отображается состояние потока в момент возникновения исключения.

dbgcmd

```
0:000> dx
Debugger.State.Scripts.MyTraceUtils.Contents.ThreadCreateThreadStatus()
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: F:0
.  0  Id: 948.148c Suspend: 4096 Teb: 00a33000 Unfrozen
User Mode Time
  Thread       Time
    0:148c     0 days 0:00:00.000
Kernel Mode Time
  Thread       Time
    0:148c     0 days 0:00:00.000
Elapsed Time
  Thread       Time
    0:148c     3474 days 2:27:43.000
```

# Объединение функций служебной программы

В этом последнем примере можно вызвать функции служебной программы, созданные ранее. Сначала мы индексируем трассировку с помощью *IndexTraceTry* , а затем вызовем *ThreadCreateThreadStatus*. Затем мы используем *ResetTrace* для перехода к началу трассировки и последнего вызова *HardwareExceptionDisplayStack*.

JavaScript

```
function ProcessTTDFiles()
{
    try
    {
    IndexTraceTry()
    ThreadCreateThreadStatus()
    ResetTrace()
    HardwareExceptionDisplayStack()
    }

    catch(err)
    {
         host.diagnostics.debugLog("\n >>> Processing of TTD file failed
\n");
    }

}
```

Выполнение этого скрипта в файле трассировки, содержащего исключение оборудования, создает эти выходные данные.

dbgcmd

```
0:000> dx Debugger.State.Scripts.MyTraceUtils.Contents.ProcessTTDFiles()

>>> Index Return Value: Loaded

>>> Trace was successfully indexed in 0 ms
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: F:0
.  0  Id: 948.148c Suspend: 4096 Teb: 00a33000 Unfrozen
User Mode Time
  Thread       Time
    0:148c     0 days 0:00:00.000
Kernel Mode Time
  Thread       Time
    0:148c     0 days 0:00:00.000
Elapsed Time
  Thread       Time
    0:148c     3474 days 2:27:43.000
>>> Printing stack
>>> Stack Entry -> 0:  ntdll!LdrInitializeThunk
>>> Current position in trace file:  F:0
```

```
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: F:0
>>> New position in trace file:  F:0
(948.148c): Break instruction exception - code 80000003 (first/second chance
not available)
Time Travel Position: 91:0
>>> The Thread ID (TID) is : 5260
>>> Printing stack
>>> Stack Entry -> 0:  0x540020
>>> Stack Entry -> 1:  0x4d0049
>>> Stack Entry -> 2:  DisplayGreeting!__CheckForDebuggerJustMyCode + 0x16d
>>> Stack Entry -> 3:  DisplayGreeting!mainCRTStartup + 0x8
>>> Stack Entry -> 4:  KERNEL32!BaseThreadInitThunk + 0x19
>>> Stack Entry -> 5:  ntdll!__RtlUserThreadStart + 0x2f
>>> Stack Entry -> 6:  ntdll!_RtlUserThreadStart + 0x1b
```

---

## См. также

Отладка временных путешествий — обзор

Общие сведения о объектах отладки временных путешествий

Собственные объекты отладчика в расширениях JavaScript — сведения об объекте отладчика

Скрипты отладчика JavaScript

Примеры скриптов отладчика JavaScript

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Отладка временных путешествий — служебная программа командной строки TTD.exe

Статья • 10.04.2025



В этой статье описывается, когда и как использовать служебную программу командной строки TTD.exe для записи трассировки.

## Когда следует использовать служебную программу командной строки TTD.exe

Отладка перемещения по времени (TTD) позволяет записывать выполнение кода приложения или процесса и сохранять его в файле трассировки. Файл можно воспроизвести обратно в отладчике Windows, чтобы найти проблему с выполнением кода.

Для многих сценариев проще всего использовать TTD для записи приложения или процесса непосредственно из пользовательского интерфейса WinDbg. Сведения об отладке перемещения по времени с помощью пользовательского интерфейса WinDbg см. в разделе "Отладка перемещения по времени" — обзор.

У вас могут быть сценарии, в которых требуется только средство записи командной строки TTD: запись на компьютере без установки отладчика, расширенных сценариев записи, автоматизации тестирования и т. д. В этих сценариях можно установить только средство записи командной строки TTD с помощью URL-адреса.

### Запись TTD влияет на записанный процесс

Запись TTD — это инвазивная технология. Вы заметите замедление работы приложения или процесса от 5 до 20 раз или более при записи, в зависимости от выбранных параметров записи и самого приложения.

Созданные файлы трассировки растут со временем и могут занять значительное место в хранилище. Чтобы выполнить трассировку в течение короткого периода времени, захватив интересующие действия программы, закройте трассировку как можно скорее.

После присоединения TTD к процессу его невозможно удалить. Закройте приложение или завершите процесс после завершения записи TTD. Для системных критически важных процессов это потребует перезагрузки ОС.

### Записи TTD могут содержать личные или связанные с безопасностью сведения

Записи TTD фиксируют содержимое памяти и могут содержать личную или связанную с безопасностью информацию, включая, но не обязательно ограничивается путями к файлам, реестром, памятью или содержимым файла. Точные сведения зависят от целевой активности процесса во время его записи.

## Как скачать и установить служебную программу командной строки TTD.exe (предпочтительный метод)

Скачайте служебную программу командной строки TTD здесь: https://aka.ms/ttd/download

Выберите *Установить*, и TTD будет загружен и установлен. Команда TTD добавляется в системный путь и доступна для использования в командной строке после завершения установки.

При возникновении трудностей с установкой, см. раздел "Устранение проблем с установкой с помощью инсталлятора приложений".

На некоторых компьютерах может потребоваться установить Установщик приложений (Майкрософт) для Windows 10. Он доступен в приложении Microsoft Store в Windows. Диспетчер пакетов Windows поддерживается с помощью установщика приложений, начиная с Windows 10 1809.

## Скачивание и установка служебной программы командной строки TTD.exe (автономный метод)

Хотя предпочтительный метод установки — использовать установщик приложений, вы также можете скачать пакет командной строки TTD и извлечь файлы вручную. Вот два способа сделать это.

### Извлечение файлов из уже установленной программы командной строки TTD.exe

Если вы уже установили служебную программу командной строки TTD, можно извлечь файлы из установленного расположения. В PowerShell выполните следующее, чтобы найти установленное расположение:

PowerShell

```
(Get-AppxPackage | where Name -eq 'Microsoft.TimeTravelDebugging').InstallLocation
```

Оттуда можно скопировать все двоичные файлы (*.dll, *.exe, *.sys) в новое расположение. Вот один из способов сделать это в PowerShell:

PowerShell

```
robocopy.exe (Get-AppxPackage | where Name -eq
'Microsoft.TimeTravelDebugging').InstallLocation c:\myttd *.exe *.dll *.sys /E /XD
AppxMetadata
```

Замените "c:\myttd" выбранным местом назначения. Результат будет выглядеть примерно так (на компьютере x64):

Консоль

```
ls -Recurse c:\myttd

    Directory: C:\myttd

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----           11/9/2023  2:43 PM                x86
-a---           11/9/2023  2:43 PM          79240 ProcLaunchMon.sys
-a---           11/9/2023  2:43 PM         112568 TTD.exe
-a---           11/9/2023  2:43 PM         309176 TTDInject.exe
-a---           11/9/2023  2:43 PM          55328 TTDLoader.dll
-a---           11/9/2023  2:43 PM         821176 TTDRecord.dll
-a---           11/9/2023  2:43 PM        1222584 TTDRecordCPU.dll
-a---           11/9/2023  2:43 PM          63416 TTDRecordUI.dll

    Directory: C:\myttd\x86

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---           11/9/2023  2:43 PM         247728 TTDInject.exe
-a---           11/9/2023  2:43 PM          42928 TTDLoader.dll
-a---           11/9/2023  2:43 PM        1128480 TTDRecordCPU.dll
```

Обратите внимание, что двоичные файлы x86 находятся в подкаталоге. Если вам не нужно записывать 32-разрядные процессы, эту папку можно удалить (и можно добавить

/xd x86 в команду robocopy, чтобы не копировать ее в первую очередь). Версия ARM64 не имеет подкаталогов.

TTDRecordUI.dll требуется только в том случае, если вы хотите использовать пользовательский интерфейс для управления записью. Если вам не нужен пользовательский интерфейс, вы можете удалить этот файл.

## Скачайте пакет служебной программы командной строки TTD.exe и извлеките файлы вручную.

Если вы не хотите установить служебную программу командной строки TTD, вы можете скачать пакет и извлечь файлы вручную. Следующий сценарий PowerShell:

- Получите URL-адрес текущей версии TTD с https://aka.ms/ttd/download.
- Скачайте пакет MSIX.
- Извлеките MSIX запрошенной архитектуры из пакета MSIX.
- Извлеките двоичные файлы TTD из MSIX.

PowerShell

```
param(
    $OutDir = ".",
    [ValidateSet("x64", "x86", "arm64")]
    $Arch = "x64"
)

# Ensure the output directory exists
if (!(Test-Path $OutDir)) {
    $null = mkdir $OutDir
}

# Ensure the temp directory exists
$TempDir = Join-Path $OutDir "TempTtd"
if (!(Test-Path $TempDir)) {
    $null = mkdir $TempDir
}

# Determine if the destination already contains binaries
$extensions = @('.dll', '.exe', '.sys')
$existingBinaries = (Get-ChildItem -recurse $OutDir | Where-Object Extension -In
$extensions).Count -gt 0

# Download the appinstaller to find the current uri for the msixbundle
Invoke-WebRequest https://aka.ms/ttd/download -OutFile $TempDir\ttd.appinstaller

# Download the msixbundle
$msixBundleUri = ([xml](Get-Content
$TempDir\ttd.appinstaller)).AppInstaller.MainBundle.Uri
```

```
if ($PSVersionTable.PSVersion.Major -lt 6) {
    # This is a workaround to get better performance on older versions of
PowerShell
    $ProgressPreference = 'SilentlyContinue'
}

# Download the msixbundle (but name as zip for older versions of Expand-Archive)
Invoke-WebRequest $msixBundleUri -OutFile $TempDir\ttd.zip

# Extract the 3 msix files (plus other files)
Expand-Archive -DestinationPath $TempDir\UnzippedBundle $TempDir\ttd.zip -Force

# Expand the build you want - also renaming the msix to zip for Windows PowerShell
$fileName = switch ($Arch) {
    "x64"   { "TTD-x64"   }
    "x86"   { "TTD-x86"   }
    "arm64" { "TTD-ARM64" }
}

# Rename msix (for older versions of Expand-Archive) and extract the debugger
Rename-Item "$TempDir\UnzippedBundle\$fileName.msix" "$fileName.zip"
Expand-Archive -DestinationPath "$OutDir" "$TempDir\UnzippedBundle\$fileName.zip"

# Delete the temp directory
Remove-Item $TempDir -Recurse -Force

# Remove unnecessary files, if it is safe to do so
if (-not $existingBinaries) {
    Get-ChildItem -Recurse -File $OutDir |
        Where-Object Extension -NotIn $extensions |
        Remove-Item -Force

    Remove-Item -Recurse -Force (Join-Path $OutDir "AppxMetadata")
} else {
    Write-Host "Detected pre-existing binaries in '$OutDir' so did not remove any
files from TTD package."
}
```

Если вы сохранили приведенный выше сценарий как `Get-Ttd.ps1`, можно запустить его так, чтобы скачать двоичные файлы x64 в каталог c:\myttd:

PowerShell

```
md c:\myttd
cd c:\myttd
.\Get-Ttd.ps1
```

Или можно указать выходной каталог и архитектуру:

PowerShell

```
.\Get-Ttd.ps1 -OutDir c:\myttd-arm64 -Arch arm64
```

Замените "c:\myttd" или "c:\myttd-arm64" на выбранное место.

# Как записать трассировку с помощью утилиты командной строки TTD.exe

Существует три способа записи трассировки.

- Запуск процесса
- Присоединение к процессу
- Мониторинг процесса

После записи процесса необходимо активировать проблему, которую вы хотите отлаживать. Вы можете открыть проблемный файл или нажать определенную кнопку в приложении, чтобы вызвать событие, интересующее вас. После завершения записи приложения, естественно или путем сбоя, файл трассировки будет завершен.

> **Совет**
>
> Для записи трассировок TTD требуются права администратора. Обычно это делается путем запуска ttd.exe из командной строки администратора.

Дополнительные сведения о записи трассировки путешествия во времени с помощью WinDbg см. в статье "Отладка путешествий во времени" — запись трассировки.

## Запуск процесса

`-launch <Program> [<arguments>]`

Запуск и трассировка программы (режим по умолчанию).

Это единственный режим, позволяющий передавать аргументы программе. Программа запустится с теми же привилегиями, что и TTD.exe (как администратор). Используйте `-attach` или `-monitor`, чтобы записать программу с обычным набором привилегий.

Включение `-launch` является необязательным, но может использоваться для ясности.

Первый нераспознанный аргумент, который не начинается с - или / будет считаться исполняемым путем для запуска, и любые последующие аргументы будут считаться аргументами для этой программы.

Например, используйте `TTD.exe notepad.exe` для запуска и записи блокнота. Трассировка останавливается при закрытии блокнота.

Смотрите Примеры использования сценария - запись процесса.

## Присоединение к процессу

`-attach <PID>`

Присоединение к работающему процессу, указанному идентификатором процесса. Используйте TaskManager или служебную программу TaskList для идентификации номеров процессов. Дополнительные сведения см. в разделе "Поиск идентификатора процесса".

Например, используйте `TTD.exe -attach 21440 -out C:\traces\MyTraceFile.run` для запуска и записи процесса с идентификатором 21440 и сохранением трассировки в MyTraceFile.run.

Перед запуском TTD.exe убедитесь, что каталог существует (`C:\traces` в этом примере).

Например, использование см. в разделе "Сценарий" — поиск и присоединение к запущенному процессу.

## Мониторинг процесса

`-monitor <Program>`

Параметр мониторинга позволяет отслеживать и фиксировать программу при каждом запуске. Чтобы использовать этот параметр, необходимо указать полный путь к выходному расположению `-out`.

Чтобы остановить мониторинг, нажмите клавиши CTRL+C.

Основными преимуществами мониторинга по сравнению с другими методами являются:

- Вы можете запустить целевое приложение обычным способом, не нужно разбираться с командной строкой, чтобы его запустить.
- Целевое приложение будет работать со своими обычными привилегиями. Если запустить приложение непосредственно из ttd.exe он запустится с повышенными привилегиями и может изменить поведение программы.
- Это полезно для автоматизации (используйте скрипт, который отслеживает запуск программы и собирает трассировку).

Параметр -monitor можно указать несколько раз для мониторинга нескольких программ.

Например, см. Примеры использования сценария — процессы мониторинга.

# Параметры командной строки

## Синтаксис

```
TTD.exe [options] [mode] [program [<arguments>]]
```

`-?` | `-help`

Отображение справки командной строки.

## Режимы

```
-launch <Program> [<arguments>]
```

Запуск и трассировка программы (режим по умолчанию).

Это единственный режим, позволяющий передавать аргументы программе. Параметр `-launch` должен быть последним параметром TTD в командной строке, после которого указываются программа для запуска и любые параметры, которые требуются этой программе. Если режим не указан, он также будет рассматриваться как запуск. Например, `TTD.exe -out C:\traces ping.exe msn.com` рассматривается как запуск.

```
-attach <PID>
```

Присоединение к работающему процессу, указанному идентификатором процесса. Используйте служебную программу TaskManager или TaskList для идентификации идентификаторов процессов. Дополнительные сведения см. в разделе "Поиск идентификатора процесса".

```
-monitor <Program>
```

Отслеживание программ или служб при каждом их запуске (до перезагрузки). Чтобы использовать этот параметр, необходимо указать полный путь к выходному расположению `-out`.

## Основные параметры командной строки

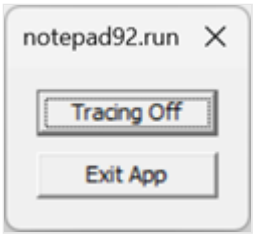
```
-out <path>
```

Укажите имя файла трассировки или каталог. Если это каталог, он уже должен существовать. Если указано имя файла, файл с таким именем не должен существовать.

`-noUI`

Отключает пользовательский интерфейс для ручного управления записью.

Если этот параметр не выбран небольшой пользовательский интерфейс отображается при активной записи. "Отключение трассировки" останавливает трассировку и приложение продолжает работу; "Выход из приложения" закрывает приложение, что также останавливает трассировку.




`-accepteula`

Используйте этот параметр, чтобы принять лицензионное соглашение с пользователем EULA. Этот параметр можно использовать в сценариях автоматизации после проверки и принятия лицензионного соглашения.

TTD отображает EULA при первом запуске. Введите Y или N, чтобы принять EULA. После принятия ELA больше не будет отображаться при запуске. Если лицензионное соглашение (EULA) не принято, TTD завершит работу, и EULA отобразится при следующем запуске TTD.

## Управление трассировкой

`-stop <process name> | <PID> | all`

Остановить трассировку указанного имени процесса, PID или "all" можно указать.

`-wait <timeout>`

Подождите до указанного количества секунд, пока все сессии трассировки в системе не завершатся. Укажите -1, чтобы ждать бесконечно.

`-tracingOff`

Запускает приложение без включения записи трассировки. Вы можете использовать флажок в пользовательском интерфейсе, чтобы снова включить трассировку после её отключения.

## Дополнительные параметры командной строки

`-children`

Запишите целевой объект, а также все процессы, созданные целевым объектом. Каждый дочерний процесс будет записан в собственный файл трассировки.

`-cmdLineFilter "<string>"`

Запишите целевой объект, если его командная строка содержит строку. Этот параметр работает только с `-monitor` режимом. Это полезно для ситуаций, когда аргумент командной строки однозначно идентифицирует интересующий вас процесс. Например, notepad.exe записывается, только если specialfile.txt присутствует в командной строке.

`-cleanup`

Удалите драйвер монитора процесса.

## Параметры поведения трассировки

`-timestampFilename`

Добавляет метку времени в последнюю часть имени файла трассировки. Например, ping_2023-06-17_103116.run.

Например, чтобы записать ping.exe с меткой времени, включенной в имя файла, используйте эту команду.

Консоль

```
ttd.exe  -out c:\traces -timestampFilename ping.exe msn.com
```

По умолчанию последовательное сканирование выполняется для поиска неиспользуемого файла в выходном каталоге. Если ping.exe записан, средство записи попытается выполнить ping01.run, ping02.run и т. д. пока не найдено неиспользуемое имя файла. Для большинства сценариев этот метод именования достаточно. Однако если вы хотите записать одну и ту же программу несколько раз, алгоритм именования файлов по умолчанию может стать неэффективным, если существует большое количество существующих файлов.

`-ring`

Трассировка до кольцевого буфера. Размер файла не будет превышать пределы, указанные в параметре `-maxFile`. Будет сохранена только последняя часть записи,

которая соответствует заданному размеру.

`-maxFile <size>`

Максимальный размер файла трассировки в МБ. Если в полном режиме трассировки значение по умолчанию равно 1024 ГБ, а минимальное значение — 1 МБ. Если в режиме буфера кольца значение по умолчанию равно 2048 МБ, минимальное значение равно 1 МБ, а максимальное значение — 32768 МБ.

Значение по умолчанию для кольца в памяти на 32-разрядных процессах составляет 256 МБ.

`-maxConcurrentRecordings <count>`

Максимальное количество записей, которые могут продолжаться в любой момент времени. Если не указано, одновременно может произойти неограниченное количество записей.

`-numVCpu <number>`

Указывает ряд виртуальных ЦП, которые необходимо зарезервировать и использовать при трассировке. Это значение влияет на общую нагрузку на память, создаваемую на память гостевого процесса отладчиком TTD. Если не указано значение по умолчанию для каждой платформы: 55 для x64/ARM64 и 32 для x86.

Измените этот параметр, чтобы ограничить влияние *памяти только* при нехватке памяти. Изменение значения numVCpu на более низкое число может сильно повлиять на производительность трассировки и должно выполняться только для решения проблем с использованием памяти.

Если TTD.exe не удаётся сделать запись или файл .out указывает на имитацию 0 секунд, использование `-numVCpu` может обеспечить успешную запись.

`-replayCpuSupport <support>`

Указывает, какая поддержка ожидается от центральных процессоров, которые будут использоваться для воспроизведения трассировки. Параметр по умолчанию рекомендуется для переносимости трассировок между компьютерами, но другие параметры могут использоваться для создания небольших файлов трассировки и записи быстрее (в зависимости от конкретных инструкций, используемых целевой программой).

Значения `<support>`

Развернуть таблицу

| значение | Описание |
| --- | --- |
| `Default` | Для поддержки ЦП по умолчанию требуется только базовая, общедоступная поддержка в процессоре воспроизведения. |
| `MostConservative` | Не требуется специальная поддержка в ЦП для воспроизведения. Достаточно для трассировок, которые будут воспроизводиться на совершенно другой архитектуре ЦП, например трассировки Intel на ЦП ARM64. |
| `MostAggressive` | Предполагается, что ЦП воспроизведения будет сходен и равной или большей мощности, чем ЦП, используемый для записи. |
| `IntelAvxRequired` | Предполагается, что процессор для воспроизведения будет 64-разрядным процессором с поддержкой AVX от Intel или AMD. |
| `IntelAvx2Required` | Предполагается, что ЦП воспроизведения будет intel/AMD 64-разрядной ЦП, поддерживающий AVX2. |

## Сокращение затрат на трассировку

Хотя TTD очень эффективен для того, что он делает (полная трассировка уровня инструкций, закодированная в меньшее число байтов или инструкций в среднем), она по-прежнему имеет заметные издержки при записи. Современные ЦП могут выполнять миллиарды инструкций в секунду, что делает даже один байт/инструкции дорогостоящим. Во многих случаях запись всего процесса не требуется.

Для уменьшения затрат на трассировку можно использовать следующие параметры:

`-module <module name>`

Запишите только указанный модуль (например, comdlg32.dll) и код, который он вызывает. Это может быть исполняемый файл или любая библиотека DLL, загруженная исполняемым файлом. Этот параметр можно указать несколько раз для записи нескольких модулей.

Если эта опция используется, целевой процесс выполняется с полной скоростью до выполнения кода в указанном(ых) модуле(ях). Затем TTD записывает процесс до тех пор, пока выполнение не покидает указанные модули, в какой момент запись отключается, а целевой объект возвращается на полную скорость. Поскольку включение и отключение записи обходится дорого, TTD оставляет запись включённой, когда указанный модуль вызывает другие модули в процессе.

`-recordmode <Automatic | Manual>`

Обычно запись начинается, как только TTD внедряет себя в целевой процесс ("Автоматический" режим, значение по умолчанию). Если ваша программа использует API

записи TTD в процессе для управления моментом начала записи, тогда вы можете использовать ручной режим для работы на полной скорости до тех пор, пока программа не вызовет API, чтобы начать запись.

Использование этих параметров может привести к значительному снижению нагрузки на запись и уменьшению размера файла трассировки. Отладка трассировки, записанной с помощью этих параметров, не отличается от трассировки всего процесса. Когда вы достигнете расположения в трассировке, где запись отключена, следующая инструкция в трассировке является первой инструкцией, выполняемой при возобновлении записи.

### Параметры, связанные с событиями

`-passThroughExit`

Передайте значение выхода гостевого процесса в качестве значения выхода TTD.exe. Это значение доступно пакетным файлам с помощью переменной `%ERRORLEVEL%` . PowerShell и другие среды командной строки предлагают механизмы получения значения выхода процесса.

`-onInitCompleteEvent <eventName>`

Позволяет отправить сигнал о событии после завершения инициализации трассировки.

## Примеры использования сценария — запись процесса

### Сценарий. Запуск и запись приложения Для Windows

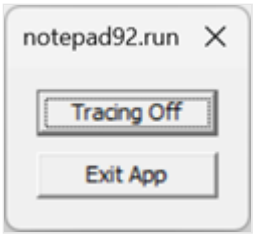
В этом сценарии блокнот запускается и создается трассировка.

1. `-launch` Используйте этот параметр, чтобы запустить блокнот и записать его.

Консоль

```
C:\TTD> TTD.exe -launch notepad.exe
Launching 'notepad.exe'
    Recording process (PID:9960) on trace file: C:\TTD\notepad01.run
notepad.exe(x64) (PID:9960): Process exited with exit code 0 after 12984ms
  Full trace dumped to C:\TTD\notepad01.run
```

2. Отображается небольшое меню приложения, показывающее, что трассировка включена.




3. При закрытии приложения создается файл трассировки. В этом примере notepad01.run.

## Сценарий. Запуск и запись приложения Windows с переданным параметром

В этом сценарии запускается ping, а адрес для проверки передается в качестве параметра.

1. В этом примере `-launch` параметр опущен так, как это режим по умолчанию.

Консоль

```
C:\TTD> TTD.exe ping.exe msn.com
Launching 'ping.exe msn.com'
    Recording process (PID:24044) on trace file: C:\TTD\ping01.run

Pinging msn.com [204.79.197.219] with 32 bytes of data:
Reply from 204.79.197.219: bytes=32 time=22ms TTL=118
Reply from 204.79.197.219: bytes=32 time=21ms TTL=118
Reply from 204.79.197.219: bytes=32 time=25ms TTL=118
Reply from 204.79.197.219: bytes=32 time=21ms TTL=118

Ping statistics for 204.79.197.219:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 21ms, Maximum = 25ms, Average = 22ms
ping.exe(x64) (PID:24044): Process exited with exit code 0 after 3390ms
  Full trace dumped to C:\TTD\ping01.run
```

2. При закрытии приложения создается файл трассировки. В этом примере ping01.run.

## Сценарий - Поиск и присоединение к запущенному процессу

В этом сценарии блокнот запущен, его идентификатор процесса определяется, и трассировка создается посредством присоединения к работающему приложению.

1. Запустите целевое приложение, например, блокнот.

2. Используйте TaskList или другие методы, описанные в описании для поиска идентификатора процесса. Дополнительные сведения см. в разделе "Поиск идентификатора процесса".

Консоль

```
C:\TTD> TaskList
...
Notepad.exe                  21440 Console                    1     73,020 K
...
```

3. Используя этот идентификатор процесса, используйте `-attach` параметр для присоединения и записи. При необходимости укажите имя файла трассировки с помощью `-out`.

Консоль

```
C:\TTD> TTD.exe -attach 21440 -out C:\TTD\MyTraceFile.run
Attaching to 21440
    Recording process (PID:21440) on trace file: C:\TTD\MyTraceFile.run
(x64) (PID:21440): Process exited with exit code 0 after 26672ms
  Full trace dumped to C:\TTD\MyTraceFile.run
```

## Сценарий: Запись родительских и дочерних процессов

В этом сценарии будут записаны родительские и дочерние процессы. Так как некоторые приложения могут использовать множество дочерних процессов, файл трассировки семьи, содержащий дочерние элементы, может стать довольно большим.

1. `-children` Укажите параметр и имя родительского приложения для записи.

Это пример записи запуска cmd.exe, которое вызывает ping.exe как дочерний процесс.

Консоль

```
ttd.exe -out d:\traces -children cmd.exe /C ping.exe msn.com

Microsoft (R) TTD 1.01.11
Release: 1.11.0.0
Copyright (C) Microsoft Corporation. All rights reserved.

Launching 'cmd.exe /C ping.exe msn.com'
    Recording process (PID:48200) on trace file: d:\traces\cmd01.run
    Recording process (PID:53724) on trace file: d:\traces\PING01.run

Pinging msn.com [204.79.197.219] with 32 bytes of data:
Reply from 204.79.197.219: bytes=32 time=6ms TTL=117
```

```
Reply from 204.79.197.219: bytes=32 time=6ms TTL=117
Reply from 204.79.197.219: bytes=32 time=7ms TTL=117
Reply from 204.79.197.219: bytes=32 time=7ms TTL=117

Ping statistics for 204.79.197.219:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 7ms, Average = 6ms
(x64) (PID:53724): Process exited with exit code 0 after 3516ms
  Trace family nesting level is 1; Parent process ID is 48200
  Full trace dumped to d:\traces\PING01.run

...
```

2. Создаются несколько файлов трассировки: один для родительского процесса и файла трассировки для каждого дочернего процесса. WinDbg открывает только один файл трассировки одновременно, поэтому вам потребуется запустить отдельные экземпляры WinDbg для каждой трассировки, если вы хотите отладить их одновременно.

# Примеры использования сценария — процессы мониторинга

## Сценарий— мониторинг запуска и записи программ

В этом сценарии `-monitor` параметр используется для записи всех запущенных экземпляров, а также будущих экземпляров notepad.exe, пока система не перезагружается или ttd.exe завершается с помощью CTRL+C. Параметр `-out` необходим для монитора, и папка вывода должна существовать уже.

1. Контролируйте и отслеживайте текущие, а также любые будущие экземпляры notepad.exe.

Консоль

```
C:\TTD> TTD.exe -out C:\TTD\ -monitor notepad.exe
Microsoft (R) TTD 1.01.11
Release: 1.11.121.0
Copyright (C) Microsoft Corporation. All rights reserved.


The Process Launch Monitor driver is not installed
Successfully installed the Process Launch Monitor driver
Recording process Notepad.exe(15904)        From parent process explorer.exe(8440)
    Recording process (PID:15904) on trace file: C:\TTD\notepad01.run
```

```
Recording process Notepad.exe(19920)        From parent process explorer.exe(8440)
    Recording process (PID:19920) on trace file: C:\TTD\notepad02.run
(x64) (PID:19920): Process exited with exit code 0 after 1281ms
  Full trace dumped to C:\TTD\notepad02.run

(x64) (PID:15904): Process exited with exit code 0 after 30719ms
  Full trace dumped to C:\TTD\notepad01.run
```

2. В этом примере после начала трассировки загрузились две программы notepad.exe. После фиксации интересующего вас действия, командой CTRL-C в командной строке, остановили запись.

## Сценарий - мониторинг запуска двух программ

В этом сценарии `-monitor` параметр используется для мониторинга и записи двух приложений.

1. Отслеживайте текущие и любые будущие экземпляры notepad.exe и ping.exe.

Консоль

```
C:\TTD> TTD.exe -out C:\TTD\ -monitor notepad.exe -monitor ping.exe
Microsoft (R) TTD 1.01.11
Release: 1.11.121.0
Copyright (C) Microsoft Corporation. All rights reserved.


Successfully uninstalled the Process Launch Monitor driver
Successfully installed the Process Launch Monitor driver
Recording process Notepad.exe(17972)        From parent process explorer.exe(8440)
    Recording process (PID:17972) on trace file: C:\TTD\Notepad01.run
Tracking process svchost.exe(7824)        From parent process services.exe(1292)
Tracking process sppsvc.exe(10376)        From parent process services.exe(1292)
Tracking process ClipUp.exe(15108)        From parent process svchost.exe(7824)
Tracking process ClipUp.exe(21180)        From parent process ClipUp.exe(15108)
Tracking process consent.exe(24280)        From parent process svchost.exe(892)
Tracking process ctfmon.exe(24508)        From parent process svchost.exe(5064)
Tracking process wt.exe(10768)        From parent process explorer.exe(8440)
Tracking process WindowsTerminal.exe(23296)        From parent process
wt.exe(10768)
Tracking process OpenConsole.exe(6816)        From parent process
WindowsTerminal.exe(23296)
Tracking process powershell.exe(15956)        From parent process
WindowsTerminal.exe(23296)
Tracking process git.exe(3656)        From parent process powershell.exe(15956)
Tracking process git.exe(1928)        From parent process git.exe(3656)
Tracking process git.exe(20312)        From parent process powershell.exe(15956)
Tracking process git.exe(5712)        From parent process git.exe(20312)
Tracking process csc.exe(16144)        From parent process powershell.exe(15956)
```

```
Tracking process cvtres.exe(19488)        From parent process csc.exe(16144)
Recording process PING.EXE(21468)        From parent process powershell.exe(15956)
    Recording process (PID:21468) on trace file: C:\TTD\PING01.run
(x64) (PID:21468): Process exited with exit code 1 after 234ms
  Full trace dumped to C:\TTD\PING01.run


Tracking process Teams.exe(10060)        From parent process Teams.exe(2404)
Tracking process cmd.exe(21796)        From parent process powershell.exe(15956)
Recording process PING.EXE(364)        From parent process cmd.exe(21796)
    Recording process (PID:364) on trace file: C:\TTD\PING02.run
(x64) (PID:364): Process exited with exit code 1 after 234ms
  Full trace dumped to C:\TTD\PING02.run
```

2. В этом примере notepad.exe, а затем ping.exe были загружены после начала трассировки. После того как нужное действие было записано, команда CTRL-C использовалась в командной строке для остановки записи.

## Сценарий. Остановка записи во втором окне

В этом сценарии интересующая активность была зафиксирована, и все записи были остановлены с помощью `-stop all`. Для выполнения `-stop all` параметра используется второе окно команды.

Консоль

```
C:\TTD> TTD.exe -stop all
Microsoft (R) TTD 1.01.11
Release: 1.11.121.0
Copyright (C) Microsoft Corporation. All rights reserved.

Full trace written to 'C:\TTD\Notepad01.run'
```

## Сценарий. Очистка драйвера монитора

В этом сценарии `-cleanup` параметр используется для очистки драйвера монитора после завершения записи.

Консоль

```
C:\TTD> TTD.exe -cleanup
The monitor service is not installed
Successfully uninstalled the Process Launch Monitor driver
```

## Дополнительные примеры командной строки

В этой таблице выделены некоторые дополнительные примеры использования командной строки. Дополнительные сведения о иллюстрированных параметрах см. в параметрах командной строки.

Развернуть таблицу

| Сценарий | Команда | Описание |
|---|---|---|
| Присоединитесь к процессу, но пока не начинайте запись | `Ttd.exe -tracingoff notepad.exe` | Запускает блокнот при отключенной записи. Запись может быть запущена в любое время через пользовательский интерфейс. |
| Фильтрация по командной строке | `Ttd.exe -cmdlinefilter foo.txt -monitor notepad.exe` | Записывайте notepad.exe только если foo.txt указан в командной строке при запуске, размещая выходные данные в текущем каталоге. |
| Запись звонка | `Ttd.exe -ring -attach 1234` | Записывает PID 1234 в файл трассировки, который ограничен 2 ГБ, размещая выходные данные в текущем каталоге. Старое содержимое в файле трассировки перезаписывается по мере необходимости, чтобы сохранить файл в пределах максимального размера.<br>Используется `-maxfile` для изменения максимального размера. |
| Ограничение размера файла трассировки | `Ttd.exe -maxfile 4096 notepad.exe` | Записывайте notepad.exe до тех пор, пока файл трассировки не достигнет 4 ГБ, сохраняя выходные данные в текущей директории. |
| Ограничение количества записей, которые происходят одновременно | `Ttd.exe -maxconcurrentrecordings 1 -out c:\my\dir -monitor notepad.exe` | Запись требует интенсивной загрузки ЦП, что может быть критично в некоторых случаях. |
| Сокращение использования памяти в целевом процессе | `Ttd.exe -numvcpu 8 -monitor w3wp.exe` | Некоторые процессы, такие как w3wp.exe, задают небольшую квоту на объем памяти, которую он может использовать. Если ttd.exe не удается начать запись, используйте `-numvcpu` для уменьшения числа виртуальных ЦП, выделенных TTD. Попробуйте использовать этот параметр, только если ttd.exe не может записать с помощью других средств. |
| Выбор между переносимостью трассировки и | `Ttd.exe -replaycpusupport mostaggressive notepad.exe` | По умолчанию TTD создает файлы трассировки, переносимые по широкому спектру оборудования. При выборе |

| Сценарий | Команда | Описание |
| --- | --- | --- |
| скоростью записи, а также размером файла трассировки | | параметра "mostaggressive" TTD сообщается, что допустимо записать трассировку, которая может быть воспроизведена только на ЦП с теми же возможностями, что и компьютер, который записал трассировку. В некоторых случаях это может значительно повысить скорость записи и размер файла трассировки. |

## Примеры командной строки автоматизации

В этой таблице приведены некоторые дополнительные примеры использования командной строки, которые могут быть полезны для автоматического использования служебной программы TTD.exe.

Развернуть таблицу

| Сценарий | Команда | Описание |
| --- | --- | --- |
| Отключение пользовательского интерфейса | `Ttd.exe -noui -accepteula notepad.exe` | Запись notepad.exe, размещение выходных данных в текущем каталоге без отображения пользовательского интерфейса. |
| Подождите, пока запись будет запущена программным способом | `Ttd.exe -accepteula -oninitcompleteevent ttd_notepad notepad.exe` | Создайте событие Win32 с именем "ttd_notepad" и запустите notepad.exe. TTD подаст сигнал "ttd_notepad" при инициализации записи. Автоматизация может ждать события, прежде чем продолжить работу с поведением, которое они хотят записать. |
| Сохранение кода выхода целевого объекта | `Ttd.exe -accepteula -passthroughexit ping.exe msn.com` | Записывает результат выполнения ping.exe, помещая результаты в текущий каталог. код выхода Ttd.exe будет совпадать с кодом выхода ping.exe. |
| Дождитесь завершения записи | `Ttd.exe -accepteula -wait 30` | После остановки записи подождите до 30 секунд, пока TTD завершит запись файла трассировки на диск. Используйте `-wait -1` для ожидания на неопределенный срок. |

Все эти примеры используют `-accepteula` параметр, чтобы убедиться, что автоматизация не заблокирована диалоговым окном подтверждения EULA.

# Работа с созданным файлом трассировки

Сведения о работе с трассировкой и инструкциями по воспроизведению трассировок перемещения по времени и переходу вперед и обратно в времени см. в статье "Отладка перемещения по времени" — воспроизведение трассировки.

## Советы по работе с файлами трассировки

- При обмене трассировками с другими пользователями необходимо предоставить общий доступ только к файлу .run. Файл индекса (.idx) может быть таким же большим, как файл .run, и автоматически создается при загрузке файла трассировки в WinDbg.
- При совместной работе с другими пользователями передайте все соответствующие позиции трассировки, связанные с проблемой. Сотрудник может использовать `!tt x:y` команду, чтобы перейти к этой точной точке во времени выполнения кода. Диапазоны позиций времени можно включить в описания ошибок, чтобы отслеживать, где может возникнуть проблема.
- При сообщении о проблеме с TTD, если вы предоставляете файл .run, также предоставьте и файл .out. Это позволяет подтвердить правильное выполнение процесса записи.
- Файлы трассировки (.run) хорошо сжимаются.

## Устранение неполадок TTD.exe

В некоторых случаях могут возникать ошибки в файле трассировки. Дополнительные сведения см. в статье "Отладка перемещения по времени" — устранение неполадок.

Файл .out можно использовать для устранения неполадок. Файл-пример показывает функциональный след, заканчивающийся кодом выхода равным нулю.

Консоль

```
Microsoft (R) TTDRecord 1.01.11
Release: 1.11.47.0
Copyright (C) Microsoft Corporation. All rights reserved.


Initializing Time Travel Debugging for Attach to 9916
Time: 05/08/2023 17:07:15
OS:10.0.25314 EDITION:x64

SessionID: 008F918C-B8A7-4C4E-B91B-34CFC953C501

   (TTD::ManageTTDTrace:2725)
```

```
Running
   (TTD::StartGuestProcess:1512)
Group tracing GUID: B84DF180-DA54-46E5-9019-73690C689979

Running "C:\WINDOWS\SYSTEM32\TTDInject.exe" /duration 1 /InjectMode
LoaderForCombinedRecording /ClientParams "37 C:\TTD\Notepad03.run 0 0 0 0 0 0 0 0
c06001 0" /RecordScenario 268435458 /attach 9916 -TraceFileHandle 4f8 -
GuestEventHandle 380 -ClientEventHandle 384 -ActiveEventHandle 4f4 -MutexHandle
46c -CommunicationBufferHandle 3c0 -SharedSequenceMutexHandle 3b8 -
SharedSequenceBufferHandle 330 /TelemetryFeatureSessionId "008F918C-B8A7-4C4E-
B91B-34CFC953C501"
   (TTD::StartGuestProcess:1955)
Microsoft (R) TTDInject 1.01.11
Release: 1.11.27.0
Copyright (C) Microsoft Corporation. All rights reserved.

TTDLoader Params:
 LauncherDll = TTDLoader
 ClientDll   = TTDRecordCPU
 ClientEntry = InitializeNirvanaClient
 ClientParams= 37 C:\TTD\Notepad03.run 0 0 0 0 0 0 0 0 c06001 0
 Attach
WaitForMain is off
Allocated processors:55, running threads:2.
Loader TTDLoader.dll injected at 0x00007FFF423B0000 0xc000 -- .reload
TTDLoader.dll=0x00007FFF423B0000,0xc000

Injection by thread is complete.
RecordingEngine initialization successful.
RecordVcpu initialization successful.
Loader initialization successful.
Guest Process is x64 binary.
Tracing started at: Tue May  9 00:07:16 2023 (UTC) Mon May  8 17:07:16 2023
(Local)

Guest process exited with exit code 0
Simulation time of '' (x64): 18781ms.
Tracing completed at: Tue May  9 00:07:34 2023 (UTC) Mon May  8 17:07:34 2023
(Local)
```

Основная часть содержимого файла out используется внутренне командой, занимающейся отладкой времени, для исправления ошибок записи. Следующие сведения могут оказаться полезными для других пользователей, работающих с файлом трассировки.

- Некоторые сообщения об ошибках отображаются только в файле out и могут использоваться для определения особенностей сбоя.
- Указание времени на настенных часах, когда началась или остановилась запись
- Сколько времени длилось сеанс записи (время имитации)
- Указывает, является ли запись запуском (с командной строкой) или присоединением записи

- Версия ОС

## См. также

Отладка временных путешествий — обзор

Отладка перемещения по времени — запись трассировки

Отладка перемещения по времени — воспроизведение трассировки

# Использование команд отладчика

Статья • 08.07.2024

В этом разделе описывается использование команд отладчика. WinDbg — это отладчик, который можно использовать для анализа аварийных дампов, отладки динамического пользовательского режима и кода в режиме ядра, а также проверки регистров ЦП и памяти. Дополнительные сведения см. в разделе "Обзор WinDbg".

Сведения об установке отладчика см. в разделе "Установка отладчика Windows".

Сведения о начале работы с WinDbg см. в статье "Начало работы с отладкой Windows".

## Командное окно отладчика WinDbg

Для WinDbg "Окно командной строки отладчика" ссылается на окно с меткой "Command" в строке заголовка. Это окно содержит две области:

- В небольшой нижней области введите команды.
- В большой верхней области вы просматриваете выходные данные команды.

Это окно всегда открыто в начале сеанса отладки. Вы можете повторно открыть или переключиться в это окно, выбрав **команду в меню "Вид", нажав клавиши ALT+1 или нажав кнопку "Команда" (ALT+1)** (

Для прокрутки журнала команд можно использовать клавиши СТРЕЛКА ВВЕРХ и СТРЕЛКА ВНИЗ. Когда появится предыдущая команда, ее можно изменить, а затем нажать клавишу ВВОД, чтобы выполнить предыдущую команду (или редактированную версию предыдущей команды). Курсор не должен находиться в конце строки, чтобы эта процедура работала правильно.

## KD или CDB

Для KD или CDB "Командное окно отладчика" ссылается на все окно. Введите команды в строке в нижней части окна. Если команды имеют какие-либо выходные данные, окно отображает выходные данные, а затем снова отображает запрос.

## Командная строка отладчика

При отладке в пользовательском режиме запрос в окне команды отладчика выглядит следующим образом.

```
2:005>
```

В предыдущем примере 2 — текущий номер процесса, а 005 — текущий номер потока.

При присоединении отладчика к нескольким компьютерам системный номер включается до числа процесса и потока, как показано в следующем примере.

```
3:2:005>
```

В этом примере 3 — текущий номер системы, 2 — текущий номер процесса, а 005 — текущий номер потока.

При отладке в режиме ядра на целевом компьютере с одним процессором запрос выглядит следующим образом.

```
kd>
```

Однако если целевой компьютер имеет несколько процессоров, число текущего процессора отображается перед запросом, как показано в следующем примере.

```
0: kd>
```

Если отладчик занят обработкой ранее выданной команды, новые команды временно не обрабатываются, хотя их можно добавить в буфер команд. Кроме того, вы по-прежнему можете использовать клавиши управления в KD и CDB, и вы по-прежнему можете использовать команды меню и сочетания клавиш в WinDbg. Если KD или CDB находится в этом состоянии занятости, запрос не отображается. Если WinDbg находится в этом состоянии занятости, вместо запроса появится следующий индикатор:

```
*BUSY*
```

Для добавления текста в эту строку можно использовать **команду PCMD (Set Prompt Command** ).

## Виды команд

WinDbg, KD и CDB поддерживают различные команды. Некоторые команды совместно используются между отладчиками, а некоторые доступны только для одного или двух отладчиков.

Некоторые команды доступны только в динамической отладке, а другие команды доступны только при отладке файла дампа.

Некоторые команды доступны только во время отладки в пользовательском режиме, а другие команды доступны только во время отладки в режиме ядра.

Некоторые команды доступны только в том случае, если целевой объект работает на определенных процессорах. Дополнительные сведения обо всех командах и их ограничениях см. в разделе "Команды отладчика".

## Редактирование, повторение и отмена команд

При вводе команды можно использовать стандартные ключи редактирования:

- Используйте клавиши СТРЕЛКА ВВЕРХ и СТРЕЛКА ВНИЗ, чтобы найти предыдущие команды.
- Измените текущую команду с помощью клавиш BACKSPACE, DELETE, INSERT и СТРЕЛКА ВЛЕВО и СТРЕЛКА ВПРАВО.
- Нажмите клавишу ESC, чтобы очистить текущую строку.

Вы можете нажать клавишу TAB, чтобы автоматически завершить запись текста. В любом из отладчиков нажимайте клавишу TAB после ввода хотя бы одного символа, чтобы автоматически завершить команду. Нажмите клавишу TAB несколько раз, чтобы выполнить циклический цикл по параметрам завершения текста, а затем удерживайте клавишу SHIFT и нажмите клавишу TAB, чтобы переключиться назад. Вы также можете использовать подстановочные знаки в тексте и нажать клавишу TAB, чтобы развернуть до полного набора параметров завершения текста. Например, если вы вводите **fo*!ba** , а затем нажимаете клавишу TAB, отладчик расширяет набор всех символов, начинающихся с ba, во всех модулях с именами модулей, начинающимися с "fo". В качестве другого примера можно выполнить все команды расширения, имеющие prcb в них, введя **!*prcb** и нажав клавишу TAB.

При использовании клавиши TAB для выполнения завершения текста, если фрагмент текста начинается с точки (.), текст сопоставляется с командой dot. Если фрагмент текста начинается с восклицательного знака (!), текст сопоставляется с командой расширения. В противном случае текст сопоставляется с символом. При использовании клавиши TAB для ввода символов нажатие клавиши TAB завершает код и символы типов и имена модулей. Если имя модуля не очевидно, то будут завершены локальные символы и имена модулей. Если задан шаблон модуля или модуля, завершение символов завершает код и символы типа из всех совпадений.

Вы можете выбрать и сохранить (или щелкнуть правой кнопкой мыши) в командном окне отладчика, чтобы автоматически вставить содержимое буфера обмена в команду, которую вы вводите.

Максимальная длина команды — 4096 символов. Однако если вы управляете отладчиком пользовательского режима из отладчика ядра, максимальная длина строки составляет 512 символов.

В CDB и KD нажмите клавишу ВВОД, чтобы повторить предыдущую команду. В WinDbg можно включить или отключить это поведение. Дополнительные сведения об этом поведении см. в разделе **ВВОД (повторная последняя команда)**.

Если последняя команда, выданная вами, представляет длинное отображение и вы хотите отключить ее, используйте клавиши **CTRL+C** в CDB или KD. В WinDbg используйте **отладку | Разорвать** или нажать клавиши CTRL+BREAK.

При отладке в режиме ядра можно отменить команды с клавиатуры целевого компьютера, нажав клавиши **CTRL+C**.

Для очистки всего текста из командного окна отладчика можно использовать **команду CLS (очистить экран**). Эта команда очищает всю историю команд. В WinDbg можно очистить журнал команд с помощью **команды Edit | Снимите команду "Вывод** команд" или выберите **"Очистить выходные данные** команды" в контекстном меню окна команды отладчика.

## Синтаксис выражений

Многие команды и команды расширения принимают *выражения* в качестве аргументов. Отладчик вычисляет эти выражения перед выполнением команды. Дополнительные сведения о выражениях см. в разделе "Оценка выражений".

## Aliases

*Псевдонимы* — это текстовые макросы, которые можно использовать, чтобы избежать необходимости повторно вводить сложные фразы. Существует два типа псевдонимов. Дополнительные сведения об псевдонимах см. в разделе "Использование псевдонимов".

## Само повторяющиеся команды

Для повторения действия или условного выполнения других команд можно использовать следующие команды:

- Условная **команда j (Execute If-Else)**
- Условная **команда z (выполнение в то время как)**
- **Квалификатор команды ~e (команда для конкретного потока)**
- **Команда расширения !list**

Дополнительные сведения о каждой команде см. в отдельных разделах команд.

## Управление прокруткой

Полоса прокрутки можно использовать для просмотра предыдущих команд и их выходных данных.

При использовании CDB или KD любая запись клавиатуры автоматически прокручивает окно команды отладчика вниз вниз.

В WinDbg экран автоматически прокручивается вниз, когда команда создает выходные данные или нажимаете клавишу ВВОД. Если вы хотите отключить эту автоматическую прокрутку, выберите **пункт "Параметры** " в **меню "Вид** ", а затем снимите **флажок "Автоматически прокручивать** ".

## Функции текста WinDbg

В WinDbg можно использовать несколько дополнительных функций, чтобы изменить способ отображения текста в окне команды отладчика. Вы можете получить доступ к некоторым из этих функций в окне WinDbg, некоторые из них в контекстном меню в окне команды отладчика и некоторые из них, выбрав соответствующий значок меню.

- Команда **оболочки** Word в контекстном меню включает и отключает состояние оболочки слова. Эта команда влияет на все окно, а не только команды, которые вы используете после изменения этого состояния. Так как многие команды и расширения создают отформатированные экраны, обычно не рекомендуется обтекать словами.
- Изменение | **Команда "Добавить в меню вывода** команд" добавляет комментарий в окне команды отладчика. Команда **Add to command output** command в контекстном меню имеет тот же эффект.
- Вы можете настроить цвета, используемые для текста и фона окна команды отладчика. Можно указать разные цвета для разных типов текста. Например, можно отобразить выходные данные автоматического регистра в одном

цвете, сообщения об ошибках в другом цвете и **сообщения DbgPrint** в третьем цвете.

- Вы можете использовать все функции, общие для окон отладки WinDbg, таких как настройка шрифтов и использование специальных команд редактирования.

## Удаленная отладка

При выполнении удаленной отладки с помощью отладчика клиент отладки может получить доступ к ограниченному количеству команд. Чтобы изменить количество команд, к которым клиент может получить доступ, используйте **параметр командной строки -clines** или переменную среды _NT_DEBUG_HISTORY_SIZE.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Вычисление выражений

Статья • 26.10.2023

Отладчик понимает две различные формы выражений: *выражения MASM* и *выражения C++*.

Выражения ассемблеров Microsoft (MASM) используются в примерах в этой справочной документации, за исключением случаев, когда указано иное. В выражениях MASM все символы обрабатываются как адреса.

Выражения C++ те же, что и в фактическом коде C++. В этих выражениях символы понимаются как соответствующие типы данных.

## При использовании каждого синтаксиса

Вы можете выбрать вычислитель выражений по умолчанию одним из следующих способов:

- Используйте переменную среды _NT_EXPR_EVAL перед запуском отладчика.
- При запуске отладчика используйте параметр командной строки **-ee** {**masm**|**c++**}.
- Используйте команду **.expr (Выбрать вычислитель выражений),** чтобы отобразить или изменить средство оценки выражений после запуска отладчика.

Если вы не используете ни один из предыдущих методов, отладчик использует средство оценки выражений MASM.

Если вы хотите вычислить выражение без изменения состояния отладчика, можно использовать **? (Вычисление выражения)** Команды.

Все команды и окна сведений об отладке интерпретируют свои аргументы через средство оценки выражений по умолчанию, за следующими исключениями:

- ? **? (Оценка выражения C++)** Команда всегда использует средство оценки выражений C++.
- В окне Контрольные значения всегда используется средство оценки выражений C++.
- В окне Локальные всегда используется средство оценки выражений C++.

- Некоторые команды расширения всегда используют средство оценки выражений MASM (а другие команды расширения принимают только числовые аргументы, а не полные выражения).

- Если какая-либо часть выражения заключена в круглые скобки и перед выражением вставляется два знака (@@), то выражение вычисляется вычислителем выражений, который обычно не используется в этом случае.

Два знака (@@) позволяют использовать два разных средства оценки для разных параметров одной команды. Он также позволяет оценивать различные фрагменты длинного выражения различными методами. Вы можете вложить их в знаки. Каждое появление двух знаков переключается на другой вычислитель выражений.

**Предупреждение** Синтаксис выражений C++ полезен для управления структурами и переменными, но не подходит для синтаксического анализа параметров команд отладчика. При использовании команд отладчика в общих целях или расширений отладчика следует задать синтаксис выражений MASM в качестве средства оценки выражений по умолчанию. Если у вас должен быть определенный параметр, используйте синтаксис выражений C++, используйте синтаксис двух знаков (@@).

Дополнительные сведения о двух разных типах выражений см. в разделе Синтаксис числовых выражений.

## Числа в выражениях

Числа в выражениях MASM интерпретируются в соответствии с текущим радиксом. Команда **n (задать числовую базу)** может использоваться, чтобы задать для радикса по умолчанию значение 16, 10 или 8. Все нефиксированные числа будут интерпретироваться в этой базе. Радикс по умолчанию можно переопределить, указав префикс **0x** (шестнадцатеричный), префикс **0n** (десятичный), префикс **0t** (восьмеричный) или префикс **0y** (двоичный).

Числа в выражениях C++ интерпретируются как десятичные числа, если не указано другое. Чтобы указать шестнадцатеричное целое число, добавьте **0x** перед числом. Чтобы указать восьмеричное целое число, добавьте **0** (ноль) перед числом. (Однако в *выходных данных* отладчика иногда используется десятичный префикс **0n** .)

Если вы хотите отобразить число в нескольких базах одновременно, используйте команду **.formats (Показать числовые форматы).**

## Символы в выражениях

Два типа выражений интерпретируют символы по-разному:

- В выражениях MASM каждый символ интерпретируется как адрес. В зависимости от того, к чему относится символ, этот адрес является адресом глобальной переменной, локальной переменной, функции, сегмента, модуля или любой другой распознаваемой метки.

- В выражениях C++ каждый символ интерпретируется в соответствии с его типом. В зависимости от того, на что ссылается символ, он может интерпретироваться как целое число, структура данных, указатель функции или любой другой тип данных. Символ, не соответствующий типу данных C++ (например, неизмененному имени модуля), создает синтаксическую ошибку.

Если символ может быть неоднозначным, предшествуйте ему имя модуля и восклицательный знак ( **!** ). Если имя символа может быть интерпретировано как шестнадцатеричное число, предшествуйте ему имя модуля и восклицательный знак ( **!** ) или только восклицательный знак. Чтобы указать, что символ должен быть локальным, опустите имя модуля и добавьте знак доллара и восклицательный знак ( **$!** ) перед именем символа. Дополнительные сведения о интерпретации символов см. в разделе Синтаксис символов и сопоставление символов.

## Операторы в выражениях

Каждый тип выражения использует разные коллекции операторов.

Дополнительные сведения об операторах, которые можно использовать в выражениях MASM, и правилах их приоритета см. в разделе Числа и операторы MASM.

Дополнительные сведения об операторах, которые можно использовать в выражениях C++, и правилах их приоритета см. в разделе Числа и операторы C++.

Помните, что операции MASM всегда основаны на байтах, а операции C++ соответствуют правилам типа C++ (включая масштабирование арифметики указателей).

Некоторые примеры различных синтаксисов см. в разделе Примеры смешанных выражений.

# Использование команд оболочки

Статья • 03.04.2025

Отладчик может передавать определенные команды в среду Microsoft Windows, в которой выполняется отладчик.

Команду **.shell (Command Shell)** можно использовать в любом отладчике Windows. С помощью этой команды можно выполнить приложение или команду Microsoft MS-DOS непосредственно из отладчика. При удаленной отладке эти команды оболочки выполняются на сервере.

Команда **.noshell (запретить команды оболочки)** или параметр командной строки **-noshell** отключает все команды оболочки. Команды отключены во время выполнения отладчика, даже если вы начинаете новый сеанс отладки. Команды остаются отключенными, даже если вы выдаете команду **перезапуска (перезапустить подключение ядра)** в KD.

Если вы запускаете сервер отладки, возможно, вы захотите отключить команды оболочки. Если оболочка доступна, удаленное подключение может использовать команду **.shell** для внесения изменений в ваш компьютер.

## Управление сетевым диском

В WinDbg можно использовать **файл | Сопоставление сетевого диска** и **файла | Отключите команды сетевого диска** для управления сопоставлениями сетевых дисков. Эти изменения всегда происходят на компьютере, на котором работает WinDbg, никогда не на любом компьютере, удаленно подключенном к WinDbg.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Использование псевдонимов

Статья • 26.10.2023

*Псевдонимы* — это символьные строки, которые автоматически заменяются другими строками символов. Их можно использовать в командах отладчика, чтобы избежать повторения некоторых распространенных фраз.

Псевдоним состоит из *имени псевдонима* и *эквивалентного псевдонима*. При использовании имени псевдонима в составе команды отладчика имя автоматически заменяется эквивалентным псевдонимом. Эта замена происходит непосредственно перед анализом или выполнением команды.

Отладчик поддерживает три типа псевдонимов:

- Вы можете задать *псевдонимы с именами пользователей* и присвоить им имена.
- *Псевдонимы с фиксированным именем* можно задать, но они называются **$u 0**, **$u 1**, ..., **$u 9**.
- Отладчик задает и *именует автоматические псевдонимы*.

## Определение псевдонима User-Named

При определении псевдонима с именем пользователя можно выбрать имя псевдонима и эквивалент псевдонима:

- Псевдоним может быть любой строкой, которая не содержит пробелов.
- Эквивалентом псевдонима может быть любая строка. Если ввести его на клавиатуре, эквивалентный псевдоним не может содержать пробелы в начале или возвращать каретки. Кроме того, его можно задать равным строке в памяти, значению числового выражения, содержимому файла, значению переменной среды или выходным данным одной или нескольких команд отладчика.

Имя псевдонима и эквивалент псевдонима чувствительны к регистру.

Чтобы определить или переопределить псевдоним с именем пользователя, используйте команду **(Задать псевдоним)** или **aS (Задать псевдоним).**

Чтобы удалить псевдоним, используйте команду **ad (Delete Alias).**

Чтобы получить список всех текущих псевдонимов с именами пользователей, используйте команду **al (список псевдонимов).**

## Определение псевдонима Fixed-Name

Существует 10 псевдонимов с фиксированным именем. Их псевдонимы: **$u 0**, **$u 1**, ..., **$u 9**. Их эквивалентами псевдонимов может быть любая строка, которая не содержит нажатия клавиши ВВОД.

Используйте команду **r (Registers),** чтобы определить эквиваленты псевдонимов для псевдонимов с фиксированным именем. При определении псевдонима с фиксированным именем необходимо вставить точку (.) перед буквой "u". Текст после знака равенства (=) является эквивалентным псевдонимом. Эквивалент псевдонима может включать пробелы или точки с запятой, но начальные и конечные пробелы игнорируются. Не следует заключать эквивалент псевдонима в кавычки (если в результатах не нужны кавычки).

**Примечание** Не следует путать с помощью команды **r (Registers) для псевдонимов** с фиксированным именем. Эти псевдонимы не являются регистрами или псевдорегистративными регистрами, даже если для задания эквивалентов псевдонимов используется команда **r** . Вам не нужно добавлять знак at (@) перед этими псевдонимами, и вы не можете использовать команду **r** для *отображения* значения одного из этих псевдонимов.

По умолчанию, если псевдоним с фиксированным именем не определен, это пустая строка.

## Автоматические псевдонимы

Отладчик задает следующие автоматические псевдонимы.

| Имя псевдонима | Эквивалент псевдонима |
|---|---|
| **$ntnsym** | Наиболее подходящий модуль для символов NT в собственной архитектуре компьютера. Этот псевдоним может быть равен **ntdll** или **nt**. |
| **$ntwsym** | Наиболее подходящий модуль для символов NT во время 32-разрядной отладки, использующий WOW64. Это может быть **ntdll32** или другая 32-разрядная версия Ntdll.dll. |

| Имя псевдонима | Эквивалент псевдонима |
| --- | --- |
| **$ntsym** | Наиболее подходящий модуль для символов NT, соответствующих текущему режиму компьютера. При отладке в собственном режиме этот псевдоним совпадает с **$ntnsym**. При отладке в не собственном режиме отладчик пытается найти модуль, соответствующий этому режиму. (Например, во время 32-разрядной отладки, которая использует WOW64, этот псевдоним совпадает с **$ntwsym**.) |
| **$CurrentDumpFile** | Имя последнего файла дампа, загруженного отладчиком. |
| **$CurrentDumpPath** | Путь к каталогу последнего файла дампа, загруженного отладчиком. |
| **$CurrentDumpArchiveFile** | Имя последнего файла архива дампа (CAB-файла), загруженного отладчиком. |
| **$CurrentDumpArchivePath** | Путь к каталогу последнего файла архива дампа (CAB-файла), загруженного отладчиком. |

Автоматические псевдонимы аналогичны автоматическим псевдорегистрациям, за исключением того, что можно использовать автоматические псевдонимы с маркерами, связанными с псевдонимами (например **, ${ }**), в то время как с этими маркерами нельзя использовать псевдорегистры.

## Использование псевдонима в командном окне отладчика

После определения псевдонима его можно использовать в любой записи команды. Имя псевдонима автоматически заменяется эквивалентным псевдонимом. Поэтому псевдоним можно использовать в качестве выражения или макроса.

Имя псевдонима правильно расширяется, даже если оно заключено в кавычки. Так как эквивалент псевдонима может содержать любое количество кавычек или точкой с запятой, эквивалентный псевдоним может представлять несколько команд.

Псевдоним пользователя распознается только в том случае, если его имя отделяется от других символов пробелами. Первый символ имени псевдонима должен начинаться с строки или следовать за пробелом, точкой с запятой или

кавычками. Последний символ имени псевдонима должен заканчиваться строкой или следовать пробелом, точкой с запятой или кавычками.

**Примечание** Любой текст, вводимый в окне "Команда отладчика ", начинающийся с "as", "aS", "ad" или "al", не получает замены псевдонима. Это ограничение не позволяет сделать команды псевдонимов неработоспособными. Однако это ограничение также означает, что для команд, следующих за **объявлением** или **al** в строке, псевдонимы не заменяются. Если вы хотите заменить псевдонимы в строке, которая начинается с одной из этих строк, добавьте точку с запятой перед псевдонимом.

Однако маркер **${ }** можно использовать для расширения псевдонима с именем пользователя, даже если он находится рядом с другим текстом. Вы также можете использовать этот маркер вместе с определенными переключателями, чтобы предотвратить расширение псевдонима или отобразить определенные значения, связанные с псевдонимом. Дополнительные сведения об этих ситуациях см. в разделе **${ } (интерпретатор псевдонимов).**

Псевдоним с фиксированным именем правильно расширяется из любой точки строки независимо от того, как он внедрен в текст строки.

Нельзя использовать команды, доступные только в WinDbg (**.open**, **.write_cmd_hist (Write Command History)**,**.lsrcpath** и **.lsrcfix**), а также несколько дополнительных команд (**.hh**, **.cls**, **.wtitle**, **.remote**, **kernel-mode .restart** и **user-mode .restart**) с псевдонимами.

## Использование псевдонима в файле скрипта

При использовании псевдонима в файле скрипта необходимо соблюдать особую осторожность, чтобы обеспечить правильное расширение псевдонима. Рассмотрим следующий сценарий:

text

```
.foreach (value {dd 610000 L4})
{
   as /x ${/v:myAlias} value + 1
   .echo value myAlias
}

ad myAlias
```

При первом выполнении цикла команда **as, aS (Задать псевдоним)** присваивает значение myAlias. Значение, присвоенное myAlias, равно 1 плюс 610000 (первые

выходные данные команды dd). Однако при выполнении команды **.echo (Echo Comment)** myAlias еще не развернута, поэтому вместо 610001 мы видим текст myAlias.

dbgcmd

```
0:001> $$>< c:\Script02.txt
00610000 myAlias
00905a4d 0x610001
00000003 0x905a4e
00000004 0x4
0000ffff 0x5
```

Проблема заключается в том, что myAlias не разворачивается до ввода нового блока кода. Следующая запись цикла — это новый блок, поэтому myAlias развертывается до 610001. Но это слишком поздно: мы должны были видеть, 610001 первый раз через цикл, а не во второй раз. Эту проблему можно устранить, заключив команду **.echo (Echo Comment)** в новый блок, как показано в следующем сценарии.

text

```
.foreach (value {dd 610000 L4})
{
   as /x ${/v:myAlias} value + 1
   .block{.echo value myAlias}
}

ad myAlias
```

При использовании измененного скрипта мы получаем следующие правильные выходные данные.

dbgcmd

```
0:001> $$>< c:\Script01.txt
00610000 0x610001
00905a4d 0x905a4e
00000003 0x4
00000004 0x5
0000ffff 0x10000
```

Дополнительные сведения см. в **разделах .block** и **${ } (интерпретатор псевдонимов).**

## Использование токена FOREACH в псевдониме

При использовании маркера **FOREACH** в определении псевдонима необходимо соблюдать особую осторожность, чтобы обеспечить расширение маркера. Рассмотрим следующую последовательность команд.

dbgcmd

```
r $t0 = 5
ad myAlias
.foreach /pS 2 /ps 2 (Token {?@$t0}) {as myAlias Token}
al
```

Первая команда задает для псевдорегистра **$t 0** значение 5. Вторая команда удаляет все значения, которые могли быть ранее назначены myAlias. Третья команда принимает третий маркер команды **?@$t 0** и пытается назначить значение этого маркера myAlias. Четвертая команда выводит список всех псевдонимов и их значений. Мы ожидаем, что значение myAlias будет равно 5, но вместо этого значение — слово "Токен".

dbgcmd

```
   Alias           Value
 -------         -------
 myAlias         Token
```

Проблема заключается в том, что команда **as** находится в начале строки в теле цикла **FOREACH** . Когда строка начинается с команды **as** , псевдонимы и маркеры в этой строке не расширяются. Если мы поместим точку с запятой или пустое пространство перед командой **as** , то все псевдонимы или маркеры, которые уже имеют значение, будут расширены. В этом примере myAlias не разворачивается, так как у него еще нет значения. Маркер развертывается, так как имеет значение 5. Ниже приведена та же последовательность команд с добавлением точки с запятой перед командой **as** .

dbgcmd

```
r $t0 = 5
ad myAlias
.foreach /pS 2 /ps 2 (Token {?@$t0}) {;as myAlias Token}
al
```

Теперь мы получаем ожидаемые выходные данные.

dbgcmd

```
  Alias            Value
 -------          -------
 myAlias          5
```

## Рекурсивные псевдонимы

Псевдоним с фиксированным именем можно использовать в определении любого псевдонима. Вы также можете использовать псевдоним с именем пользователя в определении псевдонима с фиксированным именем. Однако, чтобы использовать псевдоним с именем пользователя в определении другого псевдонима с именем пользователя, необходимо добавить точку с запятой перед командой **as** или **aS** , иначе замена псевдонима не будет выполнена в этой строке.

При использовании рекурсивных определений этого типа каждый псевдоним преобразуется сразу после его использования. Например, в следующем примере отображается **значение 3**, а не **7**.

dbgcmd

```
0:000> r $.u2=2
0:000> r $.u1=1+$u2
0:000> r $.u2=6
0:000> ? $u1
Evaluate expression: 3 = 00000003
```

Аналогичным образом в следующем примере отображается **значение 3**, а не **7**.

dbgcmd

```
0:000> as fred 2
0:000> r $.u1= 1 + fred
0:000> as fred 6
0:000> ? $u1
Evaluate expression: 3 = 00000003
```

Следующий пример также разрешен и отображает **9**.

dbgcmd

```
0:000> r $.u0=2
0:000> r $.u0=7+$u0
0:000> ? $u0
Evaluate expression: 9 = 00000009
```

## Примеры

Псевдонимы можно использовать, чтобы не вводить длинные или сложные имена символов, как показано в следующем примере.

dbgcmd

```
0:000> as Short usersrv!NameTooLongToWantToType
0:000> dw Short +8
```

Следующий пример аналогичен предыдущему, но в нем используется псевдоним с фиксированным именем.

dbgcmd

```
0:000> r $.u0=usersrv!NameTooLongToWantToType
0:000> dw $u0+8
```

Псевдонимы можно использовать в качестве макросов для часто используемых команд. В следующем примере регистр **eax** и **ebx** регистрируются в два раза.

dbgcmd

```
0:000> as GoUp r eax=eax+1; r ebx=ebx+1
0:000> GoUp
0:000> GoUp
```

В следующем примере используется псевдоним для упрощения ввода команд.

dbgcmd

```
0:000> as Cmd "dd esp 14; g"
0:000> bp MyApi Cmd
```

Следующий пример аналогичен предыдущему, но в нем используется псевдоним с фиксированным именем.

dbgcmd

```
0:000> r $.u5="dd esp 14; g"
0:000> bp MyApi $u5
```

Оба предыдущих примера эквивалентны следующей команде.

dbgcmd

```
0:000> bp MyApi "dd esp 14; g"
```

## Файл Tools.ini

В CDB (и NTSD) можно предварительно определить псевдонимы с фиксированным именем в файлетools.ini . Чтобы предопределить псевдоним с фиксированным именем, добавьте **$u** поля, которые вы хотите, в запись [NTSD], как показано в следующем примере.

ini

```
[NTSD]
$u1:_ntdll!_RtlRaiseException
$u2:"dd esp 14;g"
$u9:$u1 + 42
```

В файле Tools.ini нельзя задать псевдонимы с именами пользователей.

## Псевдонимы с фиксированным именем и псевдонимы User-Named

Псевдонимы имен пользователей проще использовать, чем псевдонимы с фиксированным именем. Синтаксис их определения проще, и их можно получить с помощью команды **al (list aliases).**

Псевдонимы с фиксированными именами заменяются, если они используются рядом с другим текстом. Чтобы заменить псевдоним с именем пользователя, если он находится рядом с другим текстом, заключите его в маркер **${ } (интерпретатор псевдонимов).**

Замена псевдонима с фиксированным именем происходит до замены псевдонима с именем пользователя.

# Использование файлов скриптов

Статья • 08.07.2024

*Файл скрипта — это текстовый файл*, содержащий последовательность команд отладчика. Для отладчика можно загрузить файл скрипта и выполнить его различными способами. Файл скрипта может содержать команды для последовательного выполнения или использовать более сложный поток выполнения.

Чтобы выполнить файл скрипта, можно выполнить одно из следующих действий:

- (только KD и CDB; только при запуске отладчика) Создайте файл скрипта с именем Ntsd.ini и поместите его в каталог, из которого запускается отладчик. Отладчик автоматически выполняет этот файл при запуске отладчика. Чтобы использовать другой файл для файла скрипта запуска, укажите путь и имя файла с помощью **параметра командной строки -cf** или с помощью **записи IniFile** в файле Tools.ini.

- (только KD и CDB; при запуске каждого сеанса) Создайте файл скрипта и укажите путь и имя файла с помощью **параметра командной строки -cfr**. Отладчик автоматически выполняет этот файл скрипта при запуске отладчика и при каждом перезапуске целевого объекта.

- **Используйте команды $, $<, $><, $$<** и **$$><** для выполнения файла скрипта после запуска отладчика. Дополнительные сведения о синтаксисе см. в разделе $, $, $><, $><, $$>< (файл скрипта run).<

**Команды $><** и **$$><** отличаются от других методов выполнения скриптов одним важным способом. При использовании этих команд отладчик открывает указанный файл скрипта, заменяет все каретки с запятой и выполняет результирующий текст в виде одного блока команд. Эти команды полезны для выполнения скриптов, содержащих программы команд отладчика. Дополнительные сведения об этих программах см. в разделе "Использование программ команд отладчика". X

Команды, доступные только в WinDbg (например, **lsrcfix (использование локального исходного сервера)**,**lsrcpath (set Local Source Path)**, **.open (Open Source File)**и **.write_cmd_hist (Write Command History)**) в файлах скриптов, даже если файл скрипта выполняется в WinDbg. Кроме того, вы не можете использовать **.beep (beep)**, .cls (Clear Screen), **.hh (Open HTML Help File)**, **.idle_cmd (Set Idle Command)**, **.remote (Create Remote.exe Server)**, **kernel-mode .restart (Restart Kernel Connection)**,

**user-mode .restart (Restart Target Application**) or **.wtitle (Set Window Title) (Set Window Title)** команды в файле скрипта.

WinDbg поддерживает те же сценарии, что и KD и CDB, с одним незначительным исключением. Команду .remote_exit (выход из клиента отладки) **можно использовать** только в файле скрипта, который использует KD или CDB. Вы не можете выйти из клиента отладки, хотя скрипт, выполняемый в WinDbg.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Использование расширений отладчика

Статья • 26.10.2023

WinDbg, KD и CDB позволяют использовать команды расширения отладчика. Эти расширения предоставляют этим трем отладчикам Майкрософт большую степень мощности и гибкости.

Команды расширения отладчика используются так же, как и стандартные команды отладчика. Однако, хотя встроенные команды отладчика сами являются частью двоичных файлов отладчика, команды расширения отладчика предоставляются библиотеками DLL, отличными от отладчика.

Это позволяет создавать новые команды отладчика, адаптированные к конкретным потребностям. Кроме того, ряд библиотек DLL расширения отладчика поставляется вместе с самими средствами отладки.

Этот раздел состоит из следующих частей.

Загрузка библиотек DLL расширения отладчика

Использование команд расширения отладчика

Создание новых расширений отладчика

# Загрузка DLL расширений для отладки

Статья • 07.04.2025

Существует несколько методов загрузки библиотек DLL расширений отладчика, а также управления библиотекой DLL расширения отладчика по умолчанию и пути расширения отладчика по умолчанию:

- (Перед запуском отладчика) Используйте переменную среды _NT_DEBUGGER_EXTENSION_PATH, чтобы задать путь по умолчанию для библиотек DLL расширения. Это может быть ряд путей к каталогу, разделенных точкой с запятой.
- Используйте команду **load (load Extension DLL),** чтобы загрузить новую библиотеку DLL.
- Используйте команду **.unload (Выгрузка расширения DLL)**, чтобы выгрузить библиотеку DLL.
- Чтобы выгрузить все расширения отладчика, используйте команду **.unloadall (Выгрузка всех файлов DLL расширений)**.
- (Перед запуском отладчика; Только CDB) Используйте файлtools.ini , чтобы задать библиотеку DLL расширения по умолчанию.
- (Перед запуском отладчика) Используйте **-a**параметр командной строки для задания библиотеки DLL расширения по умолчанию.
- Используйте команду **extpath (Set Extension Path)** для задания пути поиска библиотеки расширения DLL.
- Используйте команду **.setdll (Set Default Extension DLL)** для установки библиотеки расширения DLL по умолчанию.
- Используйте команду **chain (Расширения отладчика)**, чтобы отобразить все загруженные модули расширений отладчика в их порядке поиска по умолчанию.

Вы также можете загрузить расширение DLL, просто использовав полный синтаксис **!***module.extension* при первом выполнении команды из этого модуля. Дополнительные сведения см. в разделе "Использование команд расширения отладчика ".

Библиотеки DLL расширения, которые вы используете, должны соответствовать операционной системе целевого компьютера. Библиотеки DLL расширения, которые поставляются с пакетом средств отладки для Windows, размещаются в разных подкаталогах директории установки:

- Каталог winxp содержит расширения, которые можно использовать с Windows XP и более поздними версиями Windows.
- Каталог winext содержит расширения, которые можно использовать с любой версией Windows. Модуль dbghelp.dll, расположенный в базовом каталоге средств отладки для Windows, также содержит расширения этого типа.

Если вы пишете собственные расширения отладчика, их можно разместить в любом каталоге. Однако рекомендуется поместить их в новый каталог и добавить этот каталог в путь расширения отладчика.

Могут быть загружены до 32 библиотек расширения DLL.

# Использование команд расширения отладчика

Статья • 08.07.2024

Использование команд расширения отладчика очень похоже на использование команд отладчика. Команда вводится в командном окне отладчика, создавая выходные данные в этом окне или изменение целевого приложения или целевого компьютера.

Фактическая команда расширения отладчика — это точка входа в библиотеке DLL, вызываемой отладчиком.

Расширения отладчика вызываются следующим синтаксисом:

**!** [*модуль***.**]*extension* [*аргументы*]

Имя модуля не должно следовать с расширением имени файла .dll. Если *модуль* содержит полный путь, ограничение размера строки по умолчанию составляет 255 символов.

Если модуль еще не загружен, он будет загружен в отладчик с помощью вызова **LoadLibrary**(*module*). После загрузки библиотеки расширений отладчик вызывает **функцию GetProcAddress** , чтобы найти имя расширения в модуле расширения. Имя расширения учитывает регистр и должно быть введено точно так же, как оно отображается в def-файле модуля расширения. Если адрес расширения найден, вызывается расширение.

## Порядок поиска

Если имя модуля не указано, отладчик будет искать загруженные модули расширения для этого экспорта.

Порядок поиска по умолчанию выглядит следующим образом:

1. Модули расширения, работающие со всеми операционными системами и в обоих режимах: Dbghelp.dll и winext\ext.dll.
2. Модуль расширения, работающий во всех режимах, но зависящий от операционной системы. Для Windows XP и более поздних версий Windows это winxp\exts.dll.

3. Модуль расширения, работающий со всеми операционными системами, но зависящий от режима. Для режима ядра это winext\kext.dll. Для пользовательского режима это winext\uext.dll.

4. Модуль расширения, предназначенный как для операционной системы, так и для конкретного режима. В следующей таблице указывается этот модуль.

Развернуть таблицу

| Режим пользователя | Режим ядра |
|---|---|
| winxp \ ntsdexts.dll | winxp \ kdexts.dll |

Когда модуль расширения выгружается, он удаляется из цепочки поиска. При загрузке модуля расширения он добавляется в начало порядка поиска. **Команда .setdll (Set Default Extension DLL)** может использоваться для повышения уровня любого модуля в верхней части цепочки поиска. С помощью этой команды можно полностью управлять цепочкой поиска.

**Используйте команду .chain (Расширения отладчика списка)** для отображения списка всех загруженных модулей расширений в текущем порядке поиска.

Если вы попытаетесь выполнить команду расширения, которая не находится ни в одном из загруженных модулей расширения, вы получите сообщение об ошибке export Not Found.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Создание новых расширений отладчика

Статья • 26.10.2023

Вы можете создать собственные команды отладки, написав библиотеку DLL расширения. Например, может потребоваться написать команду для отображения сложной структуры данных или команду, которая остановит и запустит целевой объект в зависимости от значения определенных переменных или расположений памяти.

Существует два разных типа расширений отладчика:

- *Расширения DbgEng*. Они основаны на прототипах в файле заголовка dbgeng.h, а также в файле заголовка wdbgexts.h.
- *Расширения WdbgExts*. Они основаны только на прототипах в файле заголовка wdbgexts.h.

Сведения о написании расширений отладчика см. в разделах Написание расширений DbgEng и Написание расширений WdbgExts.

# Команды отладчика

Статья • 27.03.2025

В этом разделе приведены следующие разделы:

Правила синтаксиса

маркеры команд

Команды

метакоманд

ключи управления

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Syntax Rules

18.07.2025

В этом разделе описаны правила синтаксиса, которые необходимо выполнить для использования команд отладчика.

При отладке следует соблюдать следующие общие правила синтаксиса:

- Вы можете использовать любое сочетание прописных и строчных букв в командах и аргументах, за исключением случаев, когда они были указаны в разделах этого раздела.
- You can separate multiple command parameters by one or more spaces or by a comma (,).
- Обычно пробел между командой и его первым параметром можно опустить. Часто можно опустить другие пробелы, если это упущение не вызывает неоднозначности.

Справочные разделы команды в этом разделе используют следующие элементы:

- Characters in **bold** font style indicate items that you must literally type.
- Characters in *italic* font style indicate parameters that are explained in the "Parameters" section of the reference topic.
- Parameters in brackets ([*xxx*]) are optional. Brackets with a vertical bar ([*xxx*|*yyy*]) indicate that you can use one, or none, of the enclosed parameters.
- Braces with vertical bars ({*xxx*|*yyy*}) indicate that you must use exactly one of the enclosed parameters .

В следующих разделах описан синтаксис, используемый следующими типами параметров:

Синтаксис числовых выражений

Синтаксис подстановочных знаков строки

Register Syntax

Pseudo-Register Syntax

Синтаксис исходной строки

Синтаксис диапазона адресов и адресов

Thread Syntax

Process Syntax

System Syntax

Multiprocessor Syntax

Синтаксис также играет важную роль в использовании символов. Дополнительные сведения см. в разделе "Синтаксис символов" и "Сопоставление символов".

# Синтаксис числовых выражений

18.07.2025

Отладчик принимает два различных типа числовых выражений: *выражения C++* и *выражения MASM*. Каждое из этих выражений соответствует собственным правилам синтаксиса входных и выходных данных.

Дополнительные сведения об использовании каждого типа синтаксиса см. в разделе "Оценка выражений".

Используйте команду EXPR (Выбор вычислителя выражений), чтобы отобразить или изменить средство оценки выражений после запуска отладчика.

В этом разделе приведены следующие разделы:

Номера и операторы MASM

Номера и операторы C++

Выражения MASM и выражения C++

Примеры смешанных выражений

Расширение знака

? (Оценить выражение)

.expr (Выберите вычислитель выражений)

# Номера и операторы MASM

18.07.2025

В этом разделе описывается использование синтаксиса выражения макросов Майкрософт (MASM) с инструментами отладки Windows.

Отладчик принимает два различных типа числовых выражений: выражения C++ и выражения MASM. Каждое из этих выражений соответствует собственным правилам синтаксиса входных и выходных данных.

Дополнительные сведения об использовании каждого типа синтаксиса см. в разделе "Оценка выражений " и ? (Вычислять выражение).

В этом примере используется ? Команда отображает значение регистра указателя инструкции с помощью вычислителя выражений MASM.

dbgcmd

```
0:000> ? @rip
Evaluate expression: 140709230544752 = 00007ff9`6bb40770
```

# Установка средства оценки выражений в MASM

Используйте expr (выбор вычислителя выражений), чтобы узнать, что такое средство оценки выражений по умолчанию, и измените его на MASM.

dbgcmd

```
0:000> .expr /s masm
Current expression evaluator: MASM - Microsoft Assembler expressions
```

Теперь, когда средство оценки выражений по умолчанию было изменено, команду ? (Вычислять выражение) можно использовать для отображения выражений MASM. В этом примере в регистр rip добавляется шестнадцатеричное значение 8.

dbgcmd

```
0:000> ? @rip + 8
Evaluate expression: 140709230544760 = 00007ff9`6bb40778
```

Ссылка на регистр @rip описана более подробно в описании синтаксиса регистра.

# Числа в выражениях отладчика MASM

В выражения MASM можно поместить числа в любом из оснований 16, 10, 8 или 2.

Используйте команду **n (Set Number Base),** чтобы задать для радикса по умолчанию значение 16, 10 или 8. Затем все числа, не имеющие префикса, интерпретируются в этой базе. Можно переопределить радикс по умолчанию, указав префикс **0x** (шестнадцатеричный), **префикс 0n** (десятичный), префикс **0t** (восьмерично) или **0y** префикс (двоичный).

Вы также можете указать шестнадцатеричные числа, добавив **h** после числа. В числах можно использовать прописные или строчные буквы. Например, "0x4AB3", "0X4aB3", "4AB3h", "4ab3h" и "4aB3H" имеют то же значение.

Если вы не добавляете число после префикса в выражении, число считывается как 0. Таким образом, можно написать значение 0 как 0, префикс, за которым следует 0, и только префикс. Например, в шестнадцатеричном, "0", "0x0" и "0x" имеют то же значение.

Шестнадцатеричные 64-разрядные значения можно ввести в формате **xxxx'xxxx.** Вы также можете опустить серьезный акцент ('). Если включить серьезный акцент, автоматическое расширение знака отключено.

В этом примере показано, как добавить десятичное, восьмеричное и двоичное значение для регистрации 10.

dbgcmd

```
? @r10 + 0x10 + 0t10 + 0y10
Evaluate expression: 26 = 00000000`0000001a
```

# Символы в выражениях MASM отладчика

В выражениях MASM числовое значение любого символа является его адресом памяти. В зависимости от того, к какому символу относится, этот адрес может быть адресом глобальной переменной, локальной переменной, функции, сегмента, модуля или любой другой распознанной метки.

Чтобы указать, с каким модулем связан адрес, добавьте имя модуля и восклицательный знак (!) перед именем символа. Если символ может быть интерпретирован как шестнадцатеричное число, добавьте имя модуля и восклицательный знак или просто восклицательный знак перед именем символа. Дополнительные сведения о распознавании символов см. в разделе "Синтаксис символов" и "Сопоставление символов".

Используйте две двоеточия (::) или два символа подчеркивания (__), чтобы указать члены класса.

Используйте серьезный акцент (') или апостроф (') в имени символа, только если добавить имя модуля и восклицательный знак перед символом.

# Числовые операторы в выражениях MASM

Вы можете изменить любой компонент выражения с помощью унарного оператора. Можно объединить любые два компонента с помощью двоичного оператора. Унарные операторы имеют приоритет над двоичными операторами. При использовании нескольких двоичных операторов операторы следуют фиксированным правилам приоритета, описанным в следующих таблицах.

Для переопределения правил приоритета всегда можно использовать круглые скобки.

Если часть выражения MASM заключена в скобки и два знака (@@) отображаются перед выражением, выражение интерпретируется в соответствии с правилами выражений C++. Нельзя добавить пробел между двумя знаками @ и открывающей скобкой. Вы также можете указать средство оценки выражений с помощью **@@c++( ... )** или **@@masm( ... )**.

При выполнении арифметических вычислений средство оценки выражений MASM обрабатывает все числа и символы как ULONG64 типы.

Унарные операторы адресов предполагают ds в качестве сегмента по умолчанию для адресов. Выражения оцениваются в порядке приоритета оператора. Если смежные операторы имеют равный приоритет, выражение вычисляется слева направо.

Вы можете использовать следующие унарные операторы.

Развернуть таблицу

| Оператор | Значение |
|---|---|
| + | Унарный плюс |
| - | Унарный минус |
| не | Возвращает значение 1, если аргумент равен нулю. Возвращает ноль для любого ненулевого аргумента. |
| **Привет** | Высокий 16 бит |
| **низкий** | Низкий 16 бит |

| Оператор | Значение |
| --- | --- |
| **от** | Байт низкого порядка из указанного адреса. |
| **$pby** | То **же самое,** что и за исключением того, что он принимает физический адрес. Можно считывать только физическую память, использующую поведение кэширования по умолчанию. |
| **wo** | Слово с низким порядком из указанного адреса. |
| **$pwo** | То же самое, что **wo** , за исключением того, что он принимает физический адрес. Можно считывать только физическую память, использующую поведение кэширования по умолчанию. |
| **dwo** | Двойное слово из указанного адреса. |
| **$pdwo** | То же самое, что **dwo** , за исключением того, что он принимает физический адрес. Можно считывать только физическую память, использующую поведение кэширования по умолчанию. |
| **qwo** | Четыре слова из указанного адреса. |
| **$pqwo** | То же, что **и qwo** за исключением того, что она принимает физический адрес. Можно считывать только физическую память, использующую поведение кэширования по умолчанию. |
| **poi** | Данные размера указателя из указанного адреса. Размер указателя составляет 32 бита или 64 бита. В отладке ядра этот размер основан на процессоре *целевого* компьютера. Таким образом, **poi** является лучшим оператором для использования, если требуются данные размером с указатель. |
| **$ppoi** | То же, что **poi**, но с использованием физического адреса. Можно считывать только физическую память, использующую поведение кэширования по умолчанию. |

## Примеры

В следующем примере показано, как использовать poi для разыменования указателя, чтобы увидеть значение, хранящееся по этому адресу памяти.

Сначала определите адрес памяти, интересующий вас. Например, можно просмотреть структуру потока и решить, что нужно увидеть значение CurrentLocale.

dbgcmd

```
0:000> dx @$teb
@$teb                 : 0x8eed57b000 [Type: _TEB *]
    [+0x000] NtTib            [Type: _NT_TIB]
    [+0x038] EnvironmentPointer : 0x0 [Type: void *]
    [+0x040] ClientId         [Type: _CLIENT_ID]
    [+0x050] ActiveRpcHandle  : 0x0 [Type: void *]
    [+0x058] ThreadLocalStoragePointer : 0x1f8f9d634a0 [Type: void *]
    [+0x060] ProcessEnvironmentBlock : 0x8eed57a000 [Type: _PEB *]
    [+0x068] LastErrorValue   : 0x0 [Type: unsigned long]
    [+0x06c] CountOfOwnedCriticalSections : 0x0 [Type: unsigned long]
    [+0x070] CsrClientThread  : 0x0 [Type: void *]
    [+0x078] Win32ThreadInfo  : 0x0 [Type: void *]
    [+0x080] User32Reserved   [Type: unsigned long [26]]
    [+0x0e8] UserReserved     [Type: unsigned long [5]]
    [+0x100] WOW32Reserved    : 0x0 [Type: void *]
    [+0x108] CurrentLocale    : 0x409 [Type: unsigned long]
```

CurrentLocale находится 0x108 за пределами начала TEB.

dbgcmd

```
0:000> ? @$teb + 0x108
Evaluate expression: 613867303176 = 0000008e`ed57b108
```

Используйте poi для разыменования этого адреса.

dbgcmd

```
0:000> ? poi(0000008e`ed57b108)
Evaluate expression: 1033 = 00000000`00000409
```

Возвращаемое значение 409 соответствует значению CurrentLocale в структуре TEB.

Или используйте poi и круглые скобки для расшифровки вычисляемого адреса.

dbgcmd

```
0:000> ? poi(@$teb + 0x108)
Evaluate expression: 1033 = 00000000`00000409
```

Используйте унарные операторы *by* или *wo*, чтобы вернуть байт или слово из целевого адреса.

dbgcmd

```
0:000> ? by(0000008e`ed57b108)
Evaluate expression: 9 = 00000000`00000009
0:000> ? wo(0000008e`ed57b108)
Evaluate expression: 1033 = 00000000`00000409
```

# Двоичные операторы

Вы можете использовать следующие двоичные операторы. Операторы в каждой ячейке имеют приоритет над операторами в нижних ячейках. Операторы в одной ячейке имеют одинаковый приоритет и анализируются слева направо.

Развернуть таблицу

| Оператор | Значение |
|---|---|
| * | Умножение |
| / | Деление целых чисел |
| **mod** (или %) | Модуль (оставшаяся часть) |
| + | Дополнение |
| - | Вычитание |
| << | Сдвиг влево |
| >> | Логическая смена вправо |
| >>> | Арифметический сдвиг вправо |
| = (или ==) | Равняется |
| < | Меньше |
| > | Больше чем |
| <= | Меньше или равно |
| >= | Больше или равно |
| != | Не равно |
| **и** (или &) | Битовое И |
| **xor** (или ^) | Побитовое XOR (исключающее ИЛИ) |
| **или** (\|) | Битовое ИЛИ |

Операторы <, >, =, ==, и != оцениваются как 1, если выражение истинно, или как 0, если выражение ложно. Единый знак равенства (=) совпадает с двойным знаком равенства (==). Побочные эффекты или назначения нельзя использовать в выражении MASM.

Недопустимая операция (например, деление по нулю) приводит к возврату ошибки Operand в окно команды отладчика.

Мы можем проверить, соответствует ли возвращаемое значение 0x409 с помощью оператора сравнения == .

dbgcmd

```
0:000> ? poi(@$teb + 0x108)==0x409
Evaluate expression: 1 = 00000000`00000001
```

# Нечисловые операторы в выражениях MASM

Кроме того, в выражениях MASM можно использовать следующие дополнительные операторы.

Развернуть таблицу

| Оператор | Значение |
|---|---|
| **$fnsucc**(*FnAddress*, *RetVal*, *Flag*) | Интерпретирует значение *RetVal* как возвращаемое значение для функции, расположенной по адресу *FnAddress* . Если это возвращаемое значение квалифициируется как код успешного выполнения, **$fnsucc** возвращает **значение TRUE**. В противном случае **$fnsucc** возвращает **значение FALSE**.<br><br>Если возвращаемый тип — BOOL, bool, HANDLE, HRESULT или NTSTATUS, **$fnsucc** правильно понимает, является ли указанное возвращаемое значение кодом успеха. Если возвращаемый тип является указателем, все значения, отличные от **NULL** , соответствуют кодам успешности. Для любого другого типа успех определяется значением *Flag*. Если *Флаг* равен 0, ненулевое значение *RetVal* означает успех. Если *флаг* равен 1, значение *RetVal* равно нулю — это успех. |
| **$iment** (*адрес*) | Возвращает адрес точки входа изображения в загруженном списке модулей. *Адрес* указывает базовый адрес переносимого исполняемого |

| Оператор | Значение |
| --- | --- |
| | файла (PE). Запись найдена путем поиска точки входа изображения в заголовке изображения PE, указанного *адресом* .<br><br>Эту функцию можно использовать для обоих модулей, которые уже находятся в списке модулей, и для установки неразрешенных точек останова с помощью команды **bu** . |
| **$scmp**("*String1*", "*String2*") | Вычисляется в -1, 0 или 1, подобно функции **strcmp** с помощью C-функции **strcmp**. |
| **$sicmp**("*String1*", "*String2*") | Оценивается как -1, 0 или 1, так же, как и функция **stricmp** в Microsoft Win32. |
| **$spat**("*String*", "*Pattern*") | Вычисляется как **TRUE** или **FALSE** в зависимости от того, соответствует ли *Строкашаблону*. Регистр при сопоставлении не учитывается. *Шаблон* может содержать различные подстановочные знаки и спецификаторы. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис подстановочных знаков строки". |
| **$vvalid**(*адрес*, *длина*) | Определяет, является ли диапазон памяти, начинающийся с *адрес* и протяженностью *длиной* байт, допустимым. Если память действительна, **$vvalid** принимает значение 1. Если память недействительна, **$vvalid** принимает значение 0. |

## Примеры

Ниже показано, как исследовать диапазон допустимой памяти вокруг загруженного модуля.

Сначала определите адрес интересующей области, например с помощью команды lm (команда List Loaded Modules ).

dbgcmd

```
0:000> lm
start             end                 module name
00007ff6`0f620000 00007ff6`0f658000   notepad    (deferred)
```

```
00007ff9`591d0000 00007ff9`5946a000   COMCTL32   (deferred)
...
```

Используйте $vvalid для проверки диапазона памяти.

dbgcmd

```
0:000> ? $vvalid(0x00007ff60f620000, 0xFFFF)
Evaluate expression: 1 = 00000000`00000001
```

Используйте $vvalid для подтверждения того, что этот большой диапазон является недопустимым диапазоном памяти.

dbgcmd

```
0:000> ? $vvalid(0x00007ff60f620000, 0xFFFFF)
Evaluate expression: 0 = 00000000`00000000
```

Это также недопустимый диапазон.

dbgcmd

```
0:000> ? $vvalid(0x0, 0xF)
Evaluate expression: 0 = 00000000`00000000
```

Используйте *не* для возврата нуля, если диапазон памяти действителен.

dbgcmd

```
0:000> ? not($vvalid(0x00007ff60f620000, 0xFFFF))
Evaluate expression: 0 = 00000000`00000000
```

Используйте $imnet, чтобы просмотреть точку входа COMCTL32, адрес которой мы ранее определили с помощью команды lm. Она начинается с 0007ff9'591d00000.

dbgcmd

```
0:000> ? $iment(00007ff9`591d0000)
Evaluate expression: 140708919287424 = 00007ff9`59269e80
```

Дизассемблируйте возвращенный адрес для проверки кода точки входа.

dbgcmd

```
0:000> u 00007ff9`59269e80
COMCTL32!DllMainCRTStartup:
00007ff9`59269e80 48895c2408      mov     qword ptr [rsp+8],rbx
00007ff9`59269e85 4889742410      mov     qword ptr [rsp+10h],rsi
00007ff9`59269e8a 57              push    rdi
```

COMCTL32 отображается в выходных данных, подтверждающих, что это точка входа для этого модуля.

# Регистр и Pseudo-Registers в выражениях MASM

В выражениях MASM можно использовать регистры и псевдорегистрации. Вы можете добавить знак (@) перед всеми регистрами и псевдорегистрами. Знак @ позволяет отладчику быстрее получать доступ к значению. Этот знак @не требуется для наиболее распространенных регистров на основе x86. Для других регистров и псевдорегистрирующих регистров рекомендуется добавить знак при входе, но на самом деле это не обязательно. Если вы опустите знак @ для реже используемых регистров, отладчик пытается сначала проанализировать текст как шестнадцатеричное число, затем как символ, и, наконец, как регистр.

Можно также использовать точку (.) для указания текущего указателя команд. Не следует добавлять знак @до этого периода, и вы не можете использовать период в качестве первого параметра **команды r**. Этот период времени обозначает то же, что и **$ip** псевдорегистр.

Дополнительные сведения о регистрах и псевдорегистрах см. в Синтаксисе регистров и СинтаксисеPseudo-Register.

Используйте команду r register, чтобы узнать, что значение регистра @rip равно 00007ffb'7ed00770.

dbgcmd

```
0:000> r
rax=0000000000000000 rbx=0000000000000010 rcx=00007ffb7eccd2c4
rdx=0000000000000000 rsi=00007ffb7ed61a80 rdi=00000027eb6a7000
rip=00007ffb7ed00770 rsp=00000027eb87f320 rbp=0000000000000000
 r8=00000027eb87f318  r9=0000000000000000 r10=0000000000000000
r11=0000000000000246 r12=0000000000000040 r13=0000000000000000
r14=00007ffb7ed548f0 r15=00000210ea090000
iopl=0         nv up ei pl zr na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!LdrpDoDebuggerBreak+0x30:
00007ffb`7ed00770 cc              int     3
```

Это же значение можно отобразить с помощью . ярлык периода.

dbgcmd

```
0:000> ? .
Evaluate expression: 140718141081456 = 00007ffb`7ed00770
```

Мы можем подтвердить, что эти значения эквивалентны и возвращают ноль, если они есть, используя это выражение MASM.

dbgcmd

```
0:000>  ? NOT(($ip = .) AND ($ip = @rip) AND (@rip =. ))
Evaluate expression: 0 = 00000000`00000000
```

# Исходные номера строк в выражениях MASM

В выражениях MASM вы можете использовать выражения исходных файлов и номеров строк. Эти выражения необходимо заключить с помощью серьезных акцентов (`). Дополнительные сведения о синтаксисе см. в разделе "Синтаксис исходной строки".

# См. также

Выражения MASM и выражения C++

Примеры смешанных выражений

Номера и операторы C++

Расширение знака

? (Оценить выражение)

# Номера и операторы C++

18.07.2025

В этой статье описывается использование синтаксиса выражений C++ с средствами отладки Windows.

Отладчик принимает два различных типа числовых выражений: выражения C++ и выражения сборщика макросов Майкрософт (MASM). Каждое из этих выражений соответствует собственным правилам синтаксиса входных и выходных данных.

Дополнительные сведения об использовании каждого типа синтаксиса см. в статье "Оценка выражений" и команда "Оценить выражение".

Средство синтаксического анализа выражений C++ поддерживает все формы синтаксиса выражений C++. Синтаксис включает все типы данных, включая указатели, числа с плавающей запятой и массивы, а также все унарные и двоичные операторы C++.

Окна "*Контрольные* " и "*Локальные*" в отладчике всегда используют средство оценки выражений C++.

В следующем примере команда ?? evaluate C++ expression отображает значение регистра указателя команд.

dbgcmd

```
0:000> ?? @eip
unsigned int 0x771e1a02
```

Для определения размера структур можно использовать функцию C++ `sizeof` .

dbgcmd

```
0:000> ?? (sizeof(_TEB))
unsigned int 0x1000
```

# Задайте для вычислителя выражений значение C++

Используйте оценщик .expr choose expression, чтобы увидеть оценщик выражений по умолчанию и изменить его на C++.

dbgcmd

```
0:000> .expr
Current expression evaluator: MASM - Microsoft Assembler expressions
0:000> .expr /s c++
Current expression evaluator: C++ - C++ source expressions
```

После изменения вычислителя выражений по умолчанию можно использовать команду для отображения выражений C++. В следующем примере отображается значение регистра указателя инструкции.

dbgcmd

```
0:000> ? @eip
Evaluate expression: 1998461442 = 771e1a02
```

Дополнительные сведения о ссылке на регистр `@eip` см. в разделе Синтаксис регистра.

В этом примере шестнадцатеричное значение 0xD добавляется в регистр eip.

dbgcmd

```
0:000> ? @eip + 0xD
Evaluate expression: 1998461455 = 771e1a0f
```

# Регистры и псевдорегистры в выражениях C++

В выражениях C++ можно использовать регистры и псевдорегистры. Перед регистрацией или псевдорегистром необходимо добавить знак @.

Средство оценки выражений автоматически выполняет правильный приведение. Фактические регистры и псевдорегистры с целочисленным значением приводятся к `ULONG64`. Все адреса приведены к `PUCHAR`, `$thread` приведен к `ETHREAD*`, `$proc` приведен к `EPROCESS*`, `$teb` приведен к `TEB*`, и `$peb` приведен к `PEB*`.

В этом примере отображается TEB.

dbgcmd

```
0:000>  ?? @$teb
struct _TEB * 0x004ec000
   +0x000 NtTib            : _NT_TIB
   +0x01c EnvironmentPointer : (null)
   +0x020 ClientId         : _CLIENT_ID
   +0x028 ActiveRpcHandle  : (null)
   +0x02c ThreadLocalStoragePointer : 0x004ec02c Void
   +0x030 ProcessEnvironmentBlock : 0x004e9000 _PEB
```

```
    +0x034 LastErrorValue   : 0xbb
    +0x038 CountOfOwnedCriticalSections : 0
```

Нельзя изменить регистр или псевдорегистр с помощью оператора присваивания или побочного эффекта. Для изменения этих значений необходимо использовать команду r registers .

В следующем примере устанавливается псевдорегистр на значение 5, после чего это значение отображается.

dbgcmd

```
0:000> r $t0 = 5

0:000> ?? @$t0
unsigned int64 5
```

Дополнительные сведения о регистрах и псевдорегистрирующих регистрах см. в разделе Register синтаксис и синтаксис псевдорегистрирующего регистра.

# Числа в выражениях C++

Числа в выражениях C++ интерпретируются как десятичные числа, если они не указаны другим способом. Чтобы указать шестнадцатеричное целое число, добавьте 0x перед числом. Чтобы указать восьмеричное целое число, добавьте 0 (ноль) перед числом.

Радикс отладчика по умолчанию не влияет на способ ввода выражений C++. Вы не можете напрямую ввести двоичное число, кроме как когда вы вкладываете выражение MASM в выражение C++.

Шестнадцатеричное 64-разрядное значение можно ввести в формате xxxx'xxxx. Вы также можете опустить серьезный акцент ('). Оба формата создают одно и то же значение.

Можно использовать суффиксы `L`, `U` и `I64` с целыми значениями. Фактический размер создаваемого числа зависит от суффикса и введенного числа. Дополнительные сведения об этой интерпретации см. в справочнике по языку C++.

*Выходные данные* вычислителя выражений C++ сохраняют тип данных, указанный правилами выражений C++. Однако при использовании этого выражения в качестве аргумента для команды всегда выполняется приведение. Например, вам не нужно приведение целых значений к указателям, когда они используются в качестве адресов в аргументах команд. Если значение выражения не может быть допустимо приведено к целому числу или указателю, возникает синтаксическая ошибка.

Префикс `0n` (десятичный) можно использовать для некоторых *выходных данных*, но он не подходит для ввода выражений на C++.

## Символы и строки в выражениях C++

Вы можете ввести символ, окружив его одними кавычками ('). Разрешены стандартные escape-символы C++.

Строковые литералы можно вводить, окружая их двойными кавычками ("). Вы можете использовать \" в качестве escape-последовательности в такой строке. Однако строки не имеют значения для вычислителя выражений.

## Символы в выражениях C++

В выражении C++ каждый символ интерпретируется в соответствии с типом. В зависимости от того, к чему относится символ, он может интерпретироваться как целое число, структура данных, указатель функции или любой другой тип данных. Синтаксическая ошибка возникает, если используется символ, который не соответствует типу данных C++ ( например, немодифицированному имени модуля) в выражении C++ .

Вы можете использовать серьезный акцент (') или апостроф (') в имени символа, только если добавить имя модуля и восклицательный знак перед именем символа. При добавлении разделителей < и > после имени шаблона можно добавлять пробелы между этими разделителями.

Если символ может быть неоднозначным, можно добавить имя модуля и восклицательный знак (!) или только восклицательный знак перед символом. Чтобы указать, что символ должен быть локальным, опустить имя модуля и включить знак доллара и восклицательный знак ($!) перед именем символа. Дополнительные сведения о распознавании символов см. в разделе "Синтаксис символов" и "Сопоставление символов".

## Структуры в выражениях C++

Оценщик выражений C++ приводит псевдорегистры к соответствующим типам. Например, `$teb` преобразуется в `TEB*`.

dbgcmd

```
0:000> ??  @$teb
struct _TEB * 0x004ec000
   +0x000 NtTib            : _NT_TIB
```

```
   +0x01c EnvironmentPointer : (null)
   +0x020 ClientId         : _CLIENT_ID
   +0x028 ActiveRpcHandle  : (null)
   +0x02c ThreadLocalStoragePointer : 0x004ec02c Void
   +0x030 ProcessEnvironmentBlock : 0x004e9000 _PEB
   +0x034 LastErrorValue   : 0xbb
   +0x038 CountOfOwnedCriticalSections : 0
```

В следующем примере отображается идентификатор процесса в структуре TEB, показывающий использование указателя на элемент ссылочной структуры.

dbgcmd

```
0:000> ??  @$teb->ClientId.UniqueProcess
void * 0x0000059c
```

# Операторы в выражениях C++

Скобки можно использовать для переопределения правил приоритета.

Если вы заключаете часть выражения C++ в скобки и добавляете *два знака* (@@) перед выражением, выражение интерпретируется в соответствии с правилами выражений MASM. Нельзя добавить пробел между двумя знаками *at* и открывающей скобкой. Окончательное значение этого выражения передается в средство оценки выражений C++ в качестве значения ULONG64. Вы также можете указать средство оценки выражений с помощью `@@c++( ... )` или `@@masm( ... )`.

Типы данных указываются как обычные на языке C++. Символы, указывающие массивы ([ ]), элементы указателя (->), элементы UDT (.) и члены классов (::) распознаются. Поддерживаются все арифметические операторы, включая операторы присваивания и операторы с побочными эффектами. Однако вы не можете использовать операторы `new`, `delete` и `throw`, и фактически не можете вызвать функцию.

Арифметика указателя поддерживается и смещения масштабируются правильно. Обратите внимание, что невозможно добавить смещение в указатель функции. Если необходимо добавить смещение в указатель функции, сначала приведите смещение к указателю символа.

Как и в C++, при использовании операторов с недопустимыми типами данных возникает синтаксическая ошибка. Средство синтаксического анализа выражений C++ отладчика использует немного более расслабленные правила, чем большинство компиляторов C++, но все основные правила применяются. Например, нельзя сместить значение, не являющееся целым числом.

Можно использовать следующие операторы. Операторы в каждой ячейке имеют приоритет над операторами в нижних ячейках. Операторы в одной ячейке имеют одинаковый приоритет и анализируются слева направо.

Как и в C++, оценка выражений заканчивается, когда его значение известно. Это окончание позволяет эффективно использовать такие выражения, как `?? myPtr && *myPtr`.

## Приведение ссылочных и типовых данных

Развернуть таблицу

| Оператор | Значение |
|---|---|
| *Выражение* // *Комментарий* | Игнорировать весь последующий текст |
| *Класс* :: *Член* | Член класса |
| *Класс* ::~*Элемент* | Член класса (деструктор) |
| :: *Имя* | Глобальный |
| *Структура* . *Поле* | Поле в структуре |
| *Указатель* ->*Поле* | Поле в указанной структуре |
| *Имя* [*целое число*] | Подстрочный индекс массива |
| *LValue* ++ | Инкремент (после вычисления) |
| *LValue* -- | Декремент (после оценки) |
| **dynamic_cast**<*тип*>(*Значение*) | Typecast (всегда выполняется) |
| **static_cast**<*тип*>(*Значение*) | Typecast (всегда выполняется) |
| **reinterpret_cast**<*тип*>(*Значение*) | Typecast (всегда выполняется) |
| **const_cast**<*тип*>(*Значение*) | Typecast (всегда выполняется) |

## Операции со значениями

Развернуть таблицу

| Оператор | Значение |
|---|---|
| (*тип*) *Ценность* | Typecast (всегда выполняется) |
| **sizeof***значение* | Размер выражения |

| Оператор | Значение |
| --- | --- |
| **sizeof**( *mun* ) | Размер типа данных |
| ++ *LValue* | Инкремент (перед оценкой) |
| -- *LValue* | Декремент (перед оценкой) |
| ~ *Ценность* | Битовое дополнение |
| ! *Ценность* | НЕ (Boolean) |
| *Ценность* | Унарный минус |
| + *Ценность* | Унарный плюс |
| & *LValue* | Адрес типа данных |
| *Ценность* | Разыменовать |
| *Структура . указателя* | Указатель на элемент структуры |
| *Указатель -> * Указатель* | Указатель на элемент ссылочной структуры |

## Арифметика

Развернуть таблицу

| Оператор | Значение |
| --- | --- |
| *Значение* | Умножение |
| *Ценность / Ценность* | Подразделение |
| *Ценность % Ценность* | Модуль |
| *Ценность + Ценность* | Дополнение |
| *Ценность - Ценность* | Вычитание |
| *Ценность<<Ценность* | Сдвиг влево по битам |
| *Ценность>>Ценность* | Побитовый сдвиг вправо |
| *Ценность<Ценность* | Меньше (сравнение) |
| *Ценность<= значение* | Меньше или равно (сравнение) |
| *Ценность>Ценность* | Больше, чем (сравнение) |
| *Ценность>= значение* | Больше или равно (сравнение) |

| Оператор | Значение |
| --- | --- |
| *Ценность* == *Ценность* | Равно (сравнение) |
| *Значение* != *Значение* | Не равно (сравнение) |
| *Значение* и *Значение* | Битовое И |
| *Ценность* ^ *Ценность* | Побитовое XOR (исключающее ИЛИ) |
| *Ценность* \| *Ценность* | Битовое ИЛИ |
| *Значение* && *Значение* | Логическое AND |
| *Ценность* \|\| *Ценность* | Логическое ИЛИ |

В следующих примерах предполагается, что псевдорегистры установлены, как показано ниже.

dbgcmd

```
0:000> r $t0 = 0
0:000> r $t1 = 1
0:000> r $t2 = 2
```

dbgcmd

```
0:000> ?? @$t1 + @$t2
unsigned int64 3
0:000> ?? @$t2/@$t1
unsigned int64 2
0:000> ?? @$t2|@$t1
unsigned int64 3
```

## Задание

Развернуть таблицу

| Оператор | Значение |
| --- | --- |
| *LValue* = *Ценность* | Назначать |
| *LValue* *= *Ценность* | *Умножить* и присвоить |
| *LValue* /= *Ценность* | Деление и назначение |
| *LValue* %= *Ценность* | Модуло и назначение |

| Оператор | Значение |
| --- | --- |
| *LValue* += *Ценность* | Добавление и назначение |
| *LValue* -= *Ценность* | Вычитание и назначение |
| *LValue*<<= *значение* | Смена влево и назначение |
| *LValue*>>= *значение* | Сдвиг вправо с присваиванием |
| *LValue* &= *Value* | И и назначение |
| *LValue* \|= *Ценность* | ИЛИ и назначение |
| *LValue* ^= *Ценность* | XOR и присвоение |

## Оценка

Развернуть таблицу

| Оператор | Значение |
| --- | --- |
| *Значение* ? *Значение* : *Значение* | Условная оценка |
| *Значение* , *Значение* | Оценка всех значений, а затем отмена всех, кроме самого правого значения |

## Макросы в выражениях C++

Макросы можно использовать в выражениях C++. Перед макросами необходимо добавить знак номера (#).

Можно использовать следующие макросы. Эти макросы имеют те же определения, что и макросы Microsoft Windows с тем же именем. Макросы Windows определены в `Winnt.h`.

Развернуть таблицу

| Макрос | Возвращаемое значение |
| --- | --- |
| #CONTAINING_RECORD(*адрес, тип, поле*) | Возвращает базовый адрес экземпляра структуры, учитывая тип структуры и адрес поля в структуре. |
| #FIELD_OFFSET(*Тип, поле*) | Возвращает смещение байтов именованного поля в известном типе структуры. |

| Макрос | Возвращаемое значение |
| --- | --- |
| #RTL_CONTAINS_FIELD(*структура, размер, поле*) | Указывает, включает ли заданный размер байтов требуемое поле. |
| #RTL_FIELD_SIZE(*Тип, поле*) | Возвращает размер поля в структуре известного типа, не требуя типа поля. |
| #RTL_NUMBER_OF(*Массив*) | Возвращает количество элементов в массиве статического размера. |
| #RTL_SIZEOF_THROUGH_FIELD(*Тип, поле*) | Возвращает размер структуры известного типа, вплоть до указанного поля. |

В этом примере показано использование `#FIELD_OFFSET` макроса для вычисления смещения байтов в поле в структуре.

dbgcmd

```
0:000> ?? #FIELD_OFFSET(_PEB, BeingDebugged)
long 0n2
```

# См. также

Выражения MASM и выражения C++

?? оценка выражения C++

? вычислять выражение

.expr выберите средство оценки выражений

Расширение для подписи

Примеры смешанных выражений

# Выражения MASM и выражения C++

Статья • 23.05.2024

Ниже приведены наиболее значительные различия между вычислением выражений MASM и оценкой выражений C++.

- В выражении MASM числовое значение любого символа является его адресом памяти. В выражении C++ числовое значение переменной является фактическим, а не его адресом. Структуры данных не имеют числовых значений. Вместо этого они рассматриваются как фактические структуры, и их необходимо использовать соответствующим образом. Значение имени функции или любой другой точки входа — адрес памяти и рассматривается как указатель функции. Если используется символ, который не соответствует типу данных C++ (например, немодифицированному имени модуля), возникает синтаксическая ошибка.

- Средство оценки выражений MASM обрабатывает все числа как ULONG64 значения. Средство оценки выражений C++ приводит числа к ULONG64 и сохраняет сведения о типе всех типов данных.

- Средство оценки выражений MASM позволяет использовать любой оператор вместе с любым числом. Средство оценки выражений C++ создает ошибку, если используется оператор вместе с неправильным типом данных.

- В средство оценки выражений MASM все арифметические выполняются буквально. В вычислителе выражений C++ арифметика указателя масштабируется должным образом и не допускается, если это недопустимо.

- Выражение MASM может использовать два символа подчеркивания ( __ ) или две двоеточия ( :: ), чтобы указать члены класса. Средство оценки выражений C++ использует только синтаксис двух двоеточия. Выходные данные *отладчика* всегда используют две двоеточия.

- В выражении MASM необходимо добавить знак (@) перед всеми, кроме наиболее распространенных регистров. Если этот знак не указан, имя регистра может быть интерпретировано как шестнадцатеричное число или как символ. В выражении C++ этот префикс требуется для всех регистров.

- Выражения MASM могут содержать ссылки на исходные строки. Эти ссылки указываются серьезными акцентами ( ` ). В выражении C++ нельзя ссылаться на номера исходной строки.

## См. также

Номера и операторы MASM

Номера и операторы C++

Примеры смешанных выражений

Расширение знака

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Примеры смешанных выражений

18.07.2025

В этих разделах содержатся примеры выражений MASM и C++, которые используются в различных командах.

Все остальные разделы этой справочной документации используют синтаксис выражения MASM (если иное не указано). Синтаксис выражений C++ очень полезен для управления структурами и переменными, но не анализирует параметры команд отладчика очень хорошо.

Если вы используете команды отладчика для общих целей или используете расширения отладчика, необходимо задать синтаксис выражения MASM в качестве синтаксиса по умолчанию, например с помощью .expr (выбор средства оценки выражений). Если у вас должен быть определенный параметр для использования синтаксиса выражения C++, используйте синтаксис **@@( )**.

Если **myInt** является ULONG32 значением и используете средство оценки выражений MASM, в следующих двух примерах показано значение **MyInt**.

dbgcmd

```
0:000> ?? myInt
0:000> dd myInt L1
```

Однако в следующем примере показан *адрес***myInt**.

dbgcmd

```
0:000> ? myInt
```

## Условные точки останова

Операторы сравнения можно использовать для создания условных точек останова. В следующем примере кода используется синтаксис выражения MASM. Так как текущий радикс по умолчанию равен 16, в примере используется префикс **0n** , чтобы число 20 было понято как десятичное число.

dbgcmd

```
0:000> bp MyFunction+0x43 "j ( poi(MyVar)>0n20 ) ''; 'gc' "
```

В предыдущем примере **MyVar** является целым числом в источнике C. Так как средство синтаксического анализа MASM обрабатывает все символы как адреса, в примере должен быть оператор **poi** для разыменования **MyVar**.

## Условные выражения

В следующем примере выводится значение **ecx**, если **eax** больше ebx, 7, если **eax** меньше ebx, и 3, если **eax** равно **ebx**. В этом примере используется средство оценки выражений MASM, поэтому знак равенства (=) является оператором сравнения, а не оператором назначения.

```
dbgcmd
```

```
0:000> ? ecx*(eax>ebx) + 7*(eax<ebx) + 3*(eax=ebx)
```

В синтаксисе C++, знак @ указывает на регистр, двойной знак равенства (==) является оператором сравнения, а код должен явно приводить из BOOL в **int**. Таким образом, в синтаксисе C++ предыдущая команда преобразуется в следующую.

```
dbgcmd
```

```
0:000> ?? @ecx*(int)(@eax>@ebx) + 7*(int)(@eax<@ebx) + 3*(int)(@eax==@ebx)
```

# Примеры смешанных выражений MASM и C++

Нельзя использовать выражения исходной строки в выражении C++. В следующем примере используется альтернативный синтаксис оценщика **@@( )** для вложения выражения MASM в выражение C++. Этот пример задает **MyPtr** равным адресу строки 43 файла Myfile.c.

```
dbgcmd
```

```
0:000> ?? MyPtr = @@( `myfile.c:43` )
```

В следующих примерах интерпретатор выражений по умолчанию задан на MASM, а затем оценить *Expression2* как C++ выражение, и оценить *Expression1* и *Expression3* как MASM выражения.

```
dbgcmd
```

```
0:000> .expr /s masm
```

```
0:000> bp Expression1 + @@( Expression2 ) + Expression3
```

Если **myInt** является значением типа ULONG64 и вы знаете, что за ним в памяти следует еще одно значение типа ULONG64, можно установить точку останова доступа в этом месте, воспользовавшись одним из следующих примеров. (Обратите внимание на использование арифметики указателя.)

dbgcmd

```
0:000> ba r8 @@( &myInt + 1 )
0:000> ba r8 myInt + 8
```

## См. также

Номера и операторы MASM

Номера и операторы C++

Выражения MASM и выражения C++

Расширение знака

# Расширение знака

Статья • 30.04.2025

Если 32-разрядное целое число со знаком отрицательное, его самый высокий бит равен одному. Если это 32-разрядное целое число со знаком преобразуется в 64-разрядное число, то высокие биты можно установить в ноль (сохраняя целое число без знака и шестнадцатеричное значение числа) или установить в единицу (сохраняя значение числа со знаком). Последняя ситуация называется *расширением знака*.

Отладчик следует различным правилам для расширения знака в выражениях MASM, в выражениях C++ и при отображении чисел.

## Расширение знака в выражениях MASM

В определенных условиях числа автоматически *расширяются со знаком* вычислителем выражений MASM. Расширение подписи может повлиять только на числа от 0x80000000 до 0xFFFFFFFF. То есть расширение знака влияет только на числа, которые могут быть записаны в 32 бита с высоким битом равным 1.

Число 0x12345678 всегда остается 0x00000000 12345678, когда отладчик обрабатывает его как 64-разрядное число. С другой стороны, если 0x890ABCDE рассматривается как 64-разрядное значение, оно может оставаться 0x00000000'890ABCDE или средство оценки выражений MASM может расширить его до 0xFFFFFFFF'890ABCDE.

Число от 0x80000000 до 0xFFFFFFFF подписывается на основе следующих критериев:

- Числовые константы никогда не регистрируются в пользовательском режиме. В режиме ядра числовая константа является знаковым расширением, если она не содержит гравис (`) до низких байт. Например, в режиме ядра шестнадцатеричные числа **EEAA1122** и **00000000EEAA1122** являются расширенными, но **00000000`EEAA1122** и **0`EEAA1122** не являются.

- 32-разрядный регистр поддерживается в обоих режимах.

- Псевдорегистры всегда хранятся в виде 64-разрядных значений. Они не подписываются при оценке. При *присвоении* значения псевдорегистру выражение оценивается согласно стандартным критериям C++.

- Отдельные числа и регистры в выражении могут быть расширены со знаком, но никакие другие вычисления во время оценки выражений не расширяются со знаком. В результате можно маскировать большие биты числа или регистрировать с помощью следующего синтаксиса.

  ```
  ```

Консоль

```
( 0x0`FFFFFFFF & expression )
```

# Расширение знака в выражениях C++

При оценке выражения C++ отладчик применяет следующие правила:

- Регистры и псевдорегистры никогда не являются расширенными.
- Все остальные значения обрабатываются точно так же, как C++ будет обрабатывать значения их типа.

# Отображение Sign-Extended и 64-разрядных чисел

Кроме 32-разрядных и 16-разрядных регистров, все числа хранятся внутри отладчика как 64-разрядные значения. Однако если число удовлетворяет определенным критериям, отладчик отображает его как 32-разрядное число в выходных данных команды.

Отладчик использует следующие критерии для определения способа отображения чисел:

- Если высокие 32 бита числа равны нулю (то есть, если число от 0x00000000'00000000 до 0x00000000'FFFFFFFFF), отладчик отображает число как 32-разрядное число.
- Если старшие 32 бита числа равны единицам и если старший бит младших 32 бит также равен единице (то есть, если число находится в диапазоне от 0xFFFFFFFF`80000000 до 0xFFFFFFFF`FFFFFFFF), отладчик предполагает, что это число является расширенным со знаком 32-разрядным числом и отображает его как 32-разрядное число.
- Если предыдущее два условия не применяются (то есть если число от 0x00000001'00000000 до 0xFFFFFFFF'7FFFFFFFFFF) отладчик отображает число как 64-разрядное число.

Из-за этих правил отображения, если число отображается как 32-разрядное число от 0x80000000 до 0xFFFFFFFF, вы не можете подтвердить, являются ли высокие 32 биты всеми или всеми нулями. Чтобы различать эти два случая, необходимо выполнить дополнительные вычисления по числу (например, маскирование одного или нескольких больших битов и отображение результата).

# См. также

Номера и операторы MASM

Номера и операторы C++

Выражения MASM и выражения C++

Примеры смешанных выражений

# Синтаксис строковых подстановочных знаков

Статья • 26.10.2023

Некоторые команды отладчика имеют строковые параметры, которые принимают различные подстановочные знаки. Эти параметры отмечаются на соответствующих справочных страницах.

Эти типы параметров поддерживают следующие функции синтаксиса:

- Звездочка (*) представляет ноль или более символов.
- Вопросительный знак (?) представляет любой один символ.
- Квадратные скобки ( [ ] ), содержащие список символов, представляют любой отдельный символ в списке. Совпадает только один символ в списке. В этих скобках можно использовать дефис (-) для указания диапазона. Например, **Prog[er-t7]am** соответствует "Progeam", "Program", "Progsam", "Progtam" и "Prog7am".
- Числовой знак (#) представляет ноль или более предыдущих символов. Например, **Lo#p** соответствует "Lp", "Lop", "Loop", "Looop" и т. д. Можно также объединить знак числа с квадратными скобками, поэтому **m[ia]#n** соответствует "mn", "min", "man", "maan", "main", "mian", "miin", "miain" и т. д.
- Знак "плюс" (+) представляет один или несколько предыдущих символов. Например, **lo+p** совпадает с **Lo#p**, за исключением того, что **Lo+p** не соответствует "Lp". Аналогичным образом **m[ia]+n** совпадает с **m[ia]#n**, за исключением того, что m**[ia]+n** не соответствует "mn". **a?+b** также совпадает с **a*b**, за исключением того, что **a?+b** не соответствует "ab".
- Если необходимо указать литеральный знак (#), вопросительный знак (?), открывающая скобка ([), закрывающая скобка (]), звездочка (*) или знак плюса (+), необходимо добавить обратную косую черту ( \ ) перед символом. Дефисы всегда являются литеральными, если вы не заключаете их в квадратные скобки. Но вы не можете указать литеральный дефис в списке, заключенном в квадратные скобки.

Параметры, указывающие символы, также поддерживают некоторые дополнительные функции. Помимо стандартных строковых подстановочных знаков можно использовать символ подчеркивания (_) перед текстовым выражением,

которое используется для указания символа. При сопоставлении этого выражения с символом отладчик обрабатывает символ подчеркивания как любое количество символов подчеркивания, даже ноль. Эта функция применяется только при сопоставлении символов. Он не применяется к строковым выражениям с подстановочными знаками в целом. Дополнительные сведения о синтаксисе символов см. в разделе Синтаксис символов и сопоставление символов.

# Синтаксис регистрации

18.07.2025

Отладчик может управлять регистрами и регистрами с плавающей запятой.

При использовании регистра в выражении следует добавить знак "@" перед регистром. Знак "@" указывает отладчику, что следующий текст является именем регистра.

Если вы используете синтаксис выражений MASM, вы можете опустить символ '@' для некоторых очень распространенных регистров. В системах на основе x86 можно опустить знак для **eax**, **ebx, ecx**, **edx**, **esi**, **edi**, **ebp**, **eip** и **efl** регистров. Однако, при указании менее распространенного регистра без знака @, отладчик сначала пытается интерпретировать текст как шестнадцатеричное число. Если текст содержит не шестнадцатеричные символы, отладчик затем интерпретирует текст как символ. Наконец, если отладчик не находит совпадения символов, отладчик интерпретирует текст как регистр.

При использовании синтаксиса выражений C++ всегда требуется ат-символ (@).

Команда **r (Registers)** является исключением из этого правила. Отладчик всегда интерпретирует свой первый аргумент как регистр. (Символ "@" не требуется и не разрешён.) Если для команды **r** есть второй аргумент, он интерпретируется в соответствии с синтаксисом выражения по умолчанию. Если синтаксис выражения по умолчанию — C++, необходимо использовать следующую команду, чтобы скопировать регистр **ebx** в регистр **eax** .

dbgcmd

```
0:000> r eax = @ebx
```

Дополнительные сведения о регистрах и инструкциях, относящихся к каждому процессору, см. в разделе "Архитектура процессора".

## Флаги на процессоре на основе x86

Процессоры на основе x86 также используют несколько *1-разрядных регистров, известных как флаги*. Дополнительные сведения об этих флагах и синтаксисе, которые можно использовать для просмотра или изменения этих флагов, см. в разделе "Флаги x86".

## Регистры и потоки

Каждый поток имеет собственные значения регистра. Эти значения хранятся в регистрах ЦП при выполнении потока и в памяти при выполнении другого потока.

В пользовательском режиме любая ссылка на регистр интерпретируется как регистр, связанный с текущим потоком. Дополнительные сведения о текущем потоке см. в разделе "Управление процессами и потоками".

В режиме ядра любая ссылка на регистр интерпретируется как регистр, связанный с текущим контекстом регистра. Контекст регистра можно задать для сопоставления с определенным потоком, записью контекста или кадром ловушки. Вы можете отобразить только наиболее важные регистры для указанного контекста регистра, и изменить их значения нельзя.

# Синтаксис Pseudo-Register

18.07.2025

Отладчик поддерживает несколько псевдорегистров, которые содержат определенные значения.

Отладчик задает *автоматические псевдорегистры* для определенных полезных значений. *Определяемые пользователем псевдорегистры* — это целые переменные, в которые можно записывать или читать.

Все псевдорегистры начинаются с знака доллара ($). Если вы используете синтаксис MASM, вы можете добавить знак по знаку ( @ ) перед знаком доллара. Этот символ сообщает отладчику, что следующий токен является регистром или псевдорегистром, а не символом. Если вы опустите значок @, отладчик будет реагировать медленнее, так как ему приходится выполнять поиск по всей таблице символов.

Например, следующие две команды создают одинаковые выходные данные, но вторая команда быстрее.

dbgcmd

```
0:000> ? $exp
Evaluate expression: 143 = 0000008f
0:000> ? @$exp
Evaluate expression: 143 = 0000008f
```

Если символ существует с тем же именем, что и псевдорегистр, необходимо добавить знак @.

Если используется синтаксис выражения C++, всегда требуется знак ( @ ).

Команда **r (Registers)** является исключением из этого правила. Отладчик всегда интерпретирует свой первый аргумент как регистр или псевдорегистратор. (Символ "@" не требуется и не разрешён.) Если для команды **r** есть второй аргумент, он интерпретируется в соответствии с синтаксисом выражения по умолчанию. Если синтаксис выражения по умолчанию — C++, необходимо использовать следующую команду, чтобы скопировать псевдорегистр **$t2** в псевдорегистр **$t1**.

dbgcmd

```
0:000> r $t1 = @$t2
```

# Автоматическая Pseudo-Registers

Отладчик автоматически задает следующие псевдорегистры.

Развернуть таблицу

| Псевдорегистр | Описание |
|---|---|
| **$ea** | Действующий адрес последней инструкции, которая была выполнена. Если эта инструкция не имеет эффективного адреса, отладчик отображает ошибку "Ошибка регистрации". Если эта инструкция имеет два эффективных адреса, отладчик отображает первый адрес. |
| **$ea 2** | Второй эффективный адрес последней инструкции, которая была выполнена. Если в этой инструкции нет двух эффективных адресов, отладчик отображает сообщение "Ошибка в регистре". |
| **$exp** | Последнее выражение, которое было оценено. |
| **$ra** | Адрес возврата, который сейчас находится в стеке.<br><br>Этот адрес особенно полезен в командах выполнения. Например, **g @$ra** продолжается до тех пор, пока возвращаемый адрес не найден (хотя **gu (Go Up)** является более точным эффективным способом "выхода" текущей функции. |
| **$ip** | Регистр указателя инструкций.<br><br>**Процессоры на основе x86:** То же самое, что **и eip**. **Процессоры на основе Itanium:** Связана с **iip**. (Дополнительные сведения см. в записке ниже этой таблицы.) **Процессоры на основе x64:** Такой же, как **rip**. |
| **$eventip** | Инструкционный указатель в момент текущего события. Этот указатель обычно соответствует **$ip**, если вы не переключили потоки или вручную не изменили значение указателя инструкции. |
| **$previp** | Указатель инструкций на момент предыдущего события. (Вторжение в отладчик считается событием.) |

| Псевдорегистр | Описание |
| --- | --- |
| **$relip** | Указатель инструкции, связанный с текущим событием. При трассировке ветви этот указатель является указателем на источник ветви. |
| **$scopeip** | Указатель инструкции для текущего локального контекста (также называется *областью видимости*). |
| **$exentry** | Адрес точки входа первого исполняемого файла текущего процесса. |
| **$retreg** | Основной регистр возвращаемого значения.<br><br>**Процессоры на основе x86:** То же самое, что **и eax**. **Процессоры на основе Itanium:** То же самое, что **и ret0**. **Процессоры на основе x64:** То же самое, что **и rax**. |
| **$retreg 64** | Основной регистр возвращаемого значения в 64-разрядном формате.<br><br>**Процессор x86:** То же, что и пара **edx:eax** . |
| **$csp** | Текущий указатель стека вызовов. Этот указатель является регистром, который наиболее точно отражает глубину стека вызовов.<br><br>**Процессоры на основе x86:** То же самое, что **и esp**. **Процессоры на основе Itanium:** То же самое, что **и bsp**. **Процессоры на основе x64:** То же самое, что **и rsp**. |
| **$p** | Значение, которое была напечатана последней командой **d* (отображаемая память)**. |
| **$proc** | Адрес текущего процесса (то есть адрес блока EPROCESS). |
| **$thread** | Адрес текущего потока. В отладке в режиме ядра этот адрес является адресом блока ETHREAD. В отладке в пользовательском режиме этот адрес — это адрес блока среды потока (TEB). |
| **$peb** | Адрес блока среды процесса (PEB) текущего процесса. |
| **$teb** | Адрес блока среды потока (TEB) текущего потока. |

| Псевдорегистр | Описание |
| --- | --- |
| **$tpid** | Идентификатор процесса (PID) для процесса, которому принадлежит текущий поток. |
| **$tid** | Идентификатор потока для текущего потока. |
| **$dtid** | |
| **$dpid** | |
| **$dsid** | |
| **номер $bp** | Адрес соответствующей точки останова. Например, **$bp 3** (или **$bp 03**) ссылается на точку останова, идентификатор точки останова которой равен 3. *Число* всегда является десятичным числом. Если ни одна точка останова не имеет *ID равного* ***Number***, **$bp***Number* оценивается как ноль. Дополнительные сведения об точках останова см. в разделе "Использование точек останова". |
| **$frame** | Текущий индекс кадров. Этот индекс совпадает с номером кадра, который использует команда **.frame (Set Local Context).** |
| **$dbgtime** | Текущее время, в соответствии с компьютером, на котором работает отладчик. |
| **$callret** | Возвращаемое значение последней функции, которая вызывается **(функция вызова)** или используется в команде **Fnret /s** . Тип данных **$callret** — это тип данных этого возвращаемого значения. |
| **$extret** | |
| **$extin** | |
| **$clrex** | |
| **$lastclrex** | **Только управляемая отладка:** Адрес последнего обнаруженного объекта исключения среды CLR. |
| **$ptrsize** | Размер указателя. В режиме ядра этот размер — это размер указателя на целевом компьютере. |
| **$pagesize** | Количество байтов в одной странице памяти. В режиме ядра этот размер — это размер страницы на целевом компьютере. |

| Псевдорегистр | Описание |
| --- | --- |
| **$pcr** | |
| **$pcrb** | |
| **$argreg** | |
| **$exr_шанс** | Вероятность текущей записи исключений. |
| **$exr_code** | Код исключения для текущей записи исключений. |
| **$exr_numparams** | Количество параметров в текущей записи исключений. |
| **$exr_param0** | Значение параметра 0 в текущей записи исключений. |
| **$exr_param1** | Значение параметра 1 в текущей записи исключений. |
| **$exr_param2** | Значение параметра 2 в текущей записи исключений. |
| **$exr_param3** | Значение параметра 3 в текущей записи исключений. |
| **$exr_param4** | Значение параметра 4 в текущей записи исключений. |
| **$exr_param5** | Значение параметра 5 в текущей записи исключений. |
| **$exr_param6** | Значение параметра 6 в текущей записи исключений. |
| **$exr_param7** | Значение параметра 7 в текущей записи исключений. |
| **$exr_param8** | Значение параметра 8 в текущей записи исключений. |
| **$exr_param9** | Значение параметра 9 в текущей записи исключений. |
| **$exr_param10** | Значение параметра 10 в текущей записи исключений. |
| **$exr_param11** | Значение параметра 11 в текущей записи исключений. |

| Псевдорегистр | Описание |
|---|---|
| **$exr_param12** | Значение параметра 12 в текущей записи исключений. |
| **$exr_param13** | Значение параметра 13 в текущей записи исключений. |
| **$exr_param14** | Значение параметра 14 в текущей записи исключений. |
| **$bug_code** | Если произошла проверка ошибок, это код ошибки. Применяется к отладке в динамическом режиме ядра и аварийным дампам ядра. |
| **$bug_param1** | Если произошла проверка ошибок, это значение параметра 1. Применяется к отладке ядра в режиме реального времени и аварийным дампам ядра. |
| **$bug_param2** | Если произошла проверка на наличие ошибки, это значение параметра 2. Применяется к отладке ядра в режиме реального времени и аварийным дампам ядра. |
| **$bug_param3** | Если произошла системная ошибка, это значение параметра 3. Применяется к отладке в динамическом режиме ядра и аварийным дампам ядра. |
| **$bug_param4** | Если произошла проверка на ошибки, это значение параметра 4. Применяется к отладке в динамическом режиме ядра и аварийным дампам ядра. |

Некоторые из этих псевдорегистрации могут быть недоступны в определенных сценариях отладки. Например, нельзя использовать **$peb**, **$tid** и **$tpid** при отладке мини-dump в пользовательском режиме или определенных файлов дампа в режиме ядра. В таких ситуациях вы можете узнать сведения о потоке из **~ (состояние потока),** но не из **$tid**. Вы не можете использовать псевдорегистратор **$previp** в первом событии отладчика. Вы не можете использовать псевдорегистр **$relip**, если только вы не трассируете ветви. При использовании недоступного псевдорегистрирующего регистра возникает синтаксическая ошибка.

Псевдорегистратор, содержащий адрес структуры, например **$thread**, **$proc, $teb**, **$peb** и **$lastclrex**, будет оцениваться в соответствии с соответствующим типом данных в средстве оценки выражений C++, но не в средстве оценки выражений MASM. Например, команда

**? $teb** отображает адрес TEB, а команда **?? @$teb** отображает всю структуру TEB. Дополнительные сведения см. в разделе "Оценка выражений".

На процессоре на основе Itanium регистр **iip***выравнен по пакетам*, что означает, что он указывает на слот 0 в пакете, содержащем текущую инструкцию, даже если выполняется другой слот. Итак, **iip** не является полным указателем инструкции. **$ip** псевдорегистр является фактическим указателем команд, включая бандл и слот. Другие псевдорегистры, которые содержат указатели адресов (**$ra**, **$retreg**, **$eventip**, **$previp**, **$relip** и **$exentry**) имеют одинаковую структуру, что и **$ip** на всех процессорах.

Команду **r** можно использовать для изменения значения **$ip**. Это изменение также автоматически изменяет соответствующий регистр. При возобновлении выполнения, выполнение продолжается по новому адресу указателя инструкции. Этот регистр является единственным автоматическим псевдорегистром, который можно изменить вручную.

**Заметка** В синтаксисе MASM вы можете указать псевдорегистр **$ip** через точку (.). Вы не добавляете знак по адресу (@) до этого периода и не используете период в качестве первого параметра команды **r** . Этот синтаксис не допускается в выражении C++.

Автоматические псевдорегистры похожи на автоматические псевдонимы. Но вы можете использовать автоматические псевдонимы вместе с маркерами, связанными с псевдонимом (например, **${ }**), и вы не можете использовать псевдорегистры с такими токенами.

## User-Defined Pseudo-Registers

Существует 20 пользовательских псевдорегистров (**$t 0**, **$t 1**, ..., **$t 19**). Эти псевдорегистраторы — это переменные, которые можно читать и записывать с помощью отладчика. Вы можете сохранить любое целочисленное значение в этих псевдорегистрации. Они могут быть особенно полезны в качестве переменных цикла.

Чтобы записать в один из этих псевдорегистров, используйте команду **r (Registers),** как показано в следующем примере.

dbgcmd

```
0:000> r $t0 = 7
0:000> r $t1 = 128*poi(MyVar)
```

Как и все псевдорегистры, вы можете использовать определяемый пользователем псевдорегистр в любом выражении, как показано в следующем примере.

dbgcmd

```
0:000> bp $t3
0:000> bp @$t4
0:000> ?? @$t1 + 4*@$t2
```

Псевдорегистр всегда является целым числом, если вы не используете параметр **?** вместе с командой **r**. Если вы используете этот параметр, псевдорегистр приобретает тип, который ему назначается. Например, следующая команда назначает тип UNICODE_STRING** и значение 0x0012FFBC **$t 15**.

dbgcmd

```
0:000> r? $t15 = * (UNICODE_STRING*) 0x12ffbc
```

Определяемые пользователем псевдорегистры используют ноль в качестве значения по умолчанию при запуске отладчика.

**Заметка** Псевдонимы **$u 0**, **$u 1**, ..., **$u 9** не являются псевдорегистрацией, несмотря на их аналогичный внешний вид. Дополнительные сведения об этих псевдонимах см. в разделе "Использование псевдонимов".

## Пример

В следующем примере устанавливается точка останова, которая достигается каждый раз, когда текущий поток вызывает **NtOpenFile**. Но эта точка останова не достигается, когда другие потоки вызывают **NtOpenFile**.

dbgcmd

```
kd> bp /t @$thread nt!ntopenfile
```

## Пример

В следующем примере выполняется команда, пока регистр не содержит указанное значение. Сначала поместите следующий код для условного шага в файл скрипта с именем eaxstep.

dbgcmd

```
.if (@eax == 1234) { .echo 1234 } .else { t "$<eaxstep" }
```

Затем выполните следующую команду.

dbgcmd

```
t "$<eaxstep"
```

Отладчик выполняет шаг, а затем выполняет команду. В этом случае отладчик запускает скрипт, который отображает **1234** или повторяет процесс.

# Синтаксис исходной строки

18.07.2025

Можно указать номера строк исходного файла как все или часть выражения MASM. Эти числа оценивают смещение исполняемого кода, соответствующего этой исходной строке.

**Заметка** В рамках выражения C++ нельзя использовать номера исходной строки. Дополнительные сведения об использовании синтаксиса выражений MASM и C++ см. в разделе "Оценка выражений".

Необходимо заключить выражения исходного файла и номера строк в грависы ( `). В следующем примере показан полный формат номеров строк исходного файла.

```text
`[[Module!]Filename][:LineNumber]`
```

Если у вас несколько файлов с одинаковыми именами файлов, *имя файла* должно содержать весь путь к каталогу и имя файла. Этот путь к каталогу должен быть тем, который используется во время компиляции. Если указать только имя файла или только часть пути, а при наличии нескольких совпадений отладчик использует первое совпадение, которое он находит.

Если не указано *имя файла*, отладчик использует исходный файл, соответствующий текущему счетчику программы.

*LineNumber* считывается как десятичное число, если не предшествует **0x**, независимо от текущего радикса по умолчанию. Если не указан *LineNumber*, выражение вычисляет исходный адрес исполняемого файла, соответствующему исходному файлу.

Выражения исходной строки не оцениваются в CDB, если вы не вводите команду **.lines (переключение поддержки строк исходного кода)** или не добавляете параметр **-lines** командной строки при запуске WinDbg.

Дополнительные сведения об отладке источника см. в разделе "Отладка в режиме источника".

# Синтаксис диапазона адресов и адресов

Статья • 08.07.2024

В отладчике можно указать адреса несколькими способами.

Адреса обычно *являются виртуальными*, за исключением случаев, когда документация специально указывает другой тип адреса. В пользовательском режиме отладчик интерпретирует виртуальные адреса в соответствии с каталогом страниц текущего процесса. В режиме ядра отладчик интерпретирует виртуальные адреса в соответствии с каталогом страницы процесса, который указывает контекст процесса. Вы также можете напрямую задать контекст *адреса в пользовательском* режиме. Дополнительные сведения о контексте адреса в пользовательском режиме см. в разделе **.context (Set User-Mode Address Context)**.

В выражениях MASM можно использовать **оператор poi** для расшифровки любого указателя. Например, если указатель на адрес 0x0000008e'ed57b108 указывает на расположение адреса 0x805287637256, следующие две команды эквивалентны.

dbgcmd

```
0:000> dd 805287637256
0:000> dd poi(000000bb`7ee23108)
```

# Пример отображения адреса памяти

Чтобы просмотреть пример использования poi, определите смещение для *CurrentLocale* блока среды потока (TEB). Используйте команду dx для отображения @$teb, который является примером псевдорегистров, которые содержат общие адреса, например текущее расположение счетчика программы.

dbgcmd

```
0:000> dx @$teb
@$teb                 : 0x1483181000 [Type: _TEB *]

...

    [+0x108] CurrentLocale    : 0x409 [Type: unsigned long]
```

*CurrentLocale* — +0x108 с начала TEB. Затем определите адрес памяти этого расположения.

```
dbgcmd

0:000> ? @$teb + 0x108
Evaluate expression: 613867303176 = 0000008e`ed57b108
```

Используйте poi для расшифровки этого адреса, чтобы увидеть, что он содержит значение CurrentLocale 0x409.

```
dbgcmd

0:000> ? poi(0000008e`ed57b108)
Evaluate expression: 1033 = 00000000`00000409
```

В выражениях отладчика C++ указатели ведут себя как указатели в C++. Однако числа интерпретируются как целые числа. Если необходимо отложить фактическое число, может потребоваться сначала привести его, как показано в следующем примере.

Чтобы попробовать это, используйте .expr , чтобы задать средство оценки выражений на C++.

```
dbgcmd

0:000> .expr /s C++
Current expression evaluator: C++ - C++ source expressions
```

С помощью средства оценки выражений, установленного на C++, мы можем привести значение long.

```
dbgcmd

0:000> d *((long*)0x00000014`83181108 )
00000000`00000409  ???????? ???????? ???????? ????????
```

Дополнительные сведения о приведение числовых значений см. в разделе чисел и операторов C++.

Если для вычислителя выражений задано значение c++, то мы можем упаковать указатель poi с *помощью @@masm()*, чтобы только часть выражения, вычисляемого вычислителем выражений MASM.

dbgcmd

```
0:000> .expr /s c++
Current expression evaluator: C++ - C++ source expressions

0:000> ? @@masm(poi(00000078`267d7108))
Evaluate expression: 1033 = 00000000`00000409
```

Дополнительные сведения о двух вычислителях выражений см. в разделе "Оценка выражений".

Можно также указать адрес в приложении, указав исходное имя исходного файла и номер строки. Дополнительные сведения об указании этой информации см. в разделе "Синтаксис исходной строки".

# Диапазоны адресов

Диапазон адресов можно указать по паре адресов или по количеству адресов и объектов.

Чтобы указать диапазон по паре адресов, укажите начальный адрес и конечный адрес. Например, в следующем примере представлен диапазон 8 байт, начиная с адреса 0x00001000.

dbgcmd

```
0x00001000  0x00001007
```

Чтобы указать диапазон адресов по количеству адресов и объектов, укажите аргумент адреса, букву L (верхний или нижний регистр) и аргумент значения. Адрес указывает начальный адрес. Значение указывает количество объектов для проверки или отображения. Размер объекта зависит от команды. Например, если размер объекта равен 1 байту, следующий пример — это диапазон 8 байтов, начиная с адреса 0x00001000.

dbgcmd

```
0x00001000  L8
```

Однако если размер объекта является двойным словом (32 бита или 4 байта), следующие два диапазона дают диапазон 8-байтов.

dbgcmd

```
0x00001000  0x00001007
0x00001000  L2
```

## Описатель диапазона L

Существует два других способа указать значение (*описатель диапазона L Size):*

- **L?** *Размер (с вопросительным знаком) означает тот же размер, что и L,* за исключением того, что **L?** *Размер* удаляет автоматическое ограничение диапазона отладчика. Как правило, существует ограничение диапазона в 256 МБ, так как большие диапазоны являются типографическими ошибками. Если вы хотите указать диапазон, размер которого превышает 256 МБ, необходимо использовать **L?** *Синтаксис размера* .

- **L-Size** (с дефисом) задает диапазон длины, который заканчивается на заданном адресе. Например, **8000000 L20** указывает диапазон от 0x80000000 до 0x8000001F, а **80000000 L-20** указывает диапазон от 0x7FFFFFE0 до 0x7FFFFFFF.

Некоторые команды, запрашивающие диапазоны адресов, принимают один адрес в качестве аргумента. В этой ситуации команда использует некоторое число объектов по умолчанию для вычисления размера диапазона. Как правило, команды, для которых диапазон адресов является окончательным параметром, разрешают этот синтаксис. Точный синтаксис и размер диапазона по умолчанию для каждой команды см. в справочных разделах для каждой команды.

## Пример диапазона памяти поиска

Сначала мы определим адрес регистра указателя инструкции rip с помощью вычислителя выражений MASM.

dbgcmd

```
0:000> ? @rip
Evaluate expression: 140720561719153 = 00007ffc`0f180771
```

Затем мы будем искать начиная с 00007ffc'0f180771, для 100000 с помощью команды s (Search Memory). Мы указываем диапазон для поиска с помощью L100000.

dbgcmd

```
0:000> s -a 00007ffc`0f180771 L100000 "ntdll"
00007ffc`0f1d48fa  6e 74 64 6c 6c 5c 6c 64-72 69 6e 69 74 2e 63 00
ntdll\ldrinit.c.
00007ffc`0f1d49c2  6e 74 64 6c 6c 5c 6c 64-72 6d 61 70 2e 63 00 00
ntdll\ldrmap.c..
00007ffc`0f1d4ab2  6e 74 64 6c 6c 5c 6c 64-72 72 65 64 69 72 65 63
ntdll\ldrredirec
00007ffc`0f1d4ad2  6e 74 64 6c 6c 5c 6c 64-72 73 6e 61 70 2e 63 00
ntdll\ldrsnap.c.
...
```

Кроме того, можно указать тот же диапазон, что и в двух адресах памяти.

dbgcmd

```
0:000> s -a 0x00007ffc`0f180771 0x00007ffc`0f280771 "ntdll"
00007ffc`0f1d48fa  6e 74 64 6c 6c 5c 6c 64-72 69 6e 69 74 2e 63 00
ntdll\ldrinit.c.
00007ffc`0f1d49c2  6e 74 64 6c 6c 5c 6c 64-72 6d 61 70 2e 63 00 00
ntdll\ldrmap.c..
00007ffc`0f1d4ab2  6e 74 64 6c 6c 5c 6c 64-72 72 65 64 69 72 65 63
ntdll\ldrredirec
00007ffc`0f1d4ad2  6e 74 64 6c 6c 5c 6c 64-72 73 6e 61 70 2e 63 00
ntdll\ldrsnap.c.
...
```

Наконец, мы можем искать назад в диапазоне памяти с помощью параметра L-length.

dbgcmd

```
0:000> s -a 00007ffc`0f1d4ad2 L-100000 "ntdll"
00007ffc`0f1d48fa  6e 74 64 6c 6c 5c 6c 64-72 69 6e 69 74 2e 63 00
ntdll\ldrinit.c.
00007ffc`0f1d49c2  6e 74 64 6c 6c 5c 6c 64-72 6d 61 70 2e 63 00 00
ntdll\ldrmap.c..
00007ffc`0f1d4ab2  6e 74 64 6c 6c 5c 6c 64-72 72 65 64 69 72 65 63
ntdll\ldrredirec
```

# Пример отмены памяти

В этом примере используется команда u (unassemble) и параметр L для отмены удаления трех байтов кода.

dbgcmd

```
0:000> u 00007ffc`0f1d48fa L3
ntdll!`string'+0xa:
00007ffc`0f1d48fa 6e              outs    dx,byte ptr [rsi]
00007ffc`0f1d48fb 7464            je      ntdll!`string'+0x21
(00007ffc`0f1d4961)
00007ffc`0f1d48fd 6c              ins     byte ptr [rdi],dx
```

Или укажите три диапазона байтов памяти, чтобы отменить это.

dbgcmd

```
0:000> u 00007ffc`0f1d48fa 00007ffc`0f1d48fd
ntdll!`string'+0xa:
00007ffc`0f1d48fa 6e              outs    dx,byte ptr [rsi]
00007ffc`0f1d48fb 7464            je      ntdll!`string'+0x21
(00007ffc`0f1d4961)
00007ffc`0f1d48fd 6c              ins     byte ptr [rdi],dx
```

# Режимы адресов и поддержка сегментов

На платформах x86 CDB и KD поддерживают следующие режимы адресации. Эти режимы отличаются их префиксами.

Развернуть таблицу

| Префикс | Имя. | Типы адресов |
|---|---|---|
| % | flat | 32-разрядные адреса (также 16-разрядные селекторы, указывающие на 32-разрядные сегменты) и 64-разрядные адреса в 64-разрядных системах. |
| & | virtual 86 | Адреса в режиме реального режима. Только на основе x86. |
| # | plain | Адреса в режиме реального режима. Только на основе x86. |

Разница между режимами обычного и виртуального 86 заключается в том, что обычный 16-разрядный адрес использует значение сегмента в качестве селектора и ищет дескриптор сегмента. Но виртуальный 86-адрес не использует селекторы и вместо этого сопоставляется непосредственно с меньшим размером 1 МБ.

Если доступ к памяти осуществляется через режим адресации, который не является текущим режимом по умолчанию, можно использовать префиксы режима адреса для переопределения текущего режима адреса.

# Аргументы адреса

Аргументы адреса указывают расположение переменных и функций. В следующей таблице объясняется синтаксис и смысл различных адресов, которые можно использовать в CDB и KD.

Развернуть таблицу

| Синтаксис | Значение |
|---|---|
| offset | Абсолютный адрес в пространстве виртуальной памяти с типом, соответствующим текущему режиму выполнения. Например, если текущий режим выполнения равен 16 биту, смещение равно 16 биту. Если режим выполнения сегментирован в 32-разрядном режиме, смещение равно 32-разрядному сегменту. |
| **&**[[ сегмент:]] смещение | Реальный адрес. на основе x86 и на основе x64. |
| %segment:[[смещение]] | Сегментированные 32-разрядные или 64-разрядные адреса. на основе x86 и на основе x64. |
| %[[смещение]] | Абсолютный адрес (32-разрядная или 64-разрядная) в пространстве виртуальной памяти. на основе x86 и на основе x64. |
| name[[ +\|- ]] смещение | Неструктурированный 32-разрядный или 64-разрядный адрес. *имя* может быть любым символом. *смещение указывает смещение* . Это смещение может быть любым режимом адреса, который указывает префикс. Префикс по умолчанию не указывает адрес режима по умолчанию. Смещение можно указать как положительное (+) или отрицательное значение (-). |

**Используйте команду dg (селектор отображения)** для просмотра сведений о дескрипторе сегмента.

# См. также

Чтобы отобразить сведения о памяти, используйте команду !address .

Для поиска памяти используйте команду s (Search Memory).

Чтобы отобразить содержимое памяти, используйте команду d, da, db, dc, dd, dD, df, dp, dq, du, dw (display Memory).

Сведения о том, как можно просматривать и изменять память с помощью окна памяти, см. в разделе "Использование окна памяти".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Thread Syntax

18.07.2025

Многие команды отладчика имеют идентификаторы потоков в качестве параметров. Тильда (~) отображается перед идентификатором потока.

Идентификатор потока может быть одним из следующих значений.

Развернуть таблицу

| Thread identifier | Description |
|---|---|
| ~. | Текущий поток. |
| ~# | Поток, вызвавшего текущее исключение или событие отладки. |
| ~* | Все потоки в процессе. |
| ~ *Number* | The thread whose index is *Number*. |
| ~~[*TID*] | The thread whose thread ID is *TID*. (Квадратные скобки обязательны, и нельзя добавить пробел между вторым тильдой и открывающей скобкой.) |
| ~[*Expression*] | The thread whose thread ID is the integer to which the numerical *Expression* resolves. |

Потоки назначаются индексами по мере их создания. Обратите внимание, что это число отличается от идентификатора потока, используемого операционной системой Microsoft Windows.

При начале отладки текущий поток является тем, который вызвал текущее исключение или событие отладки (или активный поток при подключении отладчика к процессу). Этот поток остается текущим потоком, пока не укажите новый, используя команду **~s (Set Current Thread)** или используя окно "Процессы и потоки " в WinDbg.

Идентификаторы потоков обычно отображаются в виде префиксов команд. Обратите внимание, что не все подстановочные знаки доступны во всех командах, использующих идентификаторы потока.

An example of the ~[*Expression*] syntax would be `~[@$t0]`. В этом примере поток изменяется в зависимости от значения определяемого пользователем псевдорегистрирующего регистра. Этот синтаксис позволяет скриптам отладчика программно выбирать поток.

## Управление потоками в режиме ядра

В режиме ядра потоки нельзя управлять с помощью идентификаторов потоков. For more information about how to access thread-specific information in kernel mode, see Changing Contexts.

**Note** You can use the tilde character ( ~ ) to specify threads during user-mode debugging. В отладке в режиме ядра можно использовать тильду для указания процессоров. For more information about how to specify processors, see Multiprocessor Syntax.

# Process Syntax

18.07.2025

Многие команды отладчика имеют идентификаторы процесса в качестве параметров. Вертикальная полоса (|) отображается перед идентификатором процесса.

Идентификатор процесса может быть одним из следующих значений.

Развернуть таблицу

| Process identifier | Description |
|---|---|
| \|. | Текущий процесс. |
| \|# | Процесс, вызвавшего текущее исключение или событие отладки. |
| \|* | All processes. |
| \| *Number* | The process whose ordinal is *Number*. |
| \|~[*PID*] | The process whose process ID is *PID*. (Квадратные скобки обязательны, и вы не можете добавить пробел между тильдой (~) и открывающей скобкой.) |
| \|[*Expression*] | The process whose process ID is the integer to which the numerical *Expression* resolves. |

Процессы назначаются порядковые порядковые номера по мере их создания. Обратите внимание, что это число отличается от идентификатора процесса (PID), используемого операционной системой Microsoft Windows.

Текущий процесс определяет пространство памяти и набор используемых потоков. При начале отладки текущий процесс вызывает текущее событие исключения или отладки (или процесс, к которому подключен отладчик). Этот процесс остается текущим, пока не укажите новый с помощью команды **|s (Set Current Process)** или с помощью окна "Процессы и потоки " в WinDbg.

Идентификаторы процесса используются в качестве параметров в нескольких командах, часто в качестве префикса команды. Обратите внимание, что WinDbg и CDB могут отлаживать дочерние процессы, созданные исходным процессом. WinDbg и CDB также могут подключаться к нескольким несвязанным процессам.

An example of the |[*Expression*] syntax would be |[@$t0]. В этом примере процесс изменяется в зависимости от значения определяемого пользователем псевдорегистрирующего регистра. Этот синтаксис позволяет скриптам отладчика программно выбирать процесс.

## Управление процессами в режиме ядра

В режиме ядра нельзя управлять процессами с помощью идентификаторов процессов. For more information about how to access process-specific information in kernel mode, see Changing Contexts.

# System Syntax

18.07.2025

Многие команды отладчика имеют идентификаторы процесса в качестве параметров.

Перед системным идентификатором отображаются две вертикальные полосы (||). Системный идентификатор может быть одним из следующих значений.

Развернуть таблицу

| System identifier | Description |
|---|---|
| \|\|. | Текущая система |
| \|\|# | Система, вызвавшей текущее исключение или событие отладки. |
| \|\|* | All systems. |
| \|\| *ddd* | The system whose ordinal is *ddd*. |

Системы назначаются порядковые порядки в том порядке, в который отладчик подключается к ним.

При начале отладки текущая система — это система, которая вызвала текущее событие исключения или отладки (или тот, к которому недавно подключен отладчик). Эта система остается текущей системой, пока не укажите новую систему с помощью **||команда (задать текущую систему)** или с помощью окна "Процессы и потоки " в WinDbg.

## Example

В этом примере показано, как загружаются три файла дампа. Система 1 активна и система 2 вызвала событие отладки.

dbgcmd

```
||1:1:017> ||
   0 User mini dump: c:\notepad.dmp
.  1 User mini dump: c:\paint.dmp
#  2 User mini dump: c:\calc.dmp
```

# Remarks

To work with multiple systems, you can use the .opendump to debug multiple crash dumps at the same time. Дополнительные сведения о том, как управлять сеансом с несколькими целевыми объектами, см. в разделе "Отладка нескольких целевых объектов".

**Note** There are complications, when you debug live targets and dump targets together, because commands behave differently for each type of debugging. For example, if you use the **g (Go)** command when the current system is a dump file, the debugger begins executing, but you cannot break back into the debugger, because the break command is not recognized as valid for dump file debugging.

# Синтаксис многопроцессора

18.07.2025

KD и WinDbg в режиме ядра поддерживают отладку нескольких процессоров. Эту отладку можно выполнить на любой многопроцессорной платформе.

Процессоры нумеруются от *нуля до n*.

Если текущий процессор является процессором 0 (то есть, если это процессор, который в настоящее время вызвал активность отладчика), можно проверить другие не текущие процессоры (процессоры один через *n*). Однако вы не можете изменить ничего в не текущих процессорах. Просматривать их состояние можно только.

## Выбор обработчика

Для отображения номеров процессора текущего процессора можно использовать команду **echocpunum (показать номер ЦП** ). Выходные данные этой команды позволяют немедленно сообщать о работе с несколькими процессорами по тексту в командной строке отладки ядра.

В следующем примере **0:** перед запросом **kd>** указывает, что выполняется отладка первого процессора на компьютере.

dbgcmd

```
0: kd>
```

Используйте команду **~s (Изменить текущий процессор)** для переключения между процессорами, как показано в следующем примере.

dbgcmd

```
0: kd> ~1s
1: kd>
```

Теперь выполняется отладка второго процессора на компьютере.

Может потребоваться изменить процессоры в многопроцессорной системе, если вы столкнулись с сбоем и не можете понять трассировку стека. Возможно, разрыв произошел на другом процессоре.

## Указание процессоров в других командах

Номер процессора можно добавить перед несколькими командами. Это число не предшествует тильде (~), за исключением команды **~S** .

**Заметка** При отладке в пользовательском режиме тильда используется для указания потоков. Дополнительные сведения об этом синтаксисе см. в разделе "Синтаксис потока".

Идентификаторы процессора не нужно указывать явно. Вместо этого можно использовать числовое выражение, вычисляющееся в целое число, соответствующее идентификатору процессора. Чтобы указать, что выражение должно быть интерпретировано как обработчик, используйте следующий синтаксис.

```
dbgcmd
```

```
||[Expression]
```

В этом синтаксисе квадратные скобки являются обязательными, и *Выражение* обозначает любое числовое выражение, разрешающее целое число, соответствующее идентификатору процессора.

В следующем примере обработчик изменяется в зависимости от значения определяемого пользователем псевдорегистрирующего регистра.

```
dbgcmd
```

```
||[@$t0]
```

## Примеры

В следующем примере используется команда **k (Display Stack Backtrace)** для отображения трассировки стека из процессора два.

```
dbgcmd
```

```
1: kd> 2k
```

В следующем примере используется команда **r (Registers)** для отображения регистра **eax** процессора три.

```
dbgcmd
```

```
1: kd> 3r eax
```

Однако следующая команда дает синтаксическую ошибку, так как невозможно изменить состояние процессора, отличного от текущего процессора.

dbgcmd

```
1: kd> 3r eax=808080
```

## Точки останова

Во время отладки ядра команды **bp, bu, bm (Set Breakpoint)** и **ba (Break on Access)** применяются ко всем процессорам нескольких процессоров.

Например, если текущий процессор равен трем, можно ввести следующую команду, чтобы поместить точку останова в **SomeAddress**.

dbgcmd

```
1: kd> bp SomeAddress
```

Затем любой процессор (не только процессор один), который выполняется по этому адресу, вызывает ловушку точки прерывания.

## Отображение сведений о обработчике

Вы можете использовать расширение **!running** для отображения состояния каждого процессора на целевом компьютере. Для каждого процессора **!running** может также отображать текущие и следующие поля потоков из блока управления процессами (PRCB), состояние 16 встроенных очередей спинлока и трассировку стека.

Расширения **!cpuinfo** и **!cpuid** можно использовать для отображения сведений о самих процессорах.

# ; (Разделитель команд)

Статья • 26.10.2023

Символ точки с запятой ( ; ) используется для разделения нескольких команд в одной строке.

```
dbgcmd
```

```
Command1 ; Command2 [; Command3 ...]
```

## Параметры

*Command1*, *Command2*, ...

Выполняемые команды.

## Комментарии

Команды выполняются последовательно слева направо. Все команды в одной строке ссылаются на текущий поток, если не указано иное. Если команда вызывает выполнение потока, остальные команды в строке будут отложены до тех пор, пока поток не остановится в событии отладки.

За небольшим количеством команд не может следовать точка с запятой, так как они автоматически принимают всю оставшуюся часть строки в качестве аргумента. К ним относятся как aS (Задать псевдоним),$<(Выполнить файл скрипта) и любая команда, начинающаяся с токена * (спецификатор строки комментария).

Ниже приведен пример. При этом выполняется текущая программа в исходной строке 123, выводится значение счетчика, а затем возобновляется выполнение:

```
dbgcmd
```

```
0:000> g `:123`; ? poi(counter); g
```

# { } (разделитель блоков)

Статья • 26.10.2023

Пара фигурных скобок ( **{ }** ) используется для заключения блока инструкций в командную программу отладчика.

dbgcmd

```
Statements { Statements } Statements
```

## Дополнительные сведения

Сведения о командных программах отладчика и маркерах потока управления см. в разделе Использование командных программ отладчика.

## Комментарии

При вводе каждого блока вычисляются все псевдонимы в блоке. Если изменить значение псевдонима в какой-то момент в блоке команд, команды, последующие к этой точке, не будут использовать новое значение псевдонима, если они не находятся в подчиненном блоке.

Каждый блок должен начинаться с маркера потока управления. Если вы хотите создать блок с единственной целью оценки псевдонимов, следует добавить в него префикс маркера **.block** .

# ${ } (интерпретатор псевдонимов)

Статья • 26.10.2023

Знак доллара, за которым следует пара скобок ( ${ } ), вычисляет различные значения, связанные с указанным псевдонимом пользователя.

dbgcmd

```
Text ${Alias} Text
Text ${/d:Alias} Text
Text ${/f:Alias} Text
Text ${/n:Alias} Text
Text ${/v:Alias} Text
```

## Параметры

*Псевдоним*

Указывает имя псевдонима для расширения или вычисления. *Псевдоним* должен быть псевдонимом пользователя или значением *переменной* , используемым маркером FOREACH .

**/d**

Возвращает значение 1 или ноль в зависимости от того, определен ли псевдоним в данный момент. Если псевдоним определен, `${/d:Alias}` заменяется на 1; если псевдоним не определен, `${/d:Alias}` заменяется 0.

**/f**

Вычисляет эквивалент псевдонима, если псевдоним определен в данный момент. Если псевдоним определен, `${/f:Alias}` заменяется эквивалентным псевдонимом; если псевдоним не определен, `${/f:Alias}` заменяется пустой строкой.

**/N**

Вычисляет имя псевдонима, если псевдоним определен в данный момент. Если псевдоним определен, `${/n:Alias}` заменяется именем псевдонима; если псевдоним не определен, не заменяется, `${/n:Alias}` но сохраняет литеральное значение `${/n:Alias}`.

**/v**

Запрещает вычисление псевдонимов. Независимо от того, определен ли *псевдоним* , `${/v:Alias}` всегда сохраняет литеральное значение `${/v:Alias}`.

## Дополнительные сведения

Описание использования псевдонимов см. в разделе Использование псевдонимов.

# Комментарии

Если параметры не используются и псевдоним определен в данный момент, `${Alias}` заменяется эквивалентным псевдонимом. Если параметры не используются и псевдоним не определен, `${Alias}` всегда сохраняет литеральное значение `${Alias}`.

Одним из преимуществ использования токена ${ } является то, что псевдоним будет оцениваться, даже если он находится рядом с другими символами. Без этого маркера отладчик заменяет только псевдонимы, которые отделены от других маркеров пробелом.

Как указано, существуют обстоятельства, когда токен ${ } не заменяется ничем, но сохраняет свое литеральное значение. Это происходит, если параметр не используется, *а псевдоним* не определен, если используется параметр /n и *псевдоним* не определен, и всегда при использовании параметра /v. В таких случаях маркер сохраняет свое литеральное значение, включая знак доллара и фигурные скобки. Таким образом, если этот параметр используется в качестве параметра команды, возникает синтаксическая ошибка, если этот параметр не принимает произвольные текстовые строки.

Однако есть одно исключение из этого. Если вы используете `${/v:Alias}` в качестве первого параметра для команды как (Задать псевдоним) или aS (Задать псевдоним), этот маркер будет рассматриваться только как *псевдоним* строки, а не как строка `${/v:Alias}`. Это работает только с командами as, aS и ad и работает только при использовании параметра /v— он не будет работать с `${/n:Alias}` или `${Alias}` при сохранении литеральных значений.

*Псевдоним* должен быть псевдонимом пользователя или значением *переменной* , используемым маркером foreach , а не псевдонимом с фиксированным именем. Если в строке *Alias* есть псевдоним с фиксированным именем, он будет заменен до вычисления токена ${ }.

# $$ (описатель комментария)

Статья • 26.10.2023

Если два знака доллара ( `$$` ) отображаются в начале команды, то остальная часть строки обрабатывается как комментарий, если комментарий не заканчивается точкой с запятой.

dbgcmd

```
$$ [any text]
```

## Комментарии

Маркер `$$` анализируется так же, как и любая другая команда отладчика. Поэтому, если вы хотите создать комментарий после другой команды, перед маркером `$$` необходимо добавить точку с запятой.

Маркер `$$` приведет к тому, что текст будет игнорироваться до конца строки или до тех пор, пока не будет обнаружена точка с запятой. Точка с запятой завершает комментарий. Текст после точки с запятой анализируется как стандартная команда. Это отличается от * (описатель строки комментария), который делает оставшуюся часть строки комментарием, даже если есть точка с запятой.

Например, следующая команда отображает eax и ebx, но не ecx:

Консоль

```
0:000> r eax; $$ some text; r ebx; * more text; r ecx
```

Текст с префиксом маркеров `*` или `$$` не обрабатывается каким-либо образом. Если выполняется удаленная отладка, комментарий, введенный на сервере отладки, не будет отображаться в клиенте отладки и наоборот. Если вы хотите, чтобы текст примечания отображался в окне Команда отладчика таким образом, чтобы он был виден всем сторонам, следует использовать .echo (Echo Comment).

# * (описатель строки комментария)

Статья • 07.02.2024

Если символ звездочки ( `*` ) находится в начале команды, остальная часть строки обрабатывается как комментарий, даже если после нее появляется точка с запятой.

dbgcmd

```
    * [any text]
```

## Комментарии

Токен `*` анализируется, как и любая другая команда отладчика. Поэтому, если вы хотите создать комментарий после другой команды, перед маркером `*` необходимо добавить точку с запятой.

Маркер `*` будет игнорировать оставшуюся часть строки, даже если после нее появится точка с запятой. Это отличается от $$ (описатель комментариев), который создает комментарий, который может быть завершен точкой с запятой.

Например, следующая команда отображает eax и ebx, но не ecx:

Консоль

```
0:000> r eax; $$ some text; r ebx; * more text; r ecx
```

Текст с префиксом `*` или `$$` не обрабатывается каким-либо образом. При удаленной отладке комментарий, введенный на сервере отладки, не будет отображаться в клиенте отладки и наоборот. Если вы хотите, чтобы текст комментария отображался в окне Команда отладчика таким образом, который был виден всем сторонам, следует использовать .echo (Echo Comment).

# .Блок

Статья • 26.10.2023

Маркер **.block** не выполняет никаких действий; он используется исключительно для введения блока операторов.

dbgcmd

```
    Commands ; .block { Commands } ; Commands
```

## Дополнительные сведения

Сведения об использовании нового блока для оценки псевдонима см. в разделе Использование псевдонимов и **as, aS (задать псевдоним).**

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе Использование командных программ отладчика.

# Комментарии

Блоки команд заключены в фигурные скобки. При вводе каждого блока вычисляются все псевдонимы в блоке. Если изменить значение псевдонима в какой-то момент в блоке команд, последующие команды не будут использовать новое значение псевдонима, если они не находятся в подчиненном блоке.

Каждый блок должен начинаться с маркера потока управления. Если вы хотите создать блок с единственной целью оценки псевдонимов, следует добавить в него префикс маркера **.block** , так как этот маркер не имеет никакого действия, кроме как разрешить ввод блока.

# .Перерыв

Статья • 26.10.2023

Токен **.break** работает так же, как **ключевое слово прерывания** в C.

```
dbgcmd
```

```
 .for (...) { ... ; .if (Condition) .break ; ...}

.while (...) { ... ; .if (Condition) .break ; ...}

.do { ... ; .if (Condition) .break ; ...} (...)
```

## Дополнительные сведения

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе Использование командных программ отладчика.

# Комментарии

Токен **.break** можно использовать в любом цикле **.for**, **.while** или **.do** .

Так как нет маркера потока управления, эквивалентного инструкции C **goto** , обычно используется маркер **break** в условном **объекте .if** , как показано в примерах синтаксиса выше. Однако на самом деле это не требуется.

# .Поймать

Статья • 26.10.2023

Токен **.catch** используется для предотвращения завершения программы в случае возникновения ошибки.

Он не ведет себя так, как ключевое слово **catch** в C++.

```
dbgsyntax
```

```
    Commands ; .catch { Commands } ; Commands
```

## Дополнительные сведения

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе Использование командных программ отладчика.

# Комментарии

За токеном **.catch** следуют фигурные скобки, включающие одну или несколько команд.

Если команда в блоке **CATCH** создает ошибку, отображается сообщение об ошибке, все оставшиеся команды в фигурных скобках игнорируются, а выполнение возобновляется с первой командой после закрывающей фигурной скобки.

Если **параметр .catch** не используется, ошибка приведет к завершению всей командной программы отладчика.

Для выхода из блока **.catch** можно использовать **.leave**.

# .Продолжить

Статья • 26.10.2023

Токен **.continue** ведет себя так же, как **и ключевое слово continue** в C.

dbgsyntax

```
.for (...) { ... ; .if (Condition) .continue ; ... }

.while (...) { ... ; .if (Condition) .continue ; ... }

.do { ... ; .if (Condition) .continue ; ... } (...)
```

## Дополнительные сведения

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе Использование командных программ отладчика.

# Комментарии

Токен **.continue** можно использовать в любом цикле **.for**, **.while** или **.do** .

Так как нет маркера потока управления, эквивалентного инструкции C goto, обычно используется маркер **.continue** в условном **объекте .if** , как показано в примерах синтаксиса выше. Однако на самом деле это не требуется.

# .do

Статья • 05.04.2024

**Маркер** DO ведет себя как **ключевое слово do** в C, за исключением того, что слово "while" не используется перед условием.

```
dbgcmd
```

```
.do { Commands } (Condition)
```

## Элементы синтаксиса

*Команды*
Указывает одну или несколько команд, которые будут выполняться многократно, пока условие имеет значение true, но всегда будет выполняться по крайней мере один раз. Этот блок команд должен быть заключен в фигурные скобки, даже если он состоит из одной команды. Несколько команд должны быть разделены точкой с запятой, но окончательная команда перед закрывающей фигурной скобкой не должна следовать за точкой с запятой.

*Условие*
Указывает условие. Если значение равно нулю, оно обрабатывается как false; в противном случае значение true. Заключение условия *в скобках является необязательным*. *Условие* должно быть выражением, а не командой отладчика. Он будет оцениваться по умолчанию методом оценки выражений (MASM или C++). Дополнительные сведения см. в разделе "Синтаксис числовых выражений".

## Дополнительная информация

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе "Использование программ команд отладчика".

## Замечания

***Маркеры .break*** *и* ***.continue*** *можно использовать для выхода или перезапуска* блока команд.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# еще.

Статья • 08.07.2024

Маркер **.else** ведет себя как ключевое **слово else** в C.

```dbgcmd
.if (Condition) { Commands } .else { Commands }

.if (Condition) { Commands } .elsif (Condition) { Commands } .else {
Commands }
```

## Элементы синтаксиса

*Команды*
Указывает одну или несколько команд, которые будут выполняться условно. Этот блок команд должен быть заключен в фигурные скобки, даже если он состоит из одной команды. Несколько команд должны быть разделены точкой с запятой, но окончательная команда перед закрывающей фигурной скобкой не должна следовать за точкой с запятой.

## Дополнительная информация

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе "Использование программ команд отладчика".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Elsif

Статья • 08.07.2024

Токен **elsif** ведет себя так же, как и в **случае** сочетания ключевых слов в C.

dbgcmd

```
.if (Condition) { Commands } .elsif (Condition) { Commands }

.if (Condition) { Commands } .elsif (Condition) { Commands } .else {
Commands }
```

## Элементы синтаксиса

*Состояние*
Указывает условие. Если значение равно нулю, оно обрабатывается как false; в противном случае значение true. Заключение условия *в скобках является необязательным*. *Условие* должно быть выражением, а не командой отладчика. Он будет оцениваться по умолчанию методом оценки выражений (MASM или C++). Дополнительные сведения см. в разделе "Синтаксис числовых выражений".

*Команды*
Указывает одну или несколько команд, которые будут выполняться условно. Этот блок команд должен быть заключен в фигурные скобки, даже если он состоит из одной команды. Несколько команд должны быть разделены точкой с запятой, но окончательная команда перед закрывающей фигурной скобкой не должна следовать за точкой с запятой.

## Дополнительная информация

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе "Использование программ команд отладчика".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .для

Статья • 08.07.2024

Маркер **.for** работает как ключевое слово в C, за исключением того, что несколько команд добавочного увеличения должны быть разделены запятыми, а не запятыми.

dbgcmd

```
.for (InitialCommand ; Condition ; IncrementCommands) { Commands }
```

## Элементы синтаксиса

*InitialCommand*
Указывает команду, которая будет выполнена до начала цикла. Разрешена только одна начальная команда.

*Состояние*
Указывает условие. Если значение равно нулю, оно обрабатывается как false; в противном случае значение true. Заключение условия *в скобках является необязательным*. *Условие* должно быть выражением, а не командой отладчика. Он будет оцениваться по умолчанию методом оценки выражений (MASM или C++). Дополнительные сведения см. в разделе "Синтаксис числовых выражений".

*IncrementCommands*
Указывает одну или несколько команд, которые будут выполняться в конце каждого цикла. Если вы хотите использовать несколько команд добавок, разделяйте их точкой с запятой, но не заключайте их в фигурные скобки.

*Команды*
Указывает одну или несколько команд, которые будут выполняться многократно, пока условие имеет значение true. Этот блок команд должен быть заключен в фигурные скобки, даже если он состоит из одной команды. Несколько команд должны быть разделены точкой с запятой, но окончательная команда перед закрывающей фигурной скобкой не должна следовать за точкой с запятой.

## Дополнительная информация

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе "Использование программ команд отладчика".

# Замечания

Если все действия выполняются командами добавочного увеличения, можно полностью опустить *условие* и просто использовать пустую пару фигурных скобок.

Ниже приведен пример инструкции **.for** с несколькими командами добавочного увеличения:

```
dbgcmd

0:000> .for (r eax=0; @eax < 7; r eax=@eax+1; r ebx=@ebx+1) { .... }
```

**Маркеры *.break*** и ***.continue*** *можно использовать для выхода или перезапуска* блока команд.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .foreach

Статья • 08.07.2024

Маркер **foreach** анализирует выходные данные одной или нескольких команд отладчика и использует каждое значение в этом выходных данных в качестве входных данных для одной или нескольких дополнительных команд.

dbgcmd

```
.foreach [Options] ( Variable  { InCommands } ) { OutCommands }

.foreach [Options] /s ( Variable  "InString" ) { OutCommands }

.foreach [Options] /f ( Variable  "InFile" ) { OutCommands }
```

## Элементы синтаксиса

*Параметры* могут быть любым сочетанием следующих параметров:

**/pS** *InitialSkipNumber*
Приводит к пропуску некоторых начальных маркеров. *InitialSkipNumber* указывает количество выходных маркеров, которые не будут переданы указанным *OutCommands*.

**/ps** *SkipNumber*
Приводит к тому, что маркеры пропускаются многократно при каждом обработке команды. После каждого перехода маркера в указанный *OutCommands число маркеров*, равных значению *SkipNumber* , будет игнорироваться.

*Переменная*
Задает имя переменной. Эта переменная будет использоваться для хранения выходных данных каждой команды в *строке InCommands; можно ссылаться на переменную по имени в параметрах, передаваемых в OutCommands*. Любую буквенно-цифровые строки можно использовать, хотя и с помощью строки, которая также может передаваться для допустимого шестнадцатеричного числа или команды отладчика, не рекомендуется. Если имя, используемое для *переменной* , совпадает с существующей глобальной переменной, локальной переменной или псевдонимом, их значения не будут затронуты **командой foreach** .

*InCommands*
Задает одну или несколько команд, выходные данные которых будут

проанализированы; Результирующий маркер будет передан *в OutCommands*. Выходные данные из *InCommands* не отображаются.

*InString*
Используется с **/s**. Указывает строку, которая будет проанализирована; Результирующий маркер будет передан *в OutCommands*.

*InFile*
Используется с **/f**. Указывает текстовый файл, который будет проанализирован; Результирующий маркер будет передан *в OutCommands*. Имя *файла InFile* должно быть заключено в кавычки.

*OutCommands*
Указывает одну или несколько команд, которые будут выполняться для каждого маркера. Всякий раз, когда строка переменной будет заменена текущим маркером.

**Примечание.** Если строковая *переменная* отображается в *OutCommands, она должна быть окружена пробелами*. Если он находится рядом с любым другим текстом - даже скобками - он не будет заменен текущим значением токена, если только вы не используете **токен ${ } (псевдоним интерпретатора).**

# Дополнительная информация

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе "Использование программ команд отладчика".

# Замечания

Если выходные данные из *InCommands*, *строка InString* или *файл InFile* анализируются, любое количество пробелов, вкладок или возвращаемых каретки обрабатывается как один разделитель. Каждый из результирующего фрагмента текста используется для замены *переменной* при отображении в *OutCommands*.

Ниже приведен пример инструкции foreach, которая использует **команду dds** для каждого токена, найденного в файле *myfile.txt*:

```
dbgcmd
```

```
0:000> .foreach /f ( place "g:\myfile.txt") { dds place }
```

Флаги **/pS** и **/ps** можно использовать для передачи только определенных маркеров указанным *OutCommands*. Например, следующая инструкция пропускает первые

два маркера в файле myfile.txt, а затем передает третий в **dds**. После каждого переданного маркера будет пропускаться четыре маркера. Результатом является то, что **dds** будут использоваться с 3-м, 8-м, 13-м, 18-м и 23-м маркерами и т. д.

dbgcmd

```
0:000> .foreach /pS 2 /ps 4 /f ( place "g:\myfile.txt") { dds place }
```

Дополнительные примеры использования маркера **foreach** см . в примерах программы команд отладчика.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .если

Статья • 08.07.2024

Маркер **.if** работает так же, как ключевое **слово if** в C.

dbgcmd

```
.if (Condition) { Commands }

.if (Condition) { Commands } .else { Commands }

.if (Condition) { Commands } .elsif (Condition) { Commands }

.if (Condition) { Commands } .elsif (Condition) { Commands } .else {
Commands }
```

## Элементы синтаксиса

*Состояние*
Указывает условие. Если значение равно нулю, оно обрабатывается как false; в противном случае значение true. Заключение условия *в скобках является необязательным*. *Условие* должно быть выражением, а не командой отладчика. Он будет оцениваться по умолчанию методом оценки выражений (MASM или C++). Дополнительные сведения см. в разделе "Синтаксис числовых выражений".

*Команды*
Указывает одну или несколько команд, которые будут выполняться условно. Этот блок команд должен быть заключен в фигурные скобки, даже если он состоит из одной команды. Несколько команд должны быть разделены точкой с запятой, но окончательная команда перед закрывающей фигурной скобкой не должна следовать за точкой с запятой.

## Дополнительная информация

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе "Использование программ команд отладчика".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .Оставить

Статья • 26.10.2023

Маркер **.leave** используется для выхода из блока **CATCH** .

```dbgcmd
.catch { ... ; .if (Condition) .leave ; ... }
```

## Дополнительные сведения

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе Использование командных программ отладчика.

# Комментарии

При обнаружении маркера **.leave** в блоке **.catch** программа выходит из блока и выполнение возобновляется с помощью первой команды после закрывающей фигурной скобки.

Так как не существует маркера потока управления, эквивалентного инструкции C **goto** , обычно используется маркер **.leave** в условном **объекте .if** , как показано в примерах синтаксиса выше. Однако на самом деле это не требуется.

# .printf

Статья • 08.07.2024

Маркер **.printf ведет себя как оператор printf** в C.

```
dbgcmd
```

```
.printf [/D] [Option] "FormatString" [, Argument , ...]
```

## Элементы синтаксиса

**/D**
Указывает, что строка формата содержит язык разметки отладчика (DML).

*Выбор*
(только WinDbg) Указывает тип текстового сообщения, которое WinDbg должен интерпретировать FormatString как. WinDbg назначает каждому типу сообщения командного окна отладчика цвет фона и текста; При выборе одного из этих параметров сообщение отображается в соответствующих цветах. По умолчанию текст отображается как обычное сообщение.

Имеются следующие варианты.

Развернуть таблицу

| Параметр | Тип сообщения | Заголовок цветов в диалоговом окне "Параметры" |
|---|---|---|
| /od | отладчик | Командное окно уровня отладки |
| /oD | запрос отладчика | Окно командной строки уровня отладки |
| /oe | error | Окно команд уровня ошибки |
| /on | Обычная | Командное окно "Обычный уровень" |
| /op | prompt | Окно командной строки уровня командной строки |
| /oP | регистры запроса | Окно командной строки "Запрос регистрирует |

| Параметр | Тип сообщения | Заголовок цветов в диалоговом окне "Параметры" |
|---|---|---|
| | | уровень" |
| /OC | Символы | Командное окно уровня символов |
| /ov | verbose | Командное окно уровня детализации |
| /ow | предупреждений (не рекомендуется) | Окно команд уровня предупреждения |

*FormatString*
Указывает строку формата, как в **printf**. Как правило, символы преобразования работают точно так же, как и в C. Для символов преобразования с плавающей запятой 64-разрядный аргумент интерпретируется как 32-разрядное число с плавающей запятой, если **не используется модификатор l** .

Модификатор I64 можно добавить, чтобы указать, что значение должно интерпретироваться как 64-разрядные. Например, "%I64x" можно использовать для печати 64-разрядного шестнадцатеричного числа.

Символ преобразования %p поддерживается, но он представляет указатель в виртуальном адресном пространстве целевого объекта. Он не должен иметь модификаторов и использует внутреннее форматирование адресов отладчика. Помимо стандартных описателей формата printf, поддерживаются следующие дополнительные символы преобразования.

Развернуть таблицу

| Символ | Тип аргумента | Аргумент | Печатный текст |
|---|---|---|---|
| %p | ULONG64 | Указатель на виртуальное адресное пространство целевого объекта. | Значение указателя. |
| %N | DWORD_PTR (32 или 64 бита в зависимости от архитектуры узла) | Указатель на виртуальное адресное пространство узла. | Значение указателя. (Это эквивалентно стандартному символу C %p.) |
| %ma | ULONG64 | Адрес строки ASCII, завершаемой null, в | Указанная строка. |

| Символ | Тип аргумента | Аргумент | Печатный текст |
|---|---|---|---|
| | | виртуальном адресном пространстве целевого объекта. | |
| %mu | ULONG64 | Адрес строки Юникода, завершаемой null, в виртуальном адресном пространстве целевого объекта. | Указанная строка. |
| %msa | ULONG64 | Адрес структуры ANSI_STRING в виртуальном адресном пространстве целевого объекта. | Указанная строка. |
| %msu | ULONG64 | Адрес структуры UNICODE_STRING в виртуальном адресном пространстве целевого объекта. | Указанная строка. |
| %y | ULONG64 | Адрес символа отладчика в виртуальном адресном пространстве целевого объекта. | Строка, содержащая имя указанного символа (и смещение, если таковое имеется). |
| %ly | ULONG64 | Адрес символа отладчика в виртуальном адресном пространстве целевого объекта. | Строка, содержащая имя указанного символа (и смещение, если таковые имеются), а также любые доступные сведения о исходной строке. |

*Аргументы*

Задает аргументы для строки форматирования, как в **printf**. Число указанных аргументов должно соответствовать количеству символов преобразования в *FormatString*. Каждый аргумент — это выражение, которое будет оцениваться по

умолчанию методом оценки выражений (MASM или C++). Дополнительные сведения см. в разделе "Синтаксис числовых выражений".

## Дополнительная информация

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе "Использование программ команд отладчика".

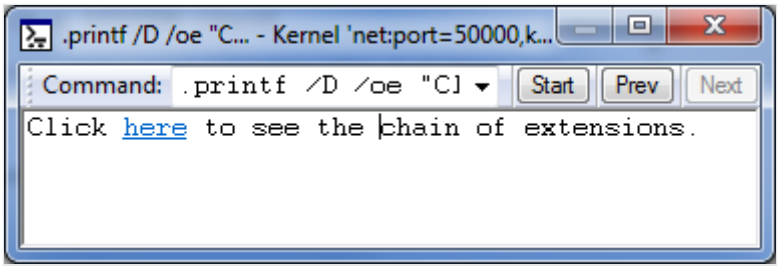
## Замечания

Параметры цвета, которые можно выбрать с помощью *параметра* "*Параметры* ", по умолчанию имеют черный текст на белом фоне. Чтобы лучше использовать эти параметры, необходимо сначала использовать **представление | Параметры, чтобы открыть диалоговое окно "Параметры** " и изменить параметры цвета для сообщений командного окна отладчика.

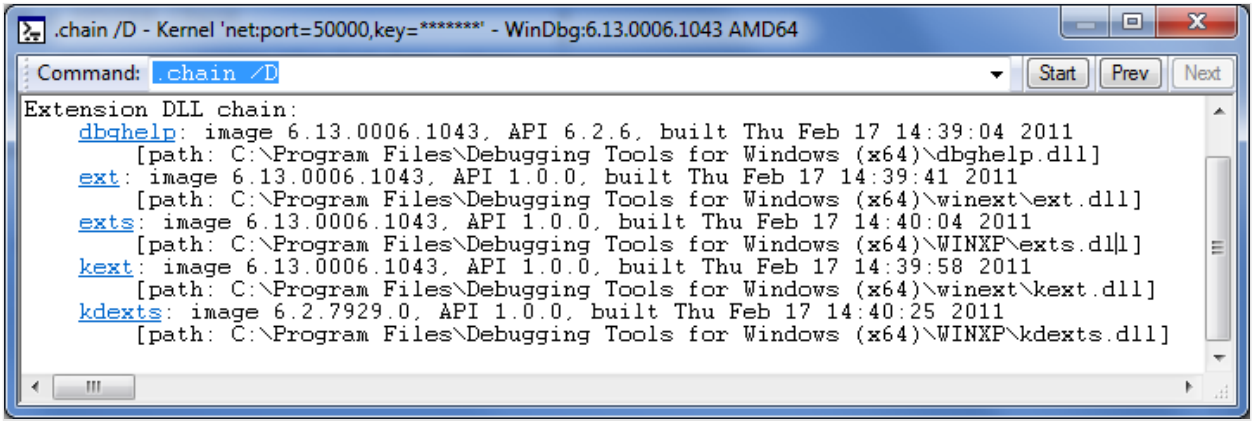
В следующем примере показано, как включить тег DML в строку формата.

dbgcmd

```
.printf /D "Click <link cmd=\".chain /D\">here</link> to see extensions
DLLs."
```




Выходные данные, показанные на предыдущем рисунке, имеют ссылку, которую можно щелкнуть, чтобы выполнить команду, указанную в теге `<link>` . На следующем рисунке показан результат нажатия ссылки.



Сведения о тегах DML см. в dml.doc в папке установки средств отладки для Windows.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .A

Статья • 05.04.2024

Маркер **.while ведет себя как в то время** как ключевое слово в C.

```dbgcmd
.while (Condition) { Commands }
```

## Элементы синтаксиса

*Условие*
Указывает условие. Если значение равно нулю, оно обрабатывается как false; в противном случае значение true. Заключение условия *в скобках является необязательным*. *Условие* должно быть выражением, а не командой отладчика. Он будет оцениваться по умолчанию методом оценки выражений (MASM или C++). Дополнительные сведения см. в разделе "Синтаксис числовых выражений".

*Команды*
Указывает одну или несколько команд, которые будут выполняться многократно, пока условие имеет значение true. Этот блок команд должен быть заключен в фигурные скобки, даже если он состоит из одной команды. Несколько команд должны быть разделены точкой с запятой, но окончательная команда перед закрывающей фигурной скобкой не должна следовать за точкой с запятой.

## Дополнительная информация

Сведения о других маркерах потока управления и их использовании в программах команд отладчика см. в разделе "Использование программ команд отладчика".

## Замечания

**Маркеры *.break*** *и* ***.continue*** *можно использовать для выхода или перезапуска* блока команд.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Commands

18.07.2025

В этом разделе ссылки рассматриваются различные команды отладчика, которые можно использовать в CDB, KD и WinDbg.

- ВВОД (повторная последняя команда)
- $, $<><, $$<, $$><> a< (запуск файла скрипта)
- ? (Command Help)
- ? (Evaluate Expression)
- ?? (Оценка выражения C++ )
- # (поиск шаблона дизассембли)
- || (Состояние системы)
- ||s (установка текущей системы)
- | (Состояние процесса)
- |s (установка текущего процесса)
- ~ (состояние треда)
- ~e ( командаThread-Specific)
- ~f (замораживание потока)
- ~u (unfreeze Thread)
- ~n (приостановка потока)
- ~m (возобновление потока)
- ~s (задать текущий поток)
- ~s (изменение текущего процессора)
- a (Assemble)
- ad (Delete Alias)
- ah (обработка утверждений)
- al (псевдонимы списка)
- как aS (set Alias)
- ba (перерыв на доступе)
- bc (точка останова clear)
- bd (отключение точки останова)
- be (включить точку останова)
- bl (список точек останова)
- bp, bu, bm (Установка точки останова)
- br (переназначка точки останова)
- bs (обновление команды точки останова)
- bsc (обновление условной точки останова)
- c (сравнение памяти)
- d, da, db, dc, dd, dD, df, dp, dq, du, dw, dW, dyb, dyd (память отображения)

- dda, ddp, ddu, dpa, dpp, dpu, dqa, dqp, dqu (отображаемая память со ссылкой)
- dds, dps, dqs (отображаемые слова и символы)
- dg (селектор отображения)
- dl (отображаемый связанный список)
- ds, dS (отображаемая строка)
- dt (тип отображения)
- dtx (тип отображения — сведения о расширенной объектной модели отладчика)
- dv (отображение локальных переменных)
- dx (выражение объектной модели отладчика)
- e, ea, eb, ed, eD, ef, ep, eq, eu, ew, eza, ezu (ввод значений)
- f, fp (заполнение памяти)
- g (Go)
- gc (переход с условной точки останова)
- gh (Go с обработкой исключений)
- gn, gN (Go с исключением не обрабатывается)
- gu (Go Up)
- ib, iw, id (входные данные из порта)
- j (Execute If - Else)
- k, kb, kc, kd, kp, kP, kv (Отобразить трассировку стека)
- l+, l- (установка параметров источника)
- ld (символы загрузки)
- lm (список загруженных модулей)
- ln (список ближайших символов)
- ls, lsa (список исходных строк)
- lsc (список текущего источника)
- lse (редактор источника запуска)
- lsf, lsf- (загрузка или выгрузка исходного файла)
- lsp (число исходных строк)
- m (перемещение памяти)
- n (настройка базы номеров)
- ob, ow, od (выходные данные в порт)
- p (Step)
- pa (шаг к адресу)
- pc (шаг к следующему вызову)
- pct (шаг к следующему вызову или возврату)
- ph (шаг к следующей ветвления инструкции)
- pt (шаг к следующему возврату)
- q, qq (выход)
- qd (Выход и отсоединение)
- r (Registers)
- rdmsr (чтение MSR)

- rm (регистрация маски)
- s (память поиска)
- so (настройка параметров отладки ядра)
- sq (установить тихий режим)
- ss (set Symbol Suffix)
- sx, sxd, sxe, sxi, sxn, sxr, sx- (set Exceptions)
- t (Trace)
- ta (трассировка по адресу)
- tb (трассировка до следующей ветви)
- tc (трассировка до следующего вызова)
- tct (трассировка до следующего вызова или возврата)
- th (трассировка до следующей ветви инструкции)
- tt (трассировка до следующего возврата)
- u (Unassemble)
- uf (Unassemble Function)
- up (Unassemble from Physical Memory)
- your (Unassemble Real Mode BIOS)
- ux (Unassemble x86 BIOS)
- vercommand (показать командную строку отладчика)
- версия (показать версию отладчика)
- vertarget (показать версию целевого компьютера)
- wrmsr (запись MSR)
- wt (данные трассировки и просмотра)
- x (проверка символов)
- z (выполнение в то время как)

# ENTER (повторить последнюю команду)

Статья • 09.05.2025

Клавиша ENTER повторяет последнюю введенную команду.

```
dbgcmd

ENTER
```

## Окружающая среда

Развернуть таблицу

| Товар | Описание |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Цели | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

В CDB, KD и WinDbg самостоятельное нажатие клавиши ENTER в командной строке отладчика повторно выполняет ранее введенную команду. Дополнительные сведения см. в разделе Использование команд отладчика.

Если вы хотите создать пробелы в окне команд отладчика, используйте маркер *** (спецификатор строки комментария),** а затем нажмите клавишу ENTER несколько раз.

# $<, $><, $$<, $$>< a >(запуск файла скрипта< )

Статья • 11.11.2023

Команды **$<**, **$><**, **$$<**, **$$><**и **$$>a<** считывают содержимое указанного файла скрипта и используют его содержимое в качестве входных данных команды отладчика.

dbgcmd

```
$<Filename
$><Filename
$$<Filename
$$><Filename
$$>a<Filename [arg1 arg2 arg3 ...]
```

## Параметры

*Имени файла* Указывает файл, содержащий допустимый текст команды отладчика. Имя файла должно соответствовать соглашениям об именах файлов Microsoft Windows. Имя файла может содержать пробелы.

*argn*

Задает любое количество строковых аргументов для передачи отладчиком в скрипт. Перед выполнением скрипта отладчик заменит любую строку в формате ${$arg*n*} в файле скрипта соответствующим *аргументом* . Аргументы не могут содержать кавычки или точки с запятой. Несколько аргументов должны быть разделены пробелами; Если аргумент содержит пробел, он должен быть заключен в кавычки. Все аргументы являются необязательными.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

Токены **$$<** и **$<** выполняют команды, которые находятся в файле скрипта буквально. Однако с **помощью $<** можно указать любое имя файла, включая имя, содержащее точку с запятой. Так как **$<** позволяет использовать точки с запятой в имени файла, нельзя сцепить **$<** с другими командами отладчика, так как точку с запятой нельзя использовать как разделитель команд, так и как часть имени файла.

Токены **$$><** и **$><** выполняют команды, которые находятся в файле скрипта буквально, то есть открывают файл скрипта, заменяют все возвращаемые каретки точкой с запятой и выполняют результирующий текст в виде одного блока команд. Как и в случае с **$<** , рассмотренным ранее, вариант **$><** разрешает имена файлов, содержащие точки с запятой, что означает, что вы не можете сцепить **$><** с другими командами отладчика.

Токены **$$><** и **$><** полезны при выполнении скриптов, содержащих программы команд отладчика. Дополнительные сведения об этих программах см. в разделе Использование командных программ отладчика.

Если у вас нет имен файлов, содержащих точки с запятой, не нужно использовать **$<** или **$><**.

Токен **$$>a<** позволяет отладчику передавать аргументы в скрипт. Если *filename* содержит пробелы, оно должно быть заключено в кавычки. Если указано слишком много аргументов, избыточные аргументы игнорируются. Если аргументов указано слишком мало, любой токен в исходном файле формы ${$arg*n*}, где *n* больше числа предоставленных аргументов, останется в его литеральной форме и не будет заменен на ничего. Для выполнения этой команды можно использовать точку с запятой и дополнительные команды. наличие точки с запятой завершает список аргументов.

Когда отладчик выполняет файл скрипта, команды и их выходные данные отображаются в окне Команда отладчика. По достижении конца файла скрипта элемент управления возвращается отладчику.

В следующей таблице показано, как можно использовать эти маркеры.

| Токен | Разрешает имена файлов, содержащих точки с запятой | Разрешает объединение дополнительных команд, разделенных точкой с запятой | Конденсируется до одного блока команд | Разрешает аргументы скрипта |
|---|---|---|---|---|
| $< | Да | Нет | Нет | Нет |
| $>< | Да | Нет | Да | Нет |
| $$< | Нет | Да | Нет | Нет |
| $$>< | Нет | Да | Да | Нет |
| $$>a< | Нет | Да | Да | Да |

Команды $<, $><, $$< и $$>< повторяют команды, содержащиеся в файле скрипта, и отображают выходные данные этих команд. Команда $$>a< не соответствует командам, найденным в файле скрипта, а просто отображает их выходные данные.

Файлы скриптов могут быть вложенными. Если отладчик обнаруживает один из этих маркеров в файле скрипта, выполнение перемещается в новый файл скрипта и возвращается в предыдущее расположение после завершения нового файла скрипта. Скрипты также можно вызывать рекурсивно.

В WinDbg можно вставить дополнительный текст команды в окно Команда отладчика.

# Примеры

В следующем примере показано, как передать аргументы в файл скрипта, Myfile.txt. Предположим, что файл содержит следующий текст:

Консоль

```
.echo The first argument is ${$arg1}.
.echo The second argument is ${$arg2}.
```

Затем можно передать аргументы в этот файл с помощью следующей команды:

Консоль

```
0:000> $$>a<myfile.txt myFirstArg mySecondArg
```

Результатом выполнения этой команды будет:

Консоль

```
The first argument is myFirstArg.
The second argument is mySecondArg.
```

Ниже приведен пример того, что происходит при указании неправильного числа аргументов. Предположим, что файл My Script.txt содержит следующий текст:

Консоль

```
.echo The first argument is ${$arg1}.
.echo The fifth argument is ${$arg5}.
.echo The fourth argument is ${$arg4}.
```

Затем следующая командная строка с разделителями точкой с запятой выдает выходные данные таким образом:

Консоль

```
0:000> $$>a< "c:\binl\my script.txt" "First one" Two "Three More" Four; recx
The first argument is First one.
The fifth argument is ${$arg5}.
The fourth argument is Four.
ecx=0021f4ac
```

В предыдущем примере имя файла заключено в кавычки, так как оно содержит пробел, а аргументы, содержащие пробелы, также заключены в кавычки. Несмотря на то, что скрипт ожидает пятый аргумент, точка с запятой завершает команду $$> **после**< четвертого аргумента.

# ? (Справка по командам)

Статья • 07.02.2024

Знак вопроса (**?**) отображает список всех команд и операторов.

**Примечание** Вопросительный знак сам по себе отображает справку по команде. Синтаксис **выражения ?** вычисляет данное выражение.

```
dbgcmd
```

```
?
```

## Среда

Развернуть таблицу

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Комментарии

Для получения дополнительных сведений о стандартных командах используйте **?**. Дополнительные сведения о метакоманд см. в **справке**. Дополнительные сведения о командах расширения см. в разделе **!help**.

# ? (Вычислять выражение)

Статья • 08.07.2024

Команда вопросительного знака (`?`) вычисляет и отображает значение выражения.

Сам по себе вопросительный знак (?) отображает справку по команде. Команда `?` *выражения* вычисляет заданное выражение.

dbgcmd

```
? Expression
```

## Параметры

*Выражение*

Указывает выражение для вычисления.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Входные и выходные данные `?` команды зависят от того, используется ли синтаксис выражения MASM или синтаксис выражения C++. Дополнительные сведения об этих типах синтаксиса выражений см. в разделе "Оценка выражений" и "Синтаксис числовых выражений".

Если используется синтаксис MASM, входные и выходные данные зависят от текущего радикса. Чтобы изменить радикс, используйте команду n (set Number Base).

Команда `?` вычисляет символы в выражении в контексте текущего потока и процесса.

Некоторые строки могут содержать escape-адреса, такие как `\n`, `\r` `\"` и `\b`, которые должны быть прочитаны буквально, а не интерпретированы вычислителем. Если экран в строке интерпретируется оценщиком, могут возникать ошибки в оценке. Например:

Консоль

```
0:000> as AliasName c:\dir\name.txt
0:000> al
  Alias            Value
 -------          -------
 AliasName        c:\dir\name.txt
0:001> ? $spat( "c:\dir\name.txt", "*name*" )
Evaluate expression: 0 = 00000000

0:001> ? $spat( "${AliasName}", "*name*" )
Evaluate expression: 0 = 00000000

0:001> ? $spat( "c:\dir\", "*filename*" )
Syntax error at '( "c:\dir\", "*filename*" )
```

В первых двух примерах, даже если строка соответствует шаблону, средство оценки возвращает значение FALSE. В третьем средство оценки не может выполнить сравнение, так как строка заканчивается обратной косой чертой ( `\` ) и поэтому `\"` преобразуется вычислителем.

Чтобы получить средство оценки для интерпретации строки буквально, необходимо использовать `@"String"` синтаксис. В следующем примере кода показаны правильные результаты:

Консоль

```
0:000> ? $spat( @"c:\dir\name.txt", "*name*" )
Evaluate expression: 1 = 00000000`00000001

0:000> ? $spat( @"${AliasName}", "*name*" )
Evaluate expression: 1 = 00000000`00000001

0:001> ? $spat( @"c:\dir\", "*filename*" )
Evaluate expression: 0 = 00000000
```

В предыдущих примерах оператор MASM проверяет первую строку, `$spat` чтобы определить, соответствует ли она (без учета регистра) шаблон второй строки.

Дополнительные сведения о операторах MASM см. в разделе "Номера и операторы MASM".

## См. также

?? (Оценка выражения C++ )

Форматы .форматов (отображение числовых форматов)

Номера и операторы MASM

Номера и операторы C++

Выражения MASM и выражения C++

Примеры смешанных выражений

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ?? (Оценка выражения C++)

Статья • 26.10.2023

Команда двойного вопросительного знака (**??**) вычисляет и отображает значение выражения в соответствии с правилами выражений C++.

dbgcmd

```
?? Expression
```

## Параметры

*Выражение*

Указывает выражение C++, которое требуется вычислить. Дополнительные сведения о синтаксисе см. в разделе Числа и операторы C++.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Комментарии

Команда **??** вычисляет символы в выражении в контексте текущего потока и процесса.

Если вы хотите оценить часть выражения в соответствии с правилами выражения MASM, заключите эту часть в круглые скобки и добавьте две части в знаки ( `@@` ) перед ней. Дополнительные сведения о выражениях MASM и C++ см. в разделе Вычисление выражений и синтаксис числовых выражений.

## См. также раздел

? (Вычисление выражения)

.formats (показать числовые форматы)

Вычисление выражений

Синтаксис числовых выражений

# # (поиск шаблона дизассемблы)

Статья • 26.10.2023

Команда числового знака (#) выполняет поиск указанного шаблона в коде дизассембля.

dbgcmd

```
# [Pattern] [Address [ L Size ]]
```

## Параметры

*Шаблон*

Указывает шаблон для поиска в коде дизассемблата. *Шаблон* может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе см. в разделе Синтаксис подстановочных знаков строки. Если вы хотите включить пробелы в *шаблон*, необходимо заключить шаблон в кавычки. В шаблоне регистр не учитывается. Если вы ранее использовали # команду и опустили *Шаблон*, команда повторно использует последний использованный шаблон.

*Адрес*

Указывает адрес, с которого начинается поиск. Дополнительные сведения о синтаксисе см. в разделе Синтаксис адреса и диапазона адресов.

*Размер*

Указывает количество инструкций для поиска. Если не указать *размер*, поиск продолжается до первого совпадения.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения об отладке сборок и связанных командах см. в разделе Отладка в режиме сборки.

# Комментарии

Если вы ранее использовали # команду и опустили *Адрес*, поиск начинается там, где был завершен предыдущий поиск.

Эта команда выполняет поиск в дизассемблированном тексте по указанному шаблону. Эту команду можно использовать для поиска имен регистров, констант или любой другой строки, которая отображается в выходных данных дизассемблата. Вы можете повторить команду без параметра *Address* , чтобы найти последовательные вхождения шаблона.

Инструкции дизассемблирования можно просмотреть с помощью команды u (Unassemble) или с помощью окна дизассемблирования в WinDbg. Дизассемблированное отображение содержит до четырех частей: смещение адреса, двоичный код, мнемонический язык ассемблера и сведения о языке сборки. В следующем примере показано возможное отображение.

```
dbgcmd
```

```
0040116b    45          inc         ebp
0040116c    fc          cld
0040116d    8945b0      mov         eax,[ebp-0x1c]
```

Команда # может искать текст в любой части дизассембличного отображения. Например, можно использовать # eax 0040116b, чтобы найти инструкцию `mov eax,[ebp-0x1c]` по адресу 0040116d. Следующие команды также находят эту инструкцию.

```
dbgcmd
```

```
#  [ebp?0x  0040116b
#  mov  0040116b
#  8945*  0040116b
#  116d  0040116b
```

Однако невозможно выполнить поиск `mov eax*` как единое целое, так как mov и eax отображаются в разных частях экрана. Вместо этого используйте `mov*eax`.

В качестве дополнительного примера можно выполнить следующую команду для поиска первой ссылки на функцию strlen после main точки входа.

dbgcmd

```
# strlen main
```

Аналогичным образом можно выполнить следующие две команды, чтобы найти первую инструкцию jnz после адреса 0x779F9FBA а затем найти следующую инструкцию jnz.

dbgcmd

```
# jnz 779f9fba#
```

Если не указать *шаблон* или *адрес*, их значения зависят от предыдущего # использования команды. Если вы опустите любой из параметров при первом выполнении # команды, поиск не выполняется. Однако значения *Pattern* и *Address* инициализируются даже в этой ситуации.

Если включить *шаблон* или *адрес*, для него будет задано введенное значение. Если пропустить *адрес*, он инициализируется текущим значением счетчика программы. Если опустить *шаблон*, он инициализируется пустым шаблоном.

# || (Состояние системы)

Статья • 26.10.2023

Команда с двойной вертикальной чертой (||) выводит состояние для указанной системы или для всех систем, которые в настоящее время отлаживаются.

Не путайте эту команду с **| (Состояние процесса)** Команды.

dbgcmd

```
|| System
```

## Параметры

*Системные функции*

Указывает отображаемую систему. Если этот параметр не указан, отображаются все отлаживаемые системы. Дополнительные сведения о синтаксисе см. в разделе Синтаксис системы.

### Среда

| Элемент | Описание |
|---|---|
| Режимы | Отладка нескольких целевых целей |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Комментарии

Команда || полезна только при отладке нескольких целевых объектов. Многие, но не все сеансы отладки с несколькими целевыми объектами используют несколько систем. Дополнительные сведения об этих сеансах см. в разделе Отладка нескольких целевых объектов.

Каждый системный список содержит имя сервера и сведения о протоколе. Система, в которую работает отладчик, определяется как **<локальная>**.

В следующих примерах показано, как использовать эту команду. Следующая команда отображает все системы.

```
dbgcmd

3:2:005> ||
```

Следующая команда также отображает все системы.

```
dbgcmd

3:2:005> ||*
```

Следующая команда отображает текущую активную систему.

```
dbgcmd

3:2:005> ||.
```

Следующая команда отображает систему, в которой было последнее исключение или прерывание.

```
dbgcmd

3:2:005> ||#
```

Следующая команда отображает системный номер 2.

```
dbgcmd

3:2:005> ||2
```

# ||s (задать текущую систему)

Статья • 26.10.2023

**||команда s** задает или отображает текущий системный номер.

```
dbgcmd
```

```
||System s
|| s
```

Не путайте эту команду с **командами s (Search Memory),~s (Change Current Processor),~s (Set Current Thread)** или **|s (Set Current Process).**

## Параметры

*Системные функции*

Указывает системный номер для активации. Дополнительные сведения о синтаксисе см. в разделе Синтаксис системы.

### Среда

| Элемент | Описание |
|---|---|
| Режимы | Отладка нескольких целевых целей |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Комментарии

**||Команда s** полезна при отладке нескольких целевых объектов или работе с несколькими файлами дампа. Дополнительные сведения об этих типах сеансов см. в разделе Отладка нескольких целевых объектов.

Если вы используете **||синтаксис,** отладчик отображает сведения о текущей системе.

Эта команда также дизассемблировать текущую инструкцию для текущей системы, процесса и потока.

**Примечание** При отладке динамических целевых объектов и целевых объектов дампа возникают сложности, так как команды ведут себя по-разному для каждого типа отладки. Например, если вы используете команду **g (Go),** когда текущая система является файлом дампа, отладчик начнет выполнение, но вы не сможете вернуться к отладчику, так как команда break не распознается как допустимая для отладки файла дампа.

# | (Состояние процесса)

Статья • 26.10.2023

Команда конвейера (|) отображает состояние для указанного процесса или для всех процессов, которые вы выполняете отладку в данный момент.

Не путайте эту команду с **|| (Состояние системы)** Команды.

```
dbgcmd

| Process
```

## Параметры

*Процесс*

Указывает отображаемый процесс. Если этот параметр не указан, отображаются все отлаживаемые процессы. Дополнительные сведения о синтаксисе см. в разделе Синтаксис процесса.

### Среда

| Элемент | Описание |
|---|---|
| Режимы | Только в пользовательском режиме |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

### Дополнительные сведения

Дополнительные сведения и другие методы отображения или управления процессами и потоками см. в разделе Управление процессами и потоками.

## Комментарии

Указать процессы можно только в пользовательском режиме.

Символ процесса можно добавить перед несколькими командами. Дополнительные сведения о значении канала (|), за которым следует команда, см. в

записи самой команды.

Если вы не включили отладку дочерних процессов при запуске сеанса отладки, существует только один процесс, доступный отладчику.

В следующих примерах показано, как использовать эту команду. Следующая команда отображает все процессы.

```
dbgcmd
```

```
2:005> |
```

Следующая команда также отображает все процессы.

```
dbgcmd
```

```
2:005> |*
```

Следующая команда отображает текущий активный процесс.

```
dbgcmd
```

```
2:005> |.
```

Следующая команда отображает процесс, который изначально вызвал исключение (или который изначально присоединен отладчик).

```
dbgcmd
```

```
2:005> |#
```

Следующая команда отображает номер процесса 2.

```
dbgcmd
```

```
2:005> |2
```

Предыдущая команда отображает следующие выходные данные.

```
dbgcmd
```

```
0:002> |
#  0 id: 224   name: myprog.exe
```

```
    1 id: 228   name: onechild.exe
  . 2 id: 22c   name: anotherchild.exe
```

В первой строке выходных данных 0 — десятичное число процесса, 224 — шестнадцатеричный идентификатор процесса, а *Myprog.exe* — имя приложения процесса. Период (.) перед процессом 2 означает, что этот процесс является текущим. Знак номера (#) перед процессом 0 означает, что этот процесс изначально вызывал исключение или к которому подключен отладчик.

# |s (установка текущего процесса)

Статья • 05.04.2024

Наборы **команд |s** или отображает текущий номер процесса.

Не путайте эту команду с s (память поиска), ~s (изменение текущего процессора), **~s (задать текущий поток) или ||команда s (Задать текущую систему).**

dbgcmd

```
|Process s
|s
```

## Параметры

*Обработать*
Указывает процесс установки или отображения. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис процесса".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только в пользовательском режиме |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о других методах отображения или управления процессами и потоками см. в разделе "Управление процессами и потоками".

## Замечания

Процессы можно указать только в пользовательском режиме.

При использовании синтаксиса **|s** отладчик отображает сведения о текущем процессе.

Эта команда также расчленяет текущую инструкцию для текущей системы, процесса и потока.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ~ (состояние потока)

Статья • 26.10.2023

Команда тильды (~) отображает состояние для указанного потока или для всех потоков в текущем процессе.

```
dbgcmd
```

```
~ Thread
```

## Параметры

*Поток*

Указывает отображаемый поток. Если этот параметр не указан, отображаются все потоки. Дополнительные сведения о синтаксисе см. в разделе Синтаксис потока.

### Среда

| Элемент | Описание |
|---|---|
| Режимы | Только в пользовательском режиме |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

### Дополнительные сведения

Дополнительные сведения и другие методы отображения или управления процессами и потоками см. в разделе Управление процессами и потоками.

## Комментарии

Потоки можно указать только в пользовательском режиме. В режиме ядра тильда (~) относится к процессору.

Символ потока можно добавить перед несколькими командами. Дополнительные сведения о значении тильды (~), за которой следует команда, см. в записи самой команды.

В следующих примерах показано, как использовать эту команду. Следующая команда отображает все потоки.

```
dbgcmd
```

```
0:001> ~
```

Следующая команда также отображает все потоки.

```
dbgcmd
```

```
0:001> ~*
```

Следующая команда отображает текущий активный поток.

```
dbgcmd
```

```
0:001> ~.
```

Следующая команда отображает поток, который изначально вызывал исключение (или который был активен при подключении отладчика к процессу).

```
dbgcmd
```

```
0:001> ~#
```

Следующая команда отображает поток с номером 2.

```
dbgcmd
```

```
0:001> ~2
```

Предыдущая команда отображает следующие выходные данные.

```
dbgcmd
```

```
0:001> ~
   0 id: 4dc.470 Suspend: 0 Teb 7ffde000 Unfrozen
 . 1 id: 4dc.534 Suspend: 0 Teb 7ffdd000 Unfrozen
#  2 id: 4dc.5a8 Suspend: 0 Teb 7ffdc000 Unfrozen
```

В первой строке выходных данных 0 — десятичное число потока, 4DC — это шестнадцатеричный идентификатор процесса, 470 — идентификатор шестнадцатеричного потока, 0x7FFDE000 — адрес TEB, а **Unfrozen** — состояние

потока. Период (.) перед потоком 1 означает, что этот поток является текущим потоком. Знак номера (#) перед потоком 2 означает, что этот поток изначально вызывал исключение или был активен при подключении отладчика к процессу.

# ~e (команда для конкретного потока)

Статья • 08.07.2024

Команда **~e** выполняет одну или несколько команд для определенного потока или для всех потоков в целевом процессе.

Не путайте эту команду с командой **e (ВВОД значений).**

dbgcmd

```
~Thread e CommandString
```

## Параметры

*Нить*
Указывает поток или потоки, для которым отладчик будет выполнять *CommandString* . Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока".

*CommandString*
Указывает одну или несколько команд для выполнения. Следует разделить несколько команд с запятой. *CommandString* включает остальную часть входной строки. Весь текст, следующий за буквой E, интерпретируется как часть этой строки. Не закрывайте *CommandString* в кавычки.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только в пользовательском режиме |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о других командах, управляющих потоками, см. в разделе "Управление процессами и потоками".

## Замечания

Потоки можно указать только в пользовательском режиме. В режиме ядра тильда (~) относится к процессору.

При использовании команды ~e вместе с одним потоком **команда ~e сохраняет только некоторые результаты** ввода. Например, следующие две команды эквивалентны.

dbgcmd

```
0:000> ~2e r; k; kd

0:000> ~2r; ~2k; ~2kd
```

Однако можно использовать **квалификатор ~e** для повторения команды или команды расширения несколько раз. При использовании квалификатора таким образом он может исключить дополнительную типизацию. Например, следующая команда повторяет **команду расширения !gle** для каждого потока, который выполняется отладка.

dbgcmd

```
0:000> ~*e !gle
```

Если в выполнении одной команды возникает ошибка, выполнение продолжается со следующей командой.

Вы не можете использовать **квалификатор ~e** вместе с командами выполнения (**g**, **gh**, **gn**, **gn**, **gN**, **gu**, **p**, **pa**, **pc**, **t**, **ta**, **tb**, tc, **wt**).

Вы не можете использовать **квалификатор ~e** вместе с **условными командами j (Execute If-Else)** или **z (Execute While)**.

При отладке нескольких процессов нельзя использовать **команду ~e** для доступа к пространству виртуальной памяти для неактивного процесса.

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ~f (замораживание потока)

Статья • 08.07.2024

Команда **~f** зависает заданный поток, что приводит к остановке и ожиданию, пока он не будет отключен.

Не путайте эту команду с командой **f (Fill Memory).**

dbgcmd

```
~Thread f
```

## Параметры

*Нить*
Указывает поток для замораживания. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только в пользовательском режиме |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о том, как работают замороженные потоки, и список других команд, которые управляют замораживанием и приостановкой потоков, см. в разделе "Управление процессами и потоками".

## Замечания

Потоки можно указать только в пользовательском режиме. В режиме ядра тильда (~) относится к процессору.

Команда **~f** приводит к заморозке указанного потока. Когда отладчик позволяет целевому приложению возобновить выполнение, другие потоки выполняются должным образом, пока этот поток остается остановленным.

В следующих примерах показано, как использовать эту команду. Следующая команда отображает текущее состояние всех потоков.

```
dbgcmd
```

```
0:000> ~* k
```

Следующая команда заморозка потока, вызвавшего текущее исключение.

```
dbgcmd
```

```
0:000> ~# f
```

Следующая команда проверяет, приостановлено ли состояние этого потока.

```
dbgcmd
```

```
0:000> ~* k
```

Следующая команда разблокирует номер потока 123.

```
dbgcmd
```

```
0:000> ~123 u
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ~u (unfreeze Thread)

Статья • 05.04.2024

Команда **~u** разблокирует указанный поток.

Не путайте эту команду с командой **U (Unassemble).**

```
dbgcmd
```

```
~Thread u
```

## Параметры

*Поток*
Указывает поток или потоки для разморозки. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только в пользовательском режиме |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о том, как работают замороженные потоки, и список других команд, которые управляют замораживанием и приостановкой потоков, см. в разделе "Управление процессами и потоками".

## Замечания

Потоки можно указать только в пользовательском режиме. В режиме ядра тильда (~) относится к процессору.

В следующих примерах показано, как использовать команды ~.

Следующая команда отображает текущее состояние всех потоков.

```
dbgcmd
```

```
0:000> ~* k
```

Следующая команда заморозка потока, вызвавшего текущее исключение.

```
dbgcmd
```

```
0:000> ~# f
```

Следующая команда проверка, что состояние этого потока приостановлено.

```
dbgcmd
```

```
0:000> ~* k
```

Следующая команда разблокирует номер потока 123.

```
dbgcmd
```

```
0:000> ~123 u
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ~n (приостановка потока)

Статья • 08.07.2024

Команда **~n** приостанавливает выполнение указанного потока.

Не путайте эту команду с командой **n (set Number Base).**

```
dbgcmd
```

```
~Thread n
```

## Параметры

*Нить*
Указывает поток или потоки для приостановки. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения о счетчике приостановки и о том, как работают приостановленные потоки, а также список других команд, которые управляют приостановкой и замораживанием потоков, см. в разделе "Управление процессами и потоками".

## Замечания

Потоки можно указать только в пользовательском режиме. В режиме ядра тильда (~) относится к процессору.

Каждый раз, когда вы используете **команду ~n** , количество приостановки потока увеличивается на один.

Адрес начала потока отображается при использовании этой команды.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ~m (возобновление потока)

Статья • 08.07.2024

Команда **~m** возобновляет выполнение указанного потока.

Не путайте эту команду с командой **m (Move Memory).**

```
dbgcmd
```

```
~Thread m
```

## Параметры

*Нить*
Указывает поток или потоки для возобновления. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения о счетчике приостановки и о том, как работают приостановленные потоки, а также список других команд, которые управляют приостановкой и замораживанием потоков, см. в разделе "Управление процессами и потоками".

## Замечания

Потоки можно указать только в пользовательском режиме. В режиме ядра тильда (~) относится к процессору.

Каждый раз, когда вы используете **команду ~m** , количество приостановки потока уменьшается на один.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ~s (установка текущего потока)

Статья • 05.04.2024

Команда **~s** задает или отображает текущий номер потока в пользовательском режиме.

Не путайте эту команду с командой **~s (Изменение текущего процессора),** которая работает только в режиме ядра, **команда |s (задать текущий процесс), ||s (задать текущую систему)** или **команду s (поиск памяти).**

dbgcmd

```
~Thread s
~ s
```

## Параметры

*Поток*
Задает поток для задания или отображения. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только в пользовательском режиме |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения и другие методы отображения или управления процессами и потоками см. в разделе "Управление процессами и потоками".

## Замечания

Потоки можно указать только в пользовательском режиме. В режиме ядра тильда (~) относится к процессору.

При использовании синтаксиса **~s** отладчик отображает сведения о текущем потоке.

Эта команда также расчленяет текущую инструкцию для текущей системы, процесса и потока.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ~s (изменение текущего процессора)

Статья • 05.04.2024

Команда **~s** задает, какой процессор отлаживается в многопроцессорной системе.

В режиме **ядра** **~s** изменяет текущий процессор. Не путайте эту команду с командой **~s (Set Current Thread),** которая работает только в пользовательском режиме, **команда |s (Задать текущий процесс),** **||s (задать текущую систему)** или **команду s (поиск памяти).**

dbgcmd

```
~Processor s
```

## Параметры

*Процессор*
Указывает количество обработчика для отладки.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Замечания

Вы можете указать процессоры только в режиме ядра. В пользовательском режиме тильда (~) относится к потоку.

Вы можете сразу же определить, когда вы работаете над системой нескольких процессоров по форме запроса отладки ядра. В следующем примере 0 означает, что выполняется отладка первого процессора на компьютере.

dbgcmd

```
0: kd>
```

Используйте следующую команду для переключения между процессорами:

dbgcmd

```
0: kd> ~1s
1: kd>
```

Теперь второй процессор на компьютере, который выполняется отладка.

## См. также

Синтаксис многопроцессора

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# a (Сборка)

Статья • 05.04.2024

**Команда** собирает 32-разрядную инструкцию x86 и помещает полученные коды инструкций в память.

dbgcmd

```
a [Address]
```

## Параметры

*Адрес*
Указывает начало блока в памяти, где помещаются полученные коды. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения об отладке сборки и связанных командах см. в разделе "Отладка в режиме сборки".

## Замечания

**Команда** не поддерживает 64-разрядную инструкцию mnemonics. Однако команда включена независимо от того, выполняется ли отладка 32-разрядного целевого объекта или 64-разрядного целевого объекта. Из-за сходства между инструкциями

x86 и x64 иногда можно использовать команду успешно при отладке 64-разрядного целевого объекта.

Если адрес не указан, сборка начинается с адреса, который указывает текущее значение указателя инструкции. Чтобы собрать новую инструкцию, введите нужный mnemonic и нажмите клавишу ВВОД. Чтобы завершить сборку, нажмите клавишу ВВОД.

Так как сборщик ищет все символы, которые называются в коде, эта команда может занять время. В течение этого времени вы не можете нажать клавиши **CTRL+C**, чтобы завершить команду.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ad (Delete Alias)

Статья • 08.07.2024

Команда **ad** удаляет псевдоним из списка псевдонимов.

```
dbgcmd
```

```
ad [/q] Name
ad *
```

## Параметры

**/q**
Задает тихий режим. Этот режим скрывает сообщение об ошибке, если псевдоним , *указывающий имя* , не существует.

*Имя*
Указывает имя псевдонима для удаления. Если указать звездочку (*), все псевдонимы удаляются (даже если псевдоним имеет значение "*").

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения об использовании псевдонимов см. в разделе "Использование псевдонимов".

## Замечания

С помощью команды ad **можно удалить любой** псевдоним с именем пользователя. Но эту команду нельзя использовать для удаления псевдонима фиксированного имени ($u 0 до $u 9).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ah (обработка утверждений)

Статья • 26.10.2023

Команда **ah** управляет состоянием обработки утверждений для определенных адресов.

dbgcmd

```
ahb [Address]
ahi [Address]
ahd [Address]
ahc
ah
```

# Параметры

**ahb**

Прерывает работу отладчика, если утверждение завершается ошибкой по указанному адресу.

**Ахи**
Игнорирует сбой утверждения по указанному адресу.

**Ahd**

Удаляет все сведения об обработке утверждений по указанному адресу. Это удаление приводит к тому, что отладчик возвращается в состояние по умолчанию для этого адреса.

*Адрес*

Указывает адрес инструкции, для которой задано состояние обработки утверждений. Если этот параметр не указан, отладчик использует текущий счетчик программы.

**Ahc**

Удаляет все сведения об обработке утверждений для текущего процесса.

**ah**

Отображает текущие параметры обработки утверждений.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | пользовательский режим, режим ядра |
| Цели | Только динамическая отладка |
| Платформы | все |

## Дополнительные сведения

Дополнительные сведения о состоянии прерывания и обработке состояния, описания всех кодов событий, список состояний по умолчанию для всех событий и сведения о других методах управления этим состоянием см. в разделе Управление исключениями и событиями.

# Комментарии

Команда **ah*** управляет состоянием обработки утверждений для определенного адреса. Команда **sx* asrt** управляет состоянием обработки глобальных утверждений. Если вы используете **ah*** для определенного адреса, а затем происходит утверждение, отладчик отвечает на основе параметров **ah*** и игнорирует параметры **sx* asrt** .

Когда отладчик обнаруживает утверждение, отладчик сначала проверяет, настроена ли обработка для этого конкретного адреса. Если обработка не настроена, отладчик использует глобальный параметр.

Команда **ah*** влияет только на текущий процесс. После завершения текущего процесса все параметры состояния теряются.

Состояние обработки утверждений влияет только на STATUS_ASSERTION_EXCEPTION исключения. Эта обработка не влияет на подпрограмму ASSERT в режиме ядра.

# al (список псевдонимов)

Статья • 11.11.2023

Команда **al** отображает список всех псевдонимов, определенных пользователем.

```
dbgcmd
```

```
al
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения об использовании псевдонимов см. в разделе Использование псевдонимов.

# Комментарии

Команда **al** выводит список всех именованных пользователем псевдонимов. Но эта команда не выводит список псевдонимов с фиксированным именем ($u 0 до $u 9).

# as, aS (Set Alias)

Статья • 08.07.2024

Команды **as** и **aS** определяют новый псевдоним или переопределить существующий.

dbgcmd

```
as Name EquivalentLine
aS Name EquivalentPhrase
aS Name "EquivalentPhrase"
as /e Name EnvironmentVariable
as /ma Name Address
as /mu Name Address
as /msa Name Address
as /msu Name Address
as /x Name Expression
aS /f Name File
as /c Name CommandString
```

# Параметры

*Имя*
Задает имя псевдонима. Это имя может быть любой текстовой строкой, которая не содержит пробела или нажатия клавиш ВВОД и не начинается с "al", "as", "aS" или "ad". *Имя* учитывает регистр.

*Эквивалентная строка*
Указывает эквивалент псевдонима. *Эквивалентная строка* учитывает регистр. Необходимо добавить по крайней мере одно пространство между *Name* и *EquivalentLine*. Количество пробелов между этими двумя параметрами не имеет значения. Эквивалент псевдонима никогда не содержит ведущих пробелов. После этих пробелов *Эквивалентная строка* включает остальную часть строки. Точки с запятой, кавычки и пробелы рассматриваются как литеральные символы, а конечные пробелы включаются.

*Эквивалентная фраза*
Указывает эквивалент псевдонима. *Эквивалентная фраза* учитывает регистр. Необходимо добавить по крайней мере одно пространство между *именем* и *эквивалентной фразой*. Количество пробелов между этими двумя параметрами не имеет значения. Эквивалент псевдонима никогда не содержит ведущих пробелов.

Эквивалентную фразу можно заключить в кавычки ("). Независимо от того, используется ли кавычки, *Эквивалентная фраза* может содержать пробелы, запятые и одинарные кавычки ('). Если вы заключаете *эквивалентную фразу* в кавычки, она может включать запятую, но не дополнительные кавычки. Если эквивалентная фраза *не заключена* в кавычки, она может включать кавычки в любое расположение, отличное от первого символа, но не может включать запятую. Конечные пробелы включаются независимо от того, используется ли кавычки.

**/e**
Задает псевдоним, эквивалентный переменной среды, указанной *EnvironmentVariable* .

*EnvironmentVariable*
Указывает переменную среды, используемую для определения эквивалента псевдонима. Используется среда отладчика, а не целевой объект. Если вы запустили отладчик в окне командной строки, используются переменные среды в этом окне.

**/мама**
Задает псевдоним, эквивалентный строке ASCII, завершаемой null, которая начинается с *адреса*.

**/мю**
Задает псевдоним, эквивалентный строке Юникода, завершаемой null, которая начинается с *адреса*.

**/msa**
Задает псевдоним, эквивалентный структуре ANSI_STRING, расположенной по *адресу Address*.

**/msu**
Задает псевдоним, эквивалентный структуре UNICODE_STRING, расположенной по *адресу Address*.

*Адрес*
Указывает расположение виртуальной памяти, используемой для определения эквивалента псевдонима.

**/x**
Задает псевдоним, эквивалентный 64-разрядному *значению выражения*.

*Выражение*
Указывает выражение для вычисления. Это значение становится эквивалентным

псевдонимом. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис числовых выражений".

**/f**
Задает псевдоним, эквивалентный содержимому *файла*. Всегда следует использовать переключатель **/f** вместе с **AS**, а не как.

*Файл*
Указывает файл, содержимое которого становится эквивалентным псевдонимом. *Файл* может содержать пробелы, но никогда не следует заключать *файл* в кавычки. Если указать недопустимый файл, появится сообщение об ошибке "Вне памяти".

**/c**
Задает псевдоним, эквивалентный выходным данным команд, которые *задает CommandString* . Эквивалент псевдонима включает возврат каретки, если они присутствуют в отображении команды и каретки возвращаются в конце отображения каждой команды (даже если указана только одна команда).

*CommandString*
Указывает команды, выходные данные которых становятся эквивалентными псевдонимами. Эта строка может включать любое количество команд, разделенных точкой с запятой.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения об использовании псевдонимов см. в разделе "Использование псевдонимов".

# Замечания

Если вы не используете никаких коммутаторов, команда использует остальную часть строки в качестве эквивалента псевдонима.

Команду AS **можно завершить** точкой с запятой. Этот метод полезен в скрипте, если необходимо поместить все команды в одну строку. Обратите внимание, что если часть строки после запятой требует расширения псевдонима, необходимо заключить вторую часть строки в новый блок. В следующем примере создаются ожидаемые выходные данные, 0x6.

dbgcmd

```
0:001> aS /x myAlias 5 + 1; .block{.echo myAlias}
0x6
```

Если опустить новый блок, вы не получите ожидаемые выходные данные. Это связано с тем, что расширение нового псевдонима не происходит до ввода нового блока кода. В следующем примере новый блок опущен, а выходные данные — текст myAlias вместо ожидаемого значения 0x6.

dbgcmd

```
0:001> aS /x myAlias 5 + 1; .echo myAlias
myAlias
```

Дополнительные сведения об использовании псевдонимов в сценариях см. в разделе "Использование псевдонимов".

Если вы используете **параметр /e**, /ma**, /mu**, /msa**, /msu** или **/x** switch, команды as **и aS работают так же,** и команда заканчивается, если обнаружена точка с запятой.

Если *имя* уже является именем существующего псевдонима, этот псевдоним переопределен.

Для создания или изменения псевдонима с именем пользователя можно использовать **команду as** или **aS** . Но вы не можете использовать команду для управления псевдонимом фиксированного имени ($u 0 до $u 9).

С помощью **переключателей /ma**, /mu**, /msa**, /msu**, /f** и **/c** можно создать псевдоним, содержащий возвращаемые каретки. Однако нельзя использовать псевдоним, содержащий каретки, для выполнения нескольких команд в последовательности. Вместо этого необходимо использовать запятую.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ba (перерыв на доступе)

Статья • 21.02.2024

Команда **ba** задает точку останова процессора (часто называемую менее точной *точкой* останова данных). Эта точка останова активируется при доступе к указанной памяти.

Режим пользователя

```
dbgcmd
```

```
[~Thread] ba[ID] Access Size [Options] [Address [Passes]] ["CommandString"]
```

Режим ядра

```
dbgcmd
```

```
ba[ID] Access Size [Options] [Address [Passes]] ["CommandString"]
```

## Параметры

*Поток*
Указывает поток, к которому применяется точка останова. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

*ИД*
Указывает необязательное число, определяющее точку останова. Если идентификатор не указан, используется первый доступный номер точки останова. Нельзя добавить пробел между **ba** и номером идентификатора. Каждый процессор поддерживает только ограниченное количество точек останова процессора, но нет ограничений на значение номера идентификатора. Если идентификатор *заключен* в квадратные скобки ([]), *идентификатор* может включать любое выражение. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис числовых выражений".

*Открыть*
Указывает тип доступа, удовлетворяющий точке останова. Этот параметр может принимать одно из указанных ниже значений.

Развернуть таблицу

| Вариант | Действие |
|---|---|
| **e** (execute) | Прерывается в отладчик, когда ЦП получает инструкцию из указанного адреса. |
| **r** (чтение и запись) | Прерывается в отладчик, когда ЦП считывает или записывает данные по указанному адресу. |
| **w** (запись) | Прерывается в отладчик при записи ЦП по указанному адресу. |
| **i (i** /o) | (Только в режиме ядра, только системы на основе x86) Прерывается в отладчик при доступе к порту ввода-вывода по указанному *адресу* . |

*Размер*
Задает размер расположения в байтах для отслеживания доступа. На процессоре на основе x86 этот параметр может быть 1, 2 или 4. Однако если *Access* равен **e**, *размер* должен иметь значение 1.

В процессоре на основе x64 этот параметр может быть 1, 2, 4 или 8. Однако если *Access* равен **e**, *размер* должен иметь значение 1.

*Параметры* указывают параметры точки останова. Можно использовать любое число следующих параметров, кроме указанных ниже.

**/1**
Создает точку останова с одним снимком. После активации этой точки останова точка останова окончательно удаляется из списка точек останова.

**/p***EProcess*
(только режим ядра) Указывает процесс, связанный с этой точкой останова. *EProcess* должен быть фактическим адресом структуры EPROCESS, а не piD. Точка останова активируется только в том случае, если она обнаружена в контексте этого процесса.

**/t***EThread*
(только режим ядра) Указывает поток, связанный с этой точкой останова. *EThread* должен быть фактическим адресом структуры ETHREAD, а не идентификатором потока. Точка останова активируется только в том случае, если она обнаружена в контексте этого потока. Если вы используете **/p***EProcess* и **/t***EThread* , их можно ввести в любом порядке.

***/c**MaxCallStackDepth*
Вызывает активную точку останова только в том случае, если глубина стека вызовов меньше *MaxCallStackDepth*. Этот параметр нельзя объединить вместе с **/C**.

***/C**MinCallStackDepth*
Вызывает активную точку останова только в том случае, если глубина стека вызовов превышает *MinCallStackDepth*. Этот параметр нельзя объединить вместе с **/c**.

**Выражение объекта** /w dx задает условную точку останова на основе логического значения, возвращаемого выражением объекта dx. Аргумент — это выражение модели данных (dx), которое оценивается как true (соответствует условию — разрыв) или false (не соответствует условию — не прерывать).

В этом примере устанавливается условная точка останова на основе значения globalVariable. Это позволяет точке останова доступа, например, проверка значение, записанное при определении необходимости разрыва отладчика.

dbgcmd

```
ba w 4 /w "mymodule!globalVariable == 4" mymodule!globalVariable
```

В этом примере показано, как задать точку останова с помощью JavaScript.

dbgcmd

```
ba w 4 /w "@$scriptContents.myFunc(mymodule!globalVariable)"
mymodule!globalVariable
```

Дополнительные сведения о объектах отладчика см. в dx (выражение объектной модели отладчика отладчика).

Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

*Адрес*
Указывает любой допустимый адрес. Если приложение обращается к памяти по этому адресу, отладчик останавливает выполнение и отображает текущие значения всех регистров и флагов. Этот адрес должен быть смещением и соответствующим образом выровнен для соответствия параметру *Size* . (Например, если *Размер* равен 4, *адрес* должен быть кратным 4.) Если не указан *адрес*, используется указатель текущей инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Проходит*
Указывает количество раз, когда точка останова передается до его активации. Это число может быть любым 16-разрядным значением. Количество раз, когда счетчик программы проходит через эту точку без разрыва, меньше, чем значение этого числа. Таким образом, опущение этого числа совпадает с параметром 1. Обратите внимание, что это число подсчитывает только время выполнения *приложения* за эту точку. Пошаговые действия или трассировка за эту точку не учитываются. После достижения полного количества можно сбросить это число только с помощью очистки и сброса точки останова.

*CommandString*
Указывает список команд, выполняемых каждый раз, когда точка останова встречается с указанным числом раз. Эти команды выполняются только в том случае, если точка останова достигнута после выполнения **команды g (Go),** а не после **команды t (Трассировка)** или **p (шаг).** Команды отладчика в *CommandString* могут включать параметры.

Необходимо заключить эту строку команды в кавычки и разделить несколько команд точкой с запятой. Можно использовать стандартные символы элемента управления C (например **, \n** и **\"**). Точки с запятой, содержащиеся в кавычках второго уровня (**\"**), интерпретируются как часть внедренной строки кавычки.

Это необязательный параметр.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения о точках останова процессора см. в статьях "Точки останова процессора" (точки останова ba). Дополнительные сведения об использовании точек останова и примеры использования других команд и методов останова и методов управления точками останова и сведения о настройке точек останова в пространстве пользователя из отладчика ядра см. в разделе

"Использование точек останова". Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

## Замечания

Отладчик использует *идентификатор,* чтобы ссылаться на точку останова в последующих **bc (точка останова clear), bd (отключить точку останова)** и **быть (включить точку останова).**

**Используйте команду BL (список точек останова)** для перечисления всех существующих точек останова, их идентификаторов и их состояния.

**Используйте команду BPCMDS (команды отображения точек останова)** для перечисления всех существующих точек останова, их идентификаторов и команд, которые использовались для их создания.

Каждая точка останова процессора имеет размер, связанный с ним. Например, **точку останова процессора w** (write) можно задать в адресе 0x70001008 размером 4 байта. Это позволит отслеживать блок адресов от 0x70001008 до 0x7000100B включительно. Если этот блок памяти записывается, точка останова будет активирована.

Это может произойти, что процессор выполняет операцию в области памяти, *которая перекрывается* , но не идентична указанному региону. В этом примере одна операция записи, содержащая диапазон 0x70001000 для 0x7000100F, или операция записи, которая включает только байт в 0x70001009, будет перекрывающейся операцией. В такой ситуации, активируется ли точка останова процессором. Чтобы получить подробные сведения, обратитесь к руководству по обработчику. Чтобы взять один конкретный экземпляр, на процессоре x86 точка чтения или записи активируется всякий раз, когда доступ к диапазону перекрывает диапазон точек останова.

Аналогичным образом, если точка останова e (execute) задана в адресе 0x00401003, а затем выполняется двухбайтовая инструкция, охватывая адреса 0x00401002 и 0x00401003, результат зависит от процессора. Опять же, обратитесь к руководству по архитектуре процессора, чтобы получить подробные сведения.

Процессор различает точки останова, заданные отладчиком пользовательского режима и точками останова, заданными отладчиком в режиме ядра. Точка останова процессора в пользовательском режиме не влияет на процессы в режиме ядра. Точка останова процессора в режиме ядра может повлиять на

процесс пользовательского режима в зависимости от того, используется ли код отладочного регистра и подключен ли отладчик пользовательского режима.

Чтобы применить существующие точки останова данных текущего процесса к другому контексту регистра, используйте **команду .apply_dbp (Применить точку останова данных к контексту).**

На компьютере с несколькими процессорами каждая точка останова процессора применяется ко всем процессорам. Например, если текущий процессор равен 3, и вы используете команду `ba e1 MyAddress` , чтобы поместить точку останова в MyAddress, любой процессор - не только процессор 3- который выполняется в этом адресе, активирует точку останова. (Это также относится к точкам останова программного обеспечения.)

Невозможно создать несколько точек останова процессора в одном адресе, которые отличаются только в значениях *CommandString* . Однако можно создать несколько точек останова в одном адресе с различными ограничениями (например, разные значения **параметров /p**, /t**, /c** и **/C**).

Дополнительные сведения о точках останова процессора и дополнительных ограничениях, которые применяются к ним, см. в статьях "Точки останова процессора" (точки останова ba).

В следующих примерах показана **команда ba** . Следующая команда задает точку останова для доступа на чтение 4 байта переменной myVar.

dbgcmd

```
0:000> ba r4 myVar
```

Следующая команда добавляет точку останова ко всем последовательным портам с адресами из 0x3F8 до 0x3FB. Эта точка останова активируется, если что-либо считывается или записывается в эти порты.

dbgcmd

```
kd> ba i4 3f8
```

# bc (точка останова clear)

Статья • 08.07.2024

Команда **bc** окончательно удаляет ранее заданные точки останова из системы.

```
dbgcmd
```

```
bc Breakpoints
```

## Параметры

*Точки останова*
Указывает идентификаторы точек останова для удаления. Можно указать любое количество точек останова. Необходимо разделить несколько идентификаторов по пробелам или запятыми. Можно указать диапазон идентификаторов точек останова с помощью дефиса (-). С помощью звездочки (*) можно указать все точки останова. Если вы хотите использовать числовое выражение для идентификатора, заключите его в квадратные скобки ([]). Если вы хотите использовать строку с подстановочными знаками для сопоставления символьного имени точки останова, заключите ее в кавычки (" ).

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения об использовании точек останова, других команд и методов управления точками останова и настройке точек останова в пользовательском пространстве из отладчика ядра см. в разделе "Использование точек останова". Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

# Замечания

**Используйте команду BL (список точек останова)** для перечисления всех существующих точек останова, их идентификаторов и их состояния.

**Используйте команду BPCMDS (команды отображения точек останова)** для перечисления всех существующих точек останова, их идентификаторов и команд, которые использовались для их создания.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# bd (отключение точки останова)

Статья • 08.07.2024

Команда **bd** отключает, но не удаляет ранее заданные точки останова.

```
dbgcmd
```

```
bd Breakpoints
```

## Параметры

*Точки останова*
Указывает идентификаторы точек останова для отключения. Можно указать любое количество точек останова. Необходимо разделить несколько идентификаторов по пробелам или запятыми. Можно указать диапазон идентификаторов точек останова с помощью дефиса (-). С помощью звездочки (*) можно указать все точки останова. Если вы хотите использовать числовое выражение для идентификатора, заключите его в квадратные скобки ([]). Если вы хотите использовать строку с подстановочными знаками для сопоставления символьного имени точки останова, заключите ее в кавычки (" ).

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения об использовании точек останова, других команд и методов управления точками останова и настройке точек останова в пользовательском пространстве из отладчика ядра см. в разделе "Использование точек останова". Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

# Замечания

Если точка останова отключена, система не проверяет допустимость условий, указанных в точке останова.

**Используйте команду be (Включить точку останова),** чтобы повторно включить отключенную точку останова.

**Используйте команду BL (список точек останова)** для перечисления всех существующих точек останова, их идентификаторов и их состояния.

**Используйте команду BPCMDS (команды отображения точек останова)** для перечисления всех существующих точек останова, их идентификаторов и команд, которые использовались для их создания.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# be (включить точку останова)

Статья • 08.07.2024

Команда **be** восстанавливает одну или несколько точек останова, которые ранее были отключены.

```
dbgcmd
```

```
be Breakpoints
```

## Параметры

*Точки останова*
Указывает идентификаторы точек останова для включения. Можно указать любое количество точек останова. Необходимо разделить несколько идентификаторов по пробелам или запятыми. Можно указать диапазон идентификаторов точек останова с помощью дефиса (-). С помощью звездочки (*) можно указать все точки останова. Если вы хотите использовать числовое выражение для идентификатора, заключите его в квадратные скобки ([ ]). Если вы хотите использовать строку с подстановочными знаками для сопоставления символьного имени точки останова, заключите ее в кавычки ("").

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения и примеры использования точек останова, других команд точки останова и методов управления точками останова и настройке точек останова в пользовательском пространстве из отладчика ядра см. в разделе

"Использование точек останова". Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

## Замечания

**Используйте команду BL (список точек останова)** для перечисления всех существующих точек останова, их идентификаторов и их состояния.

**Используйте команду BPCMDS (команды отображения точек останова)** для перечисления всех существующих точек останова, их идентификаторов и команд, которые использовались для их создания.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# BL (список точек останова)

Статья • 08.07.2024

Команда **BL** выводит сведения о существующих точках останова.

```
dbgcmd
```

```
bl [/L] [Breakpoints]
```

## Параметры

**/L**
Принудительно **выводит большие двоичные объекты** , чтобы всегда отображать адреса точек останова вместо отображения исходных файлов и номеров строк.

*Точки останова*
Указывает идентификаторы точек останова для списка. Если опущены *точки останова*, отладчик выводит список всех точек останова. Можно указать любое количество точек останова. Необходимо разделить несколько идентификаторов по пробелам или запятыми. Можно указать диапазон идентификаторов точек останова с помощью дефиса (-). С помощью звездочки (*) можно указать все точки останова. Если вы хотите использовать числовое выражение для идентификатора, заключите его в квадратные скобки ([]). Если вы хотите использовать строку с подстановочными знаками для сопоставления символьного имени точки останова, заключите ее в кавычки ("").

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения и примеры использования точек останова, других команд точки останова и методов управления точками останова и настройке точек останова в пользовательском пространстве из отладчика ядра см. в разделе "Использование точек останова". Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

# Замечания

Для каждой точки останова команда отображает следующие сведения:

- Идентификатор точки останова. Этот идентификатор представляет собой десятичное число, которое можно использовать для ссылки на точку останова в последующих командах.
- Состояние точки останова. Состояние может быть **e** (включено) или **d** (отключено).
- (Только неразрешенные точки останова) Буква u отображается, если точка останова неразрешенная. То есть точка останова не соответствует символьной ссылке в любом загруженном модуле. Сведения об этих точках останова см. в разделе "Неразрешенные точки останова" (бу точки останова).
- Виртуальный адрес или символическое выражение, составляющие расположение точки останова. Если вы включили загрузку номера исходной строки, **команда BL** отображает сведения о файле и номере строки вместо смещения адресов. Если точка останова не разрешена, адрес опущен здесь и отображается в конце списка.
- (Только точки останова данных) Сведения о типе и размере отображаются для точек останова данных. Типы могут быть **e** (execute), **r** (чтение и запись), **w** (запись) или **i** (входные и выходные данные). За этими типами следует размер блока в байтах. Сведения об этих точках останова см. в разделе "Точки останова обработчика" (точки останова ba).
- Количество проходов, которые остаются до активации точки останова, за которым следует начальное число проходов в скобках. Дополнительные сведения об этой точке останова см. в описании *параметра Passes* в **bp, bu, bm (Установка точки останова).**
- Связанный процесс и поток. Если поток получается как три звездочки (***), эта точка останова не является точкой останова для конкретного потока.

- Модуль и функция со смещением, соответствующим адресу точки останова. Если точка останова не разрешена, адрес точки останова отображается здесь, а в скобках. Если точка останова задана на допустимом адресе, но отсутствуют сведения о символах, это поле пусто.
- Команда, которая автоматически выполняется при нажатии этой точки останова. Эта команда отображается в кавычках.

Если вы не уверены, какая команда использовалась для задания существующей точки останова, используйте **bpcmds (отображение команд точки останова)** для перечисления всех точек останова вместе с командами, которые использовались для их создания.

В следующем примере показаны выходные данные **команды BL** .

Пример

dbgcmd

```
0:000> bl
 0 e 010049e0     0001 (0001)  0:**** stst!main
```

Эти выходные данные содержат следующие сведения:

- Идентификатор точки останова равен **0**.
- Состояние точки останова — **e** (включено).
- Точка останова не является неразрешенной (в выходных данных нет **u** ).
- Виртуальный адрес точки останова — **010049e0**.
- Точка останова активна в первом проходе по коду, и код еще не был выполнен в отладчике. Эти сведения указываются значением 1 (**0001**) в счетчике "проходы оставшихся" и значением 1 (**(0001)**) в начальном счетчике передачи.
- Эта точка останова не является точкой останова для конкретного потока (***).
- Точка останова устанавливается в **главном** модуле **stst** .

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# bp, bu, bm (установка точки останова)

Статья • 26.10.2023

Команды **bp**, **bu** и **bm** устанавливают одну или несколько точек останова программного обеспечения. Вы можете комбинировать расположения, условия и параметры, чтобы задать различные типы точек останова программного обеспечения.

User-Mode

dbgcmd

```
[~Thread] bp[ID] [Options] [Address [Passes]] ["CommandString"]
[~Thread] bu[ID] [Options] [Address [Passes]] ["CommandString"]
[~Thread] bm [Options] SymbolPattern [Passes] ["CommandString"]
```

Kernel-Mode

dbgcmd

```
bp[ID] [Options] [Address [Passes]] ["CommandString"]
bu[ID] [Options] [Address [Passes]] ["CommandString"]
bm [Options] SymbolPattern [Passes] ["CommandString"]
```

# Параметры

*Поток*
Указывает поток, к которому применяется точка останова. Дополнительные сведения о синтаксисе см. в разделе Синтаксис потока. Потоки можно указать только в пользовательском режиме. Если поток не указан, точка останова применяется ко всем потокам.

*ID*
Задает десятичное число, идентифицирующее точку останова.

Отладчик назначает *идентификатор* при создании точки останова, но его можно изменить с помощью команды **br (изменение нумерации точки останова).** *Идентификатор* можно использовать для ссылки на точку останова в последующих командах отладчика. Чтобы отобразить *идентификатор* точки останова, используйте команду **bl (список точек останова).**

При использовании *идентификатора* в команде не вводите пробел между командой (**bp** или **bu**) и идентификатором.

Параметр *ID* всегда является необязательным. Если *идентификатор* не указан, отладчик использует первый доступный номер точки останова. В режиме ядра можно задать только 32 точки останова. В пользовательском режиме можно задать любое количество точек останова. В любом случае нет ограничений на значение *идентификатора* . Если *заключить идентификатор* в квадратные скобки (**[]**), *идентификатор* может включать любое выражение. Дополнительные сведения о синтаксисе см. в разделе Синтаксис числовых выражений.

*Параметры* Задает параметры точки останова. Можно указать любое количество следующих параметров, за исключением указанных ниже.

**/1**
Создает "однократную" точку останова. После активации этой точки останова она удаляется из списка точек останова.

**/p***EProcess*
(только в режиме ядра) Указывает процесс, связанный с этой точкой останова. *EProcess* должен быть фактическим адресом структуры EPROCESS, а не PID. Точка останова активируется только в том случае, если она обнаружена в контексте этого процесса.

**/t***EThread*
(только в режиме ядра) Указывает поток, связанный с этой точкой останова. *EThread* должен быть фактическим адресом структуры ETHREAD, а не идентификатором потока. Точка останова активируется только в том случае, если она обнаружена в контексте этого потока. При использовании **/p***EProcess* и **/t***EThread* их можно ввести в любом порядке.

**/c***MaxCallStackDepth*
Активирует точку останова, только если глубина стека вызовов меньше *MaxCallStackDepth*. Этот параметр нельзя использовать вместе с **параметром /C**.

**/C***MinCallStackDepth*
Активирует точку останова, только если глубина стека вызовов больше , *чем MinCallStackDepth*. Этот параметр нельзя использовать вместе с **параметром /c**.

**/a**
(Только для **bm** ) Задает точки останова для всех указанных расположений, независимо от того, находятся ли они в пространстве данных или в пространстве кода. Так как точки останова в данных могут привести к сбоям программы,

используйте этот параметр только в расположениях, которые известны как безопасные.

**/D**
(Только для **bm** ) Преобразует расположения точек останова в адреса. Таким образом, при перемещении кода точки останова остаются на том же адресе, а не в соответствии с *SymbolPattern*. Используйте **параметр /d** , чтобы избежать повторной оценки изменений в точках останова при загрузке или выгрузке модулей.

**/(**
(Только для **bm** ) Включает сведения о списке параметров в строку символов, *определяемую SymbolString* .

Эта функция позволяет задавать точки останова для перегруженных функций с одинаковыми именами, но разными списками параметров. Например, bm /( myFunc задает точки останова для **myFunc(int a)** и **myFunc(char a)**. Без "/(" точка останова, заданная для **myFunc** , завершается сбоем, так как она не указывает, для какой функции **myFunc** предназначена точка останова.

**Выражение объекта /w dx** Задает условную точку останова на основе логического значения, возвращаемого выражением объекта dx. Аргумент — это выражение модели данных (dx), которое принимает значение true (соответствует условию — разрыв) или false (не соответствует условию — не прерывается).

В этом примере задается условная точка останова на основе значения localVariable.

```
dbgcmd
```

```
bp /w "localVariable == 4" mymodule!myfunction
```

В этом примере показано, как задать точку останова с помощью JavaScript.

```
dbgcmd
```

```
bp /w "@$scriptContents.myFunc(localVariable)" @rip
```

Дополнительные сведения об объектах отладчика см. в разделе dx (отображение выражения объектной модели отладчика).

Дополнительные сведения об условных точках останова см. в разделе Установка условной точки останова.

*Адрес*
Указывает первый байт инструкции, в которой установлена точка останова. Если не указать *адрес*, используется текущий указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе Синтаксис адреса и диапазона адресов.

*Проходит*
Указывает число проходов выполнения, на которых активирована точка останова. Отладчик пропускает расположение точки останова, пока не достигнет указанного прохода. Значение *Passes* может быть любым 16-разрядным или 32-разрядным значением.

По умолчанию точка останова активна при первом выполнении приложением кода, содержащего расположение точки останова. Эта ситуация по умолчанию эквивалентна значению **1** для *проходов*. Чтобы активировать точку останова только после того, как приложение выполнит код хотя бы один раз, введите значение **2** или более. Например, значение **2** активирует точку останова при втором выполнении кода.

Этот параметр создает счетчик, который уменьшается при каждом проходе кода. Чтобы просмотреть начальные и текущие значения счетчика *Passes*, используйте **bl (список точек останова).**

Счетчик *Passes* уменьшается, только если приложение *выполняется* после точки останова в ответ на команду **g (Go).** Счетчик не уменьшается, если выполняется пошаговое выполнение кода или его трассировка. Когда значение счетчика *Passes* достигает **1**, его можно сбросить только путем очистки и сброса точки останова.

*CommandString*
Указывает список команд, которые выполняются при каждом обнаружении точки останова указанное количество раз. Параметр *CommandString* необходимо заключить в кавычки. Используйте точку с запятой для разделения нескольких команд.

Команды отладчика в *CommandString* могут включать параметры. Можно использовать стандартные управляющие символы C (например **,\n** и **\"**). Точки с запятой, содержащиеся в кавычках второго уровня (**\"),** интерпретируются как часть внедренной строки в кавычках.

Команды *CommandString* выполняются только в том случае, если точка останова достигнута во время *выполнения* приложения в ответ на команду **g (Go).** Команды не выполняются, если вы выполняете пошаговое выполнение кода или трассировку после этого момента.

Любая команда, возобновляющая выполнение программы после точки останова (например **, g** или **t**), завершает выполнение списка команд.

*SymbolPattern*
Задает шаблон. Отладчик пытается сопоставить этот шаблон с существующими символами и установить точки останова для всех совпадений шаблонов. *SymbolPattern* может содержать различные подстановочные знаки и описатели. Дополнительные сведения об этом синтаксисе см. в разделе Синтаксис подстановочных знаков строк. Так как эти символы сопоставляются с символами, в сопоставлении регистр не учитывается, а один символ подчеркивания в начале (_) представляет любое количество символов подчеркивания в начале.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | пользовательский режим, режим ядра |
| Цели | Только динамическая отладка |
| Платформы | все |

## Дополнительные сведения

Дополнительные сведения об использовании точек останова, других командах и методах управления точками останова, а также о том, как задать точки останова в пространстве пользователя из отладчика ядра, см. в статье Использование точек останова. Дополнительные сведения об условных точках останова см. в разделе Настройка условной точки останова.

# Комментарии

Команды **bp**, **bu** и **bm** задают новые точки останова, но имеют разные характеристики:

- Команда **bp (Set Breakpoint)** задает новую точку останова по *адресу* расположения точки останова, указанному в команде. Если отладчик не может разрешить адресное выражение расположения точки останова при установке точки останова, точка останова **bp** автоматически преобразуется в точку **останова** bu. Используйте команду **bp** , чтобы создать точку останова, которая больше не активна при выгрузке модуля.

- Команда **bu (Задать неразрешенную точку останова)** задает *отложенную* или *неразрешенную точку останова* . Точка **останова** задается на символьной ссылке на расположение точки останова, указанное в команде (не по адресу), и активируется при разрешении модуля со ссылкой. Дополнительные сведения об этих точках останова см. в разделе Неразрешенные точки останова (bu Точки останова).

- Команда **bm (Set Symbol Breakpoint)** задает новую точку останова для символов, соответствующих заданному шаблону. Эта команда может создать несколько точек останова. По умолчанию после сопоставления шаблона точки останова **bm** совпадают с точками останова **bu** . То есть точки **останова bm** — это отложенные точки останова, заданные для символьной ссылки. Однако команда bm /d создает одну или несколько точек **останова bp** . Каждая точка останова задается по адресу соответствующего расположения и не отслеживает состояние модуля.

Если вы не уверены, какая команда использовалась для задания существующей точки останова, используйте **.bpcmds (Отображение команд точек останова),** чтобы получить список всех точек останова вместе с командами, которые использовались для их создания.

Существует три основных различия между точками **останова bp** и **точками останова bu** :

- Расположение точки останова **bp** всегда преобразуется в адрес. Если изменение модуля перемещает код, в котором была установлена точка останова **bp** , точка останова остается на том же адресе. С другой стороны, точка **останова bu** остается связанной с символическим значением (обычно символ плюс смещение), которое использовалось, и отслеживает это символьное расположение, даже если его адрес меняется.

- При обнаружении адреса точки **останова** в загруженном модуле и при последующей выгрузке этого модуля точка останова удаляется из списка точек останова. С другой стороны, точки **останова сохраняются** после повторных выгрузок и нагрузок.

- Точки останова, заданные с помощью **bp** , не сохраняются в рабочих областях WinDbg. Точки останова, заданные с **помощью bu,** сохраняются в рабочих областях.

Команда **bm** полезна, если вы хотите использовать подстановочные знаки в шаблоне символов для точки останова. Синтаксис **bm***SymbolPattern* эквивалентен использованию **x SymbolPattern** , а затем использованию **bu** для каждого

результата. Например, чтобы задать точки останова для всех символов в модуле *Myprogram* , которые начинаются со строки "mem", используйте следующую команду.

Пример

```
dbgcmd
```

```
0:000> bm myprogram!mem*
  4: 0040d070 MyProgram!memcpy
 5: 0040c560 MyProgram!memmove
  6: 00408960 MyProgram!memset
```

Так как команда **bm** задает точки останова программного обеспечения (не точки останова процессора), она автоматически исключает расположение данных при установке точек останова, чтобы избежать повреждения данных.

При использовании команд **bp** или bm /a можно указать адрес данных, а не адрес программы. Однако даже если указано расположение данных, эти команды создают точки останова программного обеспечения, а не точки останова процессора. Если точка останова программного обеспечения помещается в программные данные, а не в исполняемый код, это может привести к повреждению данных. Поэтому эти команды следует использовать в расположении данных, только если вы уверены, что память, хранящуюся в этом расположении, будет использоваться как исполняемый код, а не как программные данные. В противном случае следует использовать команду **ba (Break on Access).** Дополнительные сведения см. в разделе Точки останова процессора (ba Точки останова).

Дополнительные сведения о том, как задать точку останова в расположении, заданном более сложным синтаксисом, например члене открытого класса C++ или произвольной текстовой строке, содержащей символы с другим ограничением, см. в разделе Синтаксис точки останова.

Если одна логическая исходная строка охватывает несколько физических линий, точка останова устанавливается в последней физической строке оператора или вызова. Если отладчик не может установить точку останова в запрошенной позиции, он помещает точку останова в следующую допустимую позицию.

Если указать *потоk*, то точки останова устанавливаются в указанных потоках. Например, команда **~*bp** устанавливает точки останова для всех потоков, **~#bp** задает точку останова в потоке, который вызывает текущее исключение, и **~123bp** — точку останова в потоке 123. Команды **~bp** и **~.bp** задают точку останова в текущем потоке.

При отладке многопроцессорной системы в режиме ядра точки останова, заданные с помощью **bp** или **ba (break on Access),** применяются ко всем процессорам. Например, если текущий процессор равен 3 и вы вводите **bp MemoryAddress** , чтобы установить точку **останова в MemoryAddress**. Любой процессор, выполняющийся по указанному адресу (а не только процессор 3), вызывает ловушку точки останова.

Команды **bp**, **bu** и **bm** задают точки останова программного обеспечения, заменяя инструкцию процессора инструкцией прерывания. Для отладки кода, доступного только для чтения, или кода, который невозможно изменить, используйте команду ba e, где **e** представляет доступ только для выполнения.

Следующая команда устанавливает точку останова в 12 байтах после начала функции **MyTest**. Эта точка останова игнорируется для первых шести проходов кода, но выполнение останавливается на седьмом проходе кода.

dbgcmd

```
0:000> bp MyTest+0xb 7
```

Следующая команда устанавливает точку останова в **RtlRaiseException**, отображает регистр **eax** , отображает значение символа **MyVar** и продолжает.

dbgcmd

```
kd> bp ntdll!RtlRaiseException "r eax; dt MyVar; g"
```

Следующие две команды **bm** задают три точки останова. При выполнении команд отображаемый результат не различает точки останова, созданные с помощью параметра /d, и точки **останова** , созданные без него. Для различения этих двух типов можно использовать **bpcmds (команды для отображения точек останова** ). Если точка останова была создана **с помощью bm** без параметра **/d** , то отображение **.bpcmds** указывает тип точки останова как **bu**, за которым следует вычисляемый символ, заключенный в маркер **@!""** (который указывает, что это литеральный символ, а не числовое выражение или регистр). Если точка останова была создана **bm** с параметром **/d** , то отображение **.bpcmds** указывает тип точки останова как **bp**.

dbgcmd

```
0:000> bm myprog!openf*
  0: 00421200 @!"myprog!openFile"
  1: 00427800 @!"myprog!openFilter"
```

```
0:000> bm /d myprog!closef*
  2: 00421600 @!"myprog!closeFile"

0:000> .bpcmds
bu0 @!"myprog!openFile";
bu1 @!"myprog!openFilter";
bp2 0x00421600 ;
```

# br (переназначка точки останова)

Статья • 08.07.2024

Команда **br** переназначает одну или несколько точек останова.

```
dbgcmd
```

```
br OldID NewID [OldID2 NewID2 ...]
```

## Параметры

*OldID*
Указывает текущий идентификатор точки останова.

*NewID*
Задает новое число, которое становится идентификатором точки останова.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения и примеры использования точек останова, других команд точки останова и методов управления точками останова и настройке точек останова в пользовательском пространстве из отладчика ядра см. в разделе "Использование точек останова". Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

## Замечания

Команду br **можно использовать** для переуменовки любого количества точек останова одновременно. Для каждой точки останова укажите старый идентификатор и новый идентификатор в этом порядке в качестве параметров **для br**.

Если у вас уже есть точка останова с идентификатором, равным *NewID*, команда завершается ошибкой и отображается сообщение об ошибке.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# bs (обновление команды точки останова)

Статья • 08.07.2024

Команда **bs** изменяет команду, выполняемую при обнаружении указанной точки останова.

```
dbgcmd
```

```
bs ID ["CommandString"]
```

## Параметры

*Идентификатор*
Указывает идентификатор точки останова.

*CommandString*
Указывает новый список команд, выполняемых при каждом обнаружении точки останова. Необходимо заключить *параметр CommandString* в кавычки. Используйте запятую для разделения нескольких команд.

Команды отладчика в *CommandString* могут включать параметры. Можно использовать стандартные символы C-control (например **, \n** и **\"**). Точки с запятой, содержащиеся в кавычках второго уровня (**\"**), интерпретируются как часть внедренной строки кавычки.

*Команды CommandString* выполняются только в том случае, если точка останова достигнута, пока приложение выполняется в ответ на **команду g (Go).** Команды не выполняются, если выполняется шаг по коду или трассировке в прошлой точке.

Любая команда, которая возобновляет выполнение программы после точки останова (например **, g** или **t**), завершает выполнение списка команд.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |

| Позиция | Description |
| --- | --- |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения и примеры использования точек останова, других команд точки останова и методов управления точками останова и настройке точек останова в пользовательском пространстве из отладчика ядра см. в разделе "Использование точек останова". Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

## Замечания

*Если CommandString* не указан, все команды, уже заданные в точке останова, удаляются.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# bsc (обновление условной точки останова)

Статья • 08.07.2024

Команда **bsc** изменяет условие, при котором возникает точка останова или изменяет команду, выполняемую при обнаружении указанной условной точки останова.

```
dbgcmd
```

```
bsc ID Condition ["CommandString"]
```

## Параметры

*Идентификатор*
Указывает идентификатор точки останова.

*Состояние*
Указывает условие, при котором должна активироваться точка останова.

*CommandString*
Указывает новый список команд, выполняемых при каждом обнаружении точки останова. Необходимо заключить *параметр CommandString* в кавычки. Используйте запятую для разделения нескольких команд.

Команды отладчика в *CommandString* могут включать параметры. Можно использовать стандартные символы C-control (например **, \n** и **\"**). Точки с запятой, содержащиеся в кавычках второго уровня (**\"**), интерпретируются как часть внедренной строки кавычки.

Команды CommandString выполняются только в том случае, если точка останова достигнута, пока приложение выполняется в ответ на **команду g (Go).** Команды не выполняются, если выполняется шаг по коду или трассировке в прошлой точке.

Любая команда, которая возобновляет выполнение программы после точки останова (например **, g** или **t**), завершает выполнение списка команд.

## Среда

Развернуть таблицу

| | |
|---|---|
| Оду M | Режим пользователя, режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Все |

# Дополнительная информация

Дополнительные сведения и примеры использования точек останова, других команд точки останова и методов управления точками останова и настройке точек останова в пользовательском пространстве из отладчика ядра см. в разделе "Использование точек останова". Дополнительные сведения об условных точках останова см. в разделе "Настройка условной точки останова".

# Замечания

*Если CommandString* не указан, все команды, уже заданные в точке останова, удаляются.

Такой же эффект можно добиться с помощью **команды bs (обновить команду точки останова)** со следующим синтаксисом:

```
dbgcmd
```

```
bs ID "j Condition 'CommandString'; 'gc'"
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# c (сравнение памяти)

Статья • 08.07.2024

Команда **c** сравнивает значения, содержащиеся в двух областях памяти.

```
dbgcmd
```

```
c Range Address
```

## Параметры

*Диапазон*
Первый из двух диапазонов памяти, которые необходимо сравнить. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес".

*Адрес*
Начальный адрес второго диапазона памяти, который необходимо сравнить. Размер этого диапазона будет совпадать с указанным для первого диапазона. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

Если две области не идентичны, отладчик будет отображать все адреса памяти в первом диапазоне, где они не согласны.

В качестве примера рассмотрим следующий код:

C++

```
void main()
{
    char rgBuf1[100];
    char rgBuf2[100];

    memset(rgBuf1, 0xCC, sizeof(rgBuf1));
    memset(rgBuf2, 0xCC, sizeof(rgBuf2));

    rgBuf1[42] = 0xFF;
}
```

Чтобы сравнить **rgBuf1** и **rgBuf2**, используйте любую из следующих команд:

dbgcmd

```
0:000> c rgBuf1 (rgBuf1+0n100) rgBuf2

0:000> c rgBuf1 L 0n100 rgBuf2
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# d, da, db, dc, dd, dD, df, dp, dq, du, dw (display Memory)

Статья • 05.04.2024

Команды **d*** отображают содержимое памяти в заданном диапазоне.

dbgcmd

```
d{a|b|c|d|D|f|p|q|u|w|W} [Options] [Range]
dy{b|d} [Options] [Range]
d [Options] [Range]
```

## Параметры

*Параметры*

Задает один или несколько параметров отображения. Вы можете включить любой из следующих параметров, но не более одного **параметра /p***.

**/c***Width*

Указывает количество столбцов, используемых в отображении. Если этот параметр не указан, число столбцов по умолчанию зависит от типа отображения.

**/p**

(только режим ядра) Использует адреса физической памяти для отображения. Диапазон, указанный диапазоном, берется из физической памяти, а не виртуальной памяти.

**/p[c]**

(только режим ядра) То же, что и **/p**, за исключением того, что кэшированный объем памяти считывается. Включите квадратные скобки вокруг **c**.

**/p[uc]**

(только режим ядра) То же самое, что **и /p**, за исключением того, что некэшированная память считывается. Включите квадратные скобки вокруг **uc**.

**/p[wc]**

(только режим ядра) То же, что и **/p**, за исключением того, что объем памяти, объединенной записью, считывается. Включите квадратные скобки вокруг **wc**.

*Диапазон*

Указывает область памяти для отображения. Дополнительные сведения о синтаксисе см. в разделе "Адрес" и "Диапазон адресов". Если опущен **диапазон**, команда отображает память, начиная с конечного расположения последней команды отображения. Если вы опустите **диапазон** и отсутствует предыдущая команда отображения, отображение начинается с текущего указателя инструкции.

# Среда

**Режимы**: режим пользователя, режим ядра

**Целевые объекты**: динамический, аварийный дампов

**Платформы**: все

# Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

Каждая строка, отображаемая, включает адрес первого байта в строке, а затем содержимое памяти в этом и следующих расположениях.

Если опущен **диапазон**, команда отображает память, начиная с конечного расположения последней команды отображения. Такой подход позволяет непрерывно сканировать память.

Эта команда существует в следующих формах. Второе символы команд dd**, dD, dw и dW чувствительны к регистру, так как являются третьими символами команд dyb и dyd.**

Развернуть таблицу

| Команда | Отображать |
|---|---|
| d | Эта команда отображает данные в том же формате, что и последняя **команда d*** . Если предыдущая **команда d* не была выдана, d* имеет тот же эффект,** что **и** |

| Команда | Отображать |
| --- | --- |
| | **db**. Обратите внимание, что **d** повторяет самую последнюю команду, начинающуюся с *d*. Эти команды включают dda, ddp**, ddu, dpa, dpp, dpu, dqa, dqp, dqu, dds, dps, dqs, ds, dS, dg, dl, dt, dv и отображаемые команды в этой статье.** Если параметры, заданные после *d* , не соответствуют, могут привести к ошибкам. |
| da | Символы ASCII. Каждая строка отображает до 48 символов. Отображение продолжается до первого байта NULL или до отображения всех символов в диапазоне. Все непечатываемые символы, такие как возвраты каретки и каналы строк, отображаются в виде периодов (.). |
| db | Значения байтов и символы ASCII. Каждая строка отображения показывает адрес первого байта в строке, за которым следует до 16 шестнадцатеричных байтовых значений. За значениями байтов сразу же следует соответствующие значения ASCII. Восьмое и девятое шестнадцатеричные значения разделены дефисом (-). Все непечатываемые символы, такие как возвраты каретки и каналы строк, отображаются в виде периодов (.). Число по умолчанию — 128 байт. |
| dc | Значения двойного слова (4 байта) и символы ASCII. Каждая строка отображения показывает адрес первого слова в строке и до восьми шестнадцатеричных значений слов и их эквивалент ASCII. Число по умолчанию — 32 DWORD (128 байт). |
| dd | Значения двойного слова (4 байта). Число по умолчанию — 32 DWORD (128 байт). |
| Dd | Числа с плавающей запятой двойной точности (8 байт). Число по умолчанию — 15 чисел (120 байт). |
| df | Числа с плавающей запятой с одной точностью (4 байта). Число по умолчанию — 16 чисел (64 байта). |
| dp | Значения размера указателя. Эта команда эквивалентна **dd** или **dq** в зависимости от того, является ли архитектура целевого процессора компьютера 32-разрядной или 64-разрядной соответственно. По умолчанию используется число 32 DWORD или 16 четырех слов (128 байт). |
| dq | Значения четырех слов (8 байт). Число по умолчанию — 16 четырех слов (128 байт). |
| du | Знаки Юникода. Каждая строка отображает до 48 символов. Отображение продолжается до первого байта NULL или до отображения всех символов в диапазоне. Все непечатываемые символы, такие как возвраты каретки и каналы строк, отображаются в виде периодов (.). |
| dw | Значения Word (2 байта). Каждая строка отображения показывает адрес первого слова в строке и до восьми шестнадцатеричных значений слова. Число по умолчанию — 64 слова (128 байт). |

| Команда | Отображать |
| --- | --- |
| Dw | Значения Word (2 байта) и символы ASCII. Каждая строка отображения показывает адрес первого слова в строке и до восьми шестнадцатеричных значений слова. Число по умолчанию — 64 слова (128 байт). |
| dyb | Двоичные значения и байтовые значения. Число по умолчанию — 32 байта. |
| dyd | Двоичные значения и значения двойного слова (4 байта). Число по умолчанию — 8 DWORD (32 байта). |

Если вы пытаетесь отобразить недопустимый адрес, его содержимое отображается как вопросительные знаки (**?**).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# dda, ddp, ddu, dpa, dpp, dpu, dqa, dqp, dqu (отображаемая память со ссылкой)

Статья • 08.07.2024

Команды **dda**, **ddp, ddu**, **dpa, dpp, dpu**, dqa, **dqa, dqp** и **dqu** отображают указатель в указанном расположении, разыменовывая указатель, а затем отображают память в результирующем расположении в различных форматах.

dbgcmd

```
ddp [Options] [Range]
dqp [Options] [Range]
dpp [Options] [Range]
dda [Options] [Range]
dqa [Options] [Range]
dpa [Options] [Range]
ddu [Options] [Range]
dqu [Options] [Range]
dpu [Options] [Range]
```

## Параметры

*Параметры* указывают один или несколько параметров отображения. Можно включить любой из следующих параметров, за исключением того, что можно указать не более одного **параметра /p***:

**/c***Width*
Указывает количество столбцов, используемых в отображении. Если это опущено, число столбцов по умолчанию зависит от типа отображения. Из-за того, что указатели отображаются этими командами, обычно рекомендуется использовать только один столбец данных по умолчанию.

**/p**
(только в режиме ядра) Использует адреса физической памяти для отображения. Диапазон, указанный диапазоном, будет взят из физической памяти, а не виртуальной памяти.

**/p[c]**
(только в режиме ядра) То же, что и **/p**, за исключением того, что кэшированная память будет считываться. Квадратные скобки вокруг **c** должны быть включены.

**/p[uc]**
(только в режиме ядра) То же, что и **/p**, за исключением того, что некэшированная память будет считываться. Квадратные скобки вокруг **uc** должны быть включены.

**/p[wc]**
(только в режиме ядра) То же самое, что **и /p**, за исключением того, что объем памяти, объединенной записью, будет считываться. Квадратные скобки вокруг **wc** должны быть включены.

*Диапазон*
Указывает область памяти для отображения. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес". Если опущен *диапазон*, команда будет отображать память, начиная с конечного расположения последней команды отображения. Если *диапазон* опущен, и предыдущая команда отображения не использовалась, отображение начинается с текущего указателя инструкции. Если задан простой адрес, длина диапазона по умолчанию составляет 128 байт.

# Среда

**Режимы**: режим пользователя, режим ядра

**Целевые объекты**: динамический, аварийный дампов

**Платформы**: все

# Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

Второй и третий символы этой команды чувствительны к регистру.

Второй символ этой команды определяет используемый размер указателя:

Развернуть таблицу

| Команда | Отображать |
|---|---|
| dd | 32-разрядные указатели, используемые |

| Команда | Отображать |
| --- | --- |
| dq | 64-разрядные указатели, используемые |
| dp* | Стандартные размеры указателей: 32-разрядная или 64-разрядная в зависимости от архитектуры процессора целевого объекта |

Третий символ этой команды определяет, как отображается разыменовываемая память:

Развернуть таблицу

| Команда | Отображать |
| --- | --- |
| dp | Отображает содержимое памяти, на которую ссылается указатель в формате DWORD или QWORD, в зависимости от размера указателя архитектуры процессора целевого объекта. Если это значение соответствует любому известному символу, этот символ также отображается. |
| da | Отображает содержимое памяти, на которую ссылается указатель в формате символов ASCII. |
| d*u | Отображает содержимое памяти, на которую ссылается указатель в формате символов Юникода. |

Если сведения о номере строки включены, имена исходных файлов и номера строк будут отображаться при наличии.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# dds, dps, dqs (отображаемые слова и символы)

Статья • 05.04.2024

Команды **dds**, **dps** и **dqs** отображают содержимое памяти в заданном диапазоне. Предполагается, что эта память представляет собой ряд адресов в таблице символов. Также отображаются соответствующие символы.

dbgcmd

```
dds [Options] [Range]
dqs [Options] [Range]
dps [Options] [Range]
```

## Параметры

*Параметры* указывают один или несколько параметров отображения. Можно включить любой из следующих параметров, за исключением того, что можно указать не более одного **параметра /p***:

**/c***Width*
Указывает количество столбцов, используемых в отображении. Если это опущено, число столбцов по умолчанию зависит от типа отображения. Из-за того, как символы отображаются этими командами, обычно рекомендуется использовать только один столбец данных по умолчанию.

**/p**
(только в режиме ядра) Использует адреса физической памяти для отображения. Диапазон, указанный диапазоном, будет взят из физической памяти, а не виртуальной памяти.

**/p[c]**
(только в режиме ядра) То же, что и **/p**, за исключением того, что кэшированная память будет считываться. Квадратные скобки вокруг **c** должны быть включены.

**/p[uc]**
(только в режиме ядра) То же, что и **/p**, за исключением того, что некэшированная память будет считываться. Квадратные скобки вокруг **uc** должны быть включены.

**/p[wc]**
(только в режиме ядра) То же самое, что **и /p**, за исключением того, что объем

памяти, объединенной записью, будет считываться. Квадратные скобки вокруг **wc** должны быть включены.

*Диапазон*
Указывает область памяти для отображения. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес". Если опущен *диапазон*, команда будет отображать память, начиная с конечного расположения последней команды отображения. Если *диапазон* опущен, и предыдущая команда отображения не использовалась, отображение начинается с текущего указателя инструкции. Если задан простой адрес, длина диапазона по умолчанию составляет 128 байт.

# Среда

**Режимы**: режим пользователя, режим ядра

**Целевые объекты**: динамический, аварийный дампов

**Платформы**: все

# Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

Второй символ **dds** учитывает регистр. Третий символ всех этих команд учитывает регистр.

Команда **dds** отображает значения двойного слова (4 байта), такие как **команда dd** . Команда **dqs** отображает значения четырехкратного слова (8 байтов), например **команду dq** . Команда **dps** отображает значения размера указателя (4 байта или 8 байтов в зависимости от архитектуры целевого компьютера), например **команды DP** .

Каждое из этих слов рассматривается как адрес в таблице символов. Для каждого слова отображаются соответствующие сведения о символах.

Если сведения о номере строки включены, имена исходных файлов и номера строк будут отображаться при наличии.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# dg (селектор отображения)

Статья • 08.07.2024

Команда **dg** показывает дескриптор сегмента для указанного селектора.

```
dbgcmd

dg FirstSelector [LastSelector]
```

## Параметры

*FirstSelector*
Указывает шестнадцатеричное значение селектора первого селектора, отображаемого.

*LastSelector*
Указывает шестнадцатеричное значение селектора последнего селектора, отображаемого. Если это опущено, отобразится только один селектор.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | режим пользователя, режим ядра |
| **Целевые объекты** | live, аварийное дампа |
| **Платформы** | x86 |

## Замечания

С помощью этой команды можно отобразить не более 256 селекторов.

Распространенные значения селектора:

Развернуть таблицу

| Артикул | десятичное | hex |
|---|---|---|
| KGDT_NULL | 0 | 0x00 |

| Артикул | десятичное | hex |
|---|---|---|
| KGDT_R0_CODE | 8 | 0x08 |
| KGDT_R0_DATA | 16 | 0x10 |
| KGDT_R3_CODE | 24 | 0x18 |
| KGDT_R3_DATA | 32 | 0x20 |
| KGDT_TSS | 40 | 0x28 |
| KGDT_R0_PCR | 48 | 0x30 |
| KGDT_R3_TEB | 56 | 0x38 |
| KGDT_VDM_TILE | 64 | 0x40 |
| KGDT_LDT | 72 | 0x48 |
| KGDT_DF_TSS | 80 | 0x50 |
| KGDT_NMI_TSS | 88 | 0x58 |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# dl (отображаемый связанный список)

Статья • 08.07.2024

Команда **dl** отображает связанный список LIST_ENTRY или SINGLE_LIST_ENTRY.

```
dbgcmd
```

```
dl[b] Address MaxCount Size
```

## Параметры

**b**
Если это включено, список сбрасывается в обратном порядке. (Другими словами, отладчик следует **Blink s вместо Flink**s.) Это нельзя использовать с SINGLE_LIST_ENTRY.

*Адрес*
Начальный адрес списка. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес".

*MaxCount*
Максимальное количество элементов для дампа.

*Размер*
Размер каждого элемента. Это число последовательных ULONG_PTRs, которые будут отображаться для каждого элемента в списке.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

## Замечания

Этот список должен быть LIST_ENTRY или SINGLE_LIST_ENTRY структурой. Если это внедрено в большую структуру, убедитесь, что *адрес* указывает на структуру связанного списка, а не на начало внешней структуры.

Отображение начинается с *адреса*. Таким образом, если вы указываете адрес указателя, указывающего на начало списка, следует игнорировать первый элемент, напечатанный.

Параметры *Address*, *MaxCount* и *Size* находятся в текущем радиксе по умолчанию. Для изменения радикса можно использовать команду n (set Number Base) **или префикс 0x.**

Если список циклов выполняется обратно, дамп остановится. Если обнаружен указатель null, дампа остановится.

Если вы хотите выполнить определенную команду для каждого элемента списка, используйте **расширение !list** .

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ds, dS (отображаемая строка)

Статья • 08.07.2024

Команды **ds** и **dS** отображают структуры STRING, ANSI_STRING или UNICODE_STRING.

Эти команды не отображают строки символов с разделителями NULL, а скорее строковые структуры.

Если у вас есть адрес символов строки Юникода, вместо этого используйте команду du. Используйте команду da для отображения символов ASCII. Дополнительные сведения см. в разделе d, da, db, dc, dd, dD, df, dp, dq, du, dw (display Memory).

dbgcmd

```
d{s|S} [/c Width] [Address]
```

# Параметры

**s**
Указывает, что должна отображаться структура STRING или ANSI_STRING. (Это регистр учитывается.)

**S**
Указывает, что должна отображаться UNICODE_STRING структура. (Это **S** учитывает регистр.)

***/c** Ширина*
Указывает количество символов, отображаемых в каждой строке. Это число содержит пустые символы, которые не будут видимы.

*Адрес*
Адрес памяти, в котором хранится структура UNICODE_STRING.

Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес". Если опущено, предполагается последний адрес, используемый в команде отображения.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

## Замечания

Если вы хотите отобразить строки Юникода в окне "Локальные" или в окне "Просмотр" WinDbg, сначала необходимо использовать **команду .enable_unicode (включить отображение Юникода).**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# dt (тип отображения)

Статья • 08.07.2024

Команда **dt** отображает сведения о локальной переменной, глобальной переменной или типе данных. Это может отображать сведения о простых типах данных, а также структурах и объединениях.

Синтаксис пользовательского режима

dbgcmd

```
dt [-DisplayOpts] [-SearchOpts] [module!]Name [[-SearchOpts] Field]
[Address] [-l List]
dt [-DisplayOpts] Address [-l List]
dt -h
```

Синтаксис режима ядра

dbgcmd

```
[Processor] dt [-DisplayOpts] [-SearchOpts] [module!]Name [[-SearchOpts]
Field] [Address] [-l List]
dt [-DisplayOpts] Address [-l List]
dt -h
```

# Параметры

*Процессор*
Указывает процессор, выполняющий процесс, содержащий необходимые сведения. Дополнительные сведения см. в разделе "Многопроцессорный синтаксис". Процессоры могут быть указаны только в режиме ядра.

*DisplayOpts*
Указывает один или несколько параметров, указанных в следующей таблице. Перед этими параметрами предшествует дефис.

Развернуть таблицу

| Вариант | Описание |
| --- | --- |
| **-a**[*количество*] | Отображение каждого элемента массива в новой строке с его индексом. Будет отображаться общее *количество* элементов. |

| Вариант | Описание |
| --- | --- |
| | Между количеством и количеством не должно быть места. Если **значение -a** не следует за цифрой, отображаются все элементы в массиве. Переключатель -**a**[*quantity*] должен отображаться непосредственно перед каждым именем типа или именем поля, отображаемым таким образом. |
| **-b** | Отображение блоков рекурсивно. Если отображаемая структура содержит подструктуры, она развертывается рекурсивно на произвольные глубины и отображается в полном объеме. Указатели развертываются только в том случае, если они находятся в исходной структуре, а не в подструктурах. |
| **-c** | Компактные выходные данные. Все поля отображаются в одной строке, если это возможно. (При использовании с **параметром -a** каждый элемент массива принимает одну строку, а не форматируется как блок с несколькими строками.) |
| **-d** | При использовании с *именем* , заканчивающимся звездочкой, выводятся подробные выходные данные для всех типов, начинающихся с *name*. Если *имя* не заканчивается звездочкой, отображает подробные выходные данные. |
| **-e** | Принудительно перечисляет **типы dt** . Этот параметр необходим только в том случае, если **dt** ошибочно интерпретирует *значение Name* как экземпляр, а не как тип. |
| **-i** | Не отступайте подтипы. |
| **-o** | Не указывайте значения смещения полей структуры. |
| **-p** | *Адрес* — это физический адрес, а не виртуальный адрес. |
| **-r**[*глубина*] | Рекурсивно дамп полей подтипа. Если *задана глубина* , это рекурсия остановится после *уровней глубины* . Глубина *должна быть цифрой от 1 до 9, и между r **и*** |

| Вариант | Описание |
| --- | --- |
| | ***глубиной не должно быть места***. Переключатель **-r**[*глубина*] должен отображаться непосредственно перед адресом. |
| **-s** *размер* | Перечисляет только те типы, размер которых в байтах равен значению *размера*. Параметр **-s** полезен только при перечислении типов. Если **задано значение -s** , **-e** всегда подразумевается. |
| **-t** | Перечислить только типы. |
| **-v** | Подробные выходные данные. Это дает дополнительные сведения, такие как общий размер структуры и количество его элементов. Если это используется вместе с параметром **поиска -y** , отображаются все символы, даже те, у которых нет сведений о связанном типе. |

*SearchOpts*

Указывает один или несколько параметров, указанных в следующей таблице. Перед этими параметрами предшествует дефис.

Развернуть таблицу

| Вариант | Описание |
| --- | --- |
| **-n** | Это означает, что следующий параметр является именем. Это следует использовать, если следующий элемент полностью состоит из шестнадцатеричных символов, так как в противном случае он будет приниматься в качестве адреса. |
| **-y** | Это означает, что следующий параметр является началом имени, а не все имя. Если **параметр -y** включен, все совпадения перечислены, а затем подробные сведения о первом совпадении в списке. Если **значение -y** не включено, будут отображаться только точные совпадения. |

*модуль*

Необязательный параметр, указывающий модуль, определяющий эту структуру.

Если есть локальная переменная или тип с тем же именем, что и глобальная переменная или тип, следует включить модуль, *чтобы указать*, что означает глобальную переменную. В **противном случае команда dt** отобразит локальную переменную, даже если локальная переменная не учитывает регистр, а глобальная переменная — это совпадение с учетом регистра.

*Имя*
Указывает имя типа или глобальной переменной. Если *имя* заканчивается звездочкой (*), отображается список всех совпадений. Таким образом, **dt A*** выводит список всех типов данных, глобальных и статических значений, начиная с "A", но не будет отображать фактические экземпляры этих типов. (Если **параметр отображения -v** используется одновременно, все символы будут отображаться не только с данными связанного типа.) Вы также можете заменить *имя* точкой (**.**) для обозначения того, что вы хотите повторить последнее используемое значение *Name*.

Если *имя* содержит пробел, он должен быть заключен в скобки.

*Поле*
Указывает отображаемые поля. Если *поле* опущено, отображаются все поля. Если *за полем* следует период (.), то также будут отображаться подфилды первого уровня этого поля. Если *за полем* следует ряд периодов, подфилды будут отображаться на глубину, равную количеству периодов. Любое имя поля, за которым следует период, будет рассматриваться как совпадение префикса, как если **бы использовался параметр поиска -y** . Если *за полем* следует звездочка (*), оно обрабатывается как только начало поля, а не все соответствующие поля отображаются.

*Адрес*
Указывает адрес отображаемой структуры. Если *имя* опущено, *адрес* должен быть включен и должен указать адрес глобальной переменной. *Адрес* принимается как виртуальный адрес, если не указано иное. **Используйте параметр -p**, чтобы указать физический адрес. Используйте знак at ( **@** ), чтобы указать регистр (например, **@eax**).

*Список*
Указывает имя поля, которое связывает связанный список. Параметр *Address* должен быть включен.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

Выходные **данные команды dt** всегда будут отображать подписанные номера в базовой версии 10 и без знака в шестнадцатеричном виде.

Все параметры dt**, разрешающие значения символов**, также разрешают строковые подстановочные знаки. Дополнительные сведения см . в синтаксисе подстановочных знаков строки.

Параметры **-y** и **-n** могут предшествовать любому *имени* или *полю*. Параметр **-y** позволяет указать начало типа или имени структуры. Например, **dt-y ALLEN** отображает данные о типе **ALLENTOWN**. Однако не удалось отобразить тип **ALLENTOWN** с **dt-y A**. Вместо этого необходимо использовать dt-ny A**, так как A является допустимым шестнадцатеричным значением и будет интерпретирован как адрес без параметра -n.**

Если *имя* указывает структуру, будут отображаться все поля (например, **dt myStruct**). Если требуется только одно конкретное поле, можно выполнить **dt myStruct myField**. Откроется элемент, вызывающий **myStruct.myField**. Однако обратите внимание, что команда **dt myStruct myField1 myField2** отображает **myStruct.myField1** и **myStruct.myField2**. Он не отображает **myStruct.myField1.myField2**.

Если за именем структуры или полем следует подстрок, это указывает один экземпляр массива. Например, **dt myStruct myFieldArray[3]** отображает четвертый элемент массива. Но если за именем типа следует подстрок, это указывает весь массив. Например, **dt CHAR[8] myPtr** будет отображать восьмизначную строку.

Подстрок всегда принимается как десятичный, независимо от текущего радикса; **Префикс 0x** приведет к ошибке.

Так как команда использует сведения о типе из *.PDB-файл* можно свободно использовать для отладки любой платформы ЦП.

Сведения о типе, используемые **dt** , включают все имена типов, созданные с **типом**, включая все определяемые Windows типы. Например, **без знака длинные** и **символы** являются недопустимыми именами типов, но **ULONG** и **CHAR** . Полный список всех имен типов Windows см. в пакете SDK для Microsoft Windows.

Все типы, созданные **typedefs** в собственном коде, будут присутствовать, пока они фактически использовались в вашей программе. Однако типы, определенные в заголовках, но никогда не используемые на самом деле не будут храниться в файлах символов PDB и не будут доступны отладчику. Чтобы сделать такой тип доступным для отладчика, используйте его в качестве *входных данных* инструкции **typedef** . Например, если в коде отображается следующее, структура MY_DATA будет храниться в PDB-файле символов и может отображаться командой **dt** :

dbgcmd

```
typedef struct _MY_DATA {
    . . .
    } MY_DATA;
typedef  MY_DATA *PMY_DATA;
```

С другой стороны, следующий код не будет достаточно, поскольку оба MY_DATA и PMY_DATA определяются начальным **типизированным и, следовательно, MY_DATA не использовался в качестве входных данных любого оператора typedef**:

dbgcmd

```
typedef struct _MY_DATA {
    . . .
    } MY_DATA, *PMY_DATA;
```

В любом случае сведения о типе включаются только в полный файл символов, а не в файл символов, который был лишен всех сведений о закрытых символах. Дополнительные сведения см. в разделе "Открытые и частные символы".

Если вы хотите отобразить строки юникода, сначала необходимо использовать команду .enable_unicode (включить отображение Юникода). Вы можете управлять отображением длинных целых чисел с помощью **команды .enable_long_status (включить длинное целое число).**

В следующем примере **dt** отображает глобальную переменную:

dbgcmd

```
0:000> dt mt1
   +0x000 a                : 10
   +0x004 b                : 98 'b'
   +0x006 c                : 0xdd
   +0x008 d                : 0xabcd
   +0x00c gn               : [6] 0x1
   +0x024 ex               : 0x0
```

В следующем примере **dt** отображает gn **поля** массива:

dbgcmd

```
0:000> dt mt1 -a gn
   +0x00c gn :
    [00] 0x1
    [01] 0x2
    [02] 0x3
    [03] 0x4
    [04] 0x5
    [05] 0x6
```

Следующая команда отображает некоторые подфилды переменной:

dbgcmd

```
0:000> dt mcl1 m_t1 dpo
   +0x010 dpo  : DEEP_ONE
   +0x070 m_t1 : MYTYPE1
```

Следующая команда отображает подфилды поля поля **m_t1**. Так как период автоматически вызывает сопоставление префиксов, это также будет отображать подфилды любого поля, начинающегося с "m_t1":

dbgcmd

```
0:000> dt mcl1 m_t1.
   +0x070 m_t1  :
      +0x000 a     : 0
      +0x004 b     : 0 '
      +0x006 c     : 0x0
      +0x008 d     : 0x0
      +0x00c gn    : [6] 0x0
      +0x024 ex    : 0x0
```

Вы можете повторить это на любую глубину. Например, команда **dt mcl1 a.. c.** отображает все поля для глубины четырех, таким образом, что имя первого поля началось с **имени** третьего поля, начинающегося с **c**.

Ниже приведен более подробный пример отображения подфилдов. Во-первых, отобразите **поле Ldr** :

dbgcmd

```
0:000> dt nt!_PEB Ldr 7ffdf000
   +0x00c Ldr : 0x00191ea0
```

Теперь разверните поле типа указателя:

dbgcmd

```
0:000> dt nt!_PEB Ldr Ldr. 7ffdf000
   +0x00c Ldr  : 0x00191ea0
      +0x000 Length : 0x28
      +0x004 Initialized : 0x1 '
      +0x008 SsHandle : (null)
      +0x00c InLoadOrderModuleList : _LIST_ENTRY [ 0x191ee0 - 0x192848 ]
      +0x014 InMemoryOrderModuleList : _LIST_ENTRY [ 0x191ee8 - 0x192850 ]
      +0x01c InInitializationOrderModuleList : _LIST_ENTRY [ 0x191f58 -
0x192858 ]
      +0x024 EntryInProgress : (null)
```

Теперь отобразится **поле CriticalSectionTimeout** :

dbgcmd

```
0:000> dt nt!_PEB CriticalSectionTimeout 7ffdf000
   +0x070 CriticalSectionTimeout : _LARGE_INTEGER 0xffffe86d`079b8000
```

Теперь разверните **подфилды структуры CriticalSectionTimeout** на одном уровне:

dbgcmd

```
0:000> dt nt!_PEB CriticalSectionTimeout. 7ffdf000
   +0x070 CriticalSectionTimeout  :  0xffffe86d`079b8000
      +0x000 LowPart                 : 0x79b8000
      +0x004 HighPart                : -6035
      +0x000 u                       : __unnamed
      +0x000 QuadPart                : -25920000000000
```

Теперь разверните **подфилды структуры CriticalSectionTimeout** на двух уровнях:

dbgcmd

```
0:000> dt nt!_PEB CriticalSectionTimeout.. 7ffdf000
   +0x070 CriticalSectionTimeout   :  0xffffe86d`079b8000
      +0x000 LowPart                  : 0x79b8000
      +0x004 HighPart                 : -6035
      +0x000 u                        :
         +0x000 LowPart                  : 0x79b8000
         +0x004 HighPart                 : -6035
      +0x000 QuadPart                 : -25920000000000
```

Следующая команда отображает экземпляр типа данных MYTYPE1, расположенный по адресу 0x0100297C:

dbgcmd

```
0:000> dt 0x0100297c MYTYPE1
   +0x000 a                : 22
   +0x004 b                : 43 '+'
   +0x006 c                : 0x0
   +0x008 d                : 0x0
   +0x00c gn               : [6] 0x0
   +0x024 ex               : 0x0
```

Следующая команда отображает массив из 10 ULONG по адресу 0x01002BE0:

dbgcmd

```
0:000> dt -ca10 ULONG 01002be0
[0] 0x1001098
[1] 0x1
[2] 0xdead
[3] 0x7d0
[4] 0x1
[5] 0xcd
[6] 0x0
[7] 0x0
[8] 0x0
[9] 0x0
```

Следующая команда продолжает отображение предыдущего адреса по другому адресу. Обратите внимание, что "ULONG" не требуется повторно вводить:

dbgcmd

```
0:000> dt -ca4 . 01002d00
Using sym ULONG
[0] 0x12
[1] 0x4ac
```

```
[2] 0xbadfeed
[3] 0x2
```

Ниже приведены некоторые примеры отображения типов. Следующая команда отображает все типы и глобальные значения, начиная со строки "MY" в модуле этого модуля. Эти префиксы с адресом являются фактическими экземплярами; те, которые без адресов являются определениями типов:

dbgcmd

```
0:000> dt thismodule!MY*
010029b8  thismodule!myglobal1
01002990  thismodule!myglobal2
          thismodule!MYCLASS1
          thismodule!MYCLASS2
          thismodule!MYCLASS3
          thismodule!MYTYPE3::u
          thismodule!MYTYPE1
          thismodule!MYTYPE3
          thismodule!MYTYPE3
          thismodule!MYFLAGS
```

При отображении типа **параметр -v** можно использовать для отображения размера каждого элемента. Параметр **размера -*s*** можно использовать только для перечисления элементов определенного размера. Опять же, эти префиксы с адресом являются фактическими экземплярами; те, которые без адресов являются определениями типов:

dbgcmd

```
0:001> dt -s 2 -v thismodule!*
Enumerating symbols matching thismodule!*, Size = 0x2
Address   Size Symbol
           002 thismodule!wchar_t
           002 thismodule!WORD
           002 thismodule!USHORT
           002 thismodule!SHORT
           002 thismodule!u_short
           002 thismodule!WCHAR
00427a34   002 thismodule!numberOfShips
00427a32   002 thismodule!numberOfPlanes
00427a30   002 thismodule!totalNumberOfItems
```

Ниже приведен пример **параметра -b** . Структура расширяется и **массив OwnerThreads** в структуре расширяется, но **указатели списка Flink и Blink** не следуют:

dbgcmd

```
kd> dt nt!_ERESOURCE -b 0x8154f040
   +0x000 SystemResourcesList :  [ 0x815bb388 - 0x816cd478 ]
      +0x000 Flink            : 0x815bb388
      +0x004 Blink            : 0x816cd478
   +0x008 OwnerTable       : (null)
   +0x00c ActiveCount      : 1
   +0x00e Flag             : 8
   +0x010 SharedWaiters    : (null)
   +0x014 ExclusiveWaiters : (null)
   +0x018 OwnerThreads     :
    [00]
      +0x000 OwnerThread      : 0
      +0x004 OwnerCount       : 0
      +0x004 TableSize        : 0
    [01]
      +0x000 OwnerThread      : 0x8167f563
      +0x004 OwnerCount       : 1
      +0x004 TableSize        : 1
   +0x028 ContentionCount  : 0
   +0x02c NumberOfSharedWaiters : 0
   +0x02e NumberOfExclusiveWaiters : 0
   +0x030 Address          : (null)
   +0x030 CreatorBackTraceIndex : 0
   +0x034 SpinLock         : 0
```

Ниже приведен пример **dt** в режиме ядра. Следующая команда создает результаты, аналогичные **!process 0 0**:

dbgcmd

```
kd> dt nt!_EPROCESS -l ActiveProcessLinks.Flink -y Ima -yoi Uni 814856f0
## ActiveProcessLinks.Flink at 0x814856f0

UniqueProcessId : 0x00000008
ImageFileName : [16] "System"

## ActiveProcessLinks.Flink at 0x8138a030

UniqueProcessId : 0x00000084
ImageFileName : [16] "smss.exe"

## ActiveProcessLinks.Flink at 0x81372368

UniqueProcessId : 0x000000a0
ImageFileName : [16] "csrss.exe"

## ActiveProcessLinks.Flink at 0x81369930

UniqueProcessId : 0x000000b4
ImageFileName : [16] "winlogon.exe"
```

```
....
```

Если вы хотите выполнить команду для каждого элемента списка, используйте **расширение !list** .

Наконец, **команда dt -h** отобразит короткий текст справки, суммируя **синтаксис dt** .

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# dtx (тип отображения — сведения о расширенной объектной модели отладчика)

Статья • 08.07.2024

Команда dtx отображает расширенные сведения о символьных типах с помощью объектной модели отладчика. Команда dtx аналогична команде dt (display Type).

dbgcmd

```
dtx -DisplayOpts [Module!]Name Address
```

## Параметры

**DisplayOpts**
Используйте следующие необязательные флаги, чтобы изменить способ отображения выходных данных.

*-a* Отображает элементы массива в новой строке с его индексом.

*-r [n]* Рекурсивно дамп подтипов (полей) до *n* уровней.

*-h* Отображает справку командной строки для команды dtx.

**Модуль!**
Необязательный параметр, указывающий модуль, определяющий эту структуру, а затем восклицательный знак. Если есть локальная переменная или тип с тем же именем, что и глобальная переменная или тип, следует включить имя модуля, *чтобы указать* глобальную переменную.

**Имя**
Имя типа или глобальный символ.

**Адрес**
Адрес памяти, содержащий тип.

## Среда

Развернуть таблицу

| Позиция | Description |
| --- | --- |
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

В следующих примерах показано, как использовать команду dtx.

Используйте адрес и имя для отображения сведений о расширенном символьных типах.

dbgcmd

```
0: kd> dtx nt!_EPROCESS ffffb607560b56c0
(*((nt!_EPROCESS *)0xffffb607560b56c0))                [Type: _EPROCESS]
    [+0x000] Pcb              [Type: _KPROCESS]
    [+0x2d8] ProcessLock      [Type: _EX_PUSH_LOCK]
    [+0x2e0] RundownProtect   [Type: _EX_RUNDOWN_REF]
    [+0x2e8] UniqueProcessId  : 0x4 [Type: void *]
    [+0x2f0] ActiveProcessLinks [Type: _LIST_ENTRY]
```

Отображение дополнительных сведений с помощью параметра рекурсии -r.

dbgcmd

```
0: kd> dtx -r2 HdAudio!CAzMixertopoMiniport fffff806`d24992b8
(*((HdAudio!CAzMixertopoMiniport *)0xfffff806d24992b8))
[Type: CAzMixertopoMiniport]
    [+0x018] m_lRefCount      : -766760880 [Type: long]
    [+0x020] m_pUnknownOuter  : 0xfffff806d24dbc40 [Type: IUnknown *]
    [+0x028] m_FilterDesc     [Type: PCFILTER_DESCRIPTOR]
        [+0x000] Version          : 0xd24c2890 [Type: unsigned long]
        [+0x008] AutomationTable  : 0xfffff806d24c2780 [Type:
PCAUTOMATION_TABLE *]
            [+0x000] PropertyItemSize : 0x245c8948 [Type: unsigned long]
            [+0x004] PropertyCount    : 0x6c894808 [Type: unsigned long]
            [+0x008] Properties       : 0x5718247489481024 [Type:
PCPROPERTY_ITEM *]
            [+0x010] MethodItemSize   : 0x55415441 [Type: unsigned long]
            [+0x014] MethodCount      : 0x57415641 [Type: unsigned long]
            [+0x018] Methods          : 0x4ce4334540ec8348 [Type:
PCMETHOD_ITEM *]
            [+0x020] EventItemSize    : 0x8b41f18b [Type: unsigned long]
            [+0x024] EventCount       : 0xd8b48f4 [Type: unsigned long]
            [+0x028] Events           : 0x7d2d8d4cfffdf854 [Type:
```

```
PCEVENT_ITEM *]
            [+0x030] Reserved         : 0x66fffd79 [Type: unsigned long]
        [+0x010] PinSize          : 0xd24aa9b0 [Type: unsigned long]
        [+0x014] PinCount         : 0xfffff806 [Type: unsigned long]
        [+0x018] Pins             : 0xfffff806d24aa740 [Type:
PCPIN_DESCRIPTOR *]
            [+0x000] MaxGlobalInstanceCount : 0x57555340 [Type: unsigned
long]
            [+0x004] MaxFilterInstanceCount : 0x83485741 [Type: unsigned
long]
            [+0x008] MinFilterInstanceCount : 0x8b4848ec [Type: unsigned
long]
            [+0x010] AutomationTable  : 0xa5158b48ed33c000 [Type:
PCAUTOMATION_TABLE *]
            [+0x018] KsPinDescriptor  [Type: KSPIN_DESCRIPTOR]
```

Совет. Используйте **команду x (Проверить символы),** чтобы отобразить адрес интересующего элемента.

dbgcmd

```
0: kd> x /d HdAudio!CazMixertopoMiniport*
...
fffff806`d24992b8 HdAudio!CAzMixertopoMiniport::`vftable' = <no type
information>
...
```

# См. также

**dt (тип отображения)**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# dv (отображение локальных переменных)

Статья • 08.07.2024

Команда **dv** отображает имена и значения всех локальных переменных в текущей области.

```
dbgcmd
```

```
dv [Flags] [Pattern]
```

## Параметры

*Флаги*
Приводит к отображению дополнительных сведений. Можно включить любой из следующих флагов с учетом *регистра* :

**/f**<addr>
Позволяет указать произвольный адрес функции, чтобы увидеть, какие параметры и локальные параметры существуют для любого кода в любом месте. Он отключает отображение значения и подразумевает **/V**. Флаг **/f** должен быть последним флагом. Шаблон фильтра параметров по-прежнему можно указать после того, как он указан в кавычках.

**/i**
Приводит к указанию типа переменной: локальной, глобальной, параметров, функции или неизвестной.

**/t**
Приводит к включению типа данных для каждой локальной переменной.

**/v**
Вызывает отображение, чтобы включить адрес виртуальной памяти или зарегистрировать расположение каждой локальной переменной.

**/V**
То же, что **и /v**, а также адрес локальной переменной относительно соответствующего регистра.

**/a**
Сортирует выходные данные по адресу в порядке возрастания.

**/A**
Сортирует выходные данные по адресу в порядке убывания.

**/n**
Сортировка выходных данных по имени в порядке возрастания.

**/N**
Сортировка выходных данных по имени в порядке убывания.

**/z**
Сортировка выходных данных по размеру в порядке возрастания.

**/Z**
Сортировка выходных данных по размеру в порядке убывания.

*Рисунок*
Позволяет команде отображать только локальные переменные, соответствующие указанному *шаблону*. Шаблон может содержать различные подстановочные знаки и описатели; Дополнительные сведения см. в синтаксисе подстановочных знаков строки. Если *шаблон* содержит пробелы, он должен быть заключен в кавычки. Если *шаблон* опущен, будут отображаться все локальные переменные.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения о отображении и изменении локальных переменных и описании других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

В подробном режиме адреса переменных отображаются также. (Это также можно сделать с помощью **команда x (проверка символов** ).)

Структуры данных и незнакомые типы данных не отображаются в полном объеме; вместо этого отображается имя типа. Чтобы отобразить всю структуру или отобразить определенный элемент структуры, используйте **команду dt (тип отображения** ).

Локальный *контекст* определяет, какой набор локальных переменных будет отображаться. По умолчанию этот контекст соответствует текущей позиции счетчика программы. Сведения о том, как это можно изменить, см. в разделе "Локальный контекст".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# dx (выражение объектной модели отладчика)

Статья • 20.05.2024

Команда **dx** отображает выражение C++ с помощью модели расширения NatVis. Дополнительные сведения о NatVis см. в разделе "Создание пользовательских представлений собственных объектов".

dbgcmd

```
dx [-g|-gc #][-c #][-n|-v]-r[#] Expression[,<FormatSpecifier> ]
dx [{-?}|{-h}]
```

## Параметры

*Выражение*

Отображаемое выражение C++ .

**-g**

Отображение в виде объектов сетки данных, которые являются итерируемыми. Каждый итерированный элемент является строкой в сетке, и каждый отображаемый дочерний элемент этих элементов является столбцом. Это позволяет просматривать такие элементы, как массив структур, где каждый элемент массива отображается в строке, а каждое поле структуры отображается в столбце.

Выбор имени столбца (где есть доступная ссылка DML) будет отсортирован по такому столбцу. Если этот столбец уже отсортирован по указанному столбцу, порядок сортировки будет инвертирован.

Любой объект, который является итерируемым, будет иметь пункт контекстного меню выбора и удержания (или щелкните правой кнопкой мыши), добавленный через DML с именем Display as Grid. Выбор и удержание объекта (или щелчка правой кнопкой мыши) в окне вывода и выбор этого объекта будет отображаться в представлении сетки вместо стандартного представления дерева.

Значение (+), отображаемое именем столбца, предлагает поведение выбора и удержания (или правой кнопкой мыши).

- Выберите этот столбец и взорвал его в собственную таблицу. Вы увидите исходные строки и дочерние элементы развернутого столбца.
- Выберите и удерживайте (или щелкните правой кнопкой мыши) предоставляет команду "Развернуть в сетку", которая принимает столбец и добавляет его обратно в текущую таблицу в качестве правых столбцов.

**-Gc #**

Отображение в виде сетки и ограничение размеров ячеек сетки указанным числом символов (#).

**-c #** Отображает продолжение контейнера (пропуск элементов контейнера). Этот параметр обычно используется в пользовательских сценариях автоматизации выходных данных и предоставляет "..." Элемент продолжения в нижней части списка.

**-n** Существует два способа отрисовки данных. Использование визуализации NatVis (по умолчанию) или использование базовых собственных структур C/C++. Укажите параметр -n для отрисовки выходных данных, используя только собственные структуры C/C++, а не визуализации NatVis.

**-v**

Отображение подробных сведений, включающих методы и другие нетипичную объекты.

**-r**#

Рекурсивно отображают подтипы (поля) до # уровней. Если # значение по умолчанию не указано, то значением по умолчанию является уровень рекурсии одного.

**[<,FormatSpecifier>]**

Используйте любой из следующих описателей формата для изменения отрисовки по умолчанию.

Развернуть таблицу

| Спецификатор формата | Description |
|---|---|
| X | Отображение порядковых порядков в шестнадцатеричном формате |
| D | Отображение порядковых порядков в десятичном разряде |

| Спецификатор формата | Description |
|---|---|
| O | Отображение порядковых порядков в восьмерике |
| B | Отображение порядковых порядков в двоичном файле |
| En | Отображение перечислений только по имени (без значения) |
| C | Отображение в виде одного символа (а не строки) |
| ,s | Отображение 8-разрядных строк в кавычках ASCII |
| Sb | Отображение 8-разрядных строк в виде неквалированных ASCII |
| ,s8 | Отображение 8-разрядных строк в кавычках UTF-8 |
| ,s8b | Отображение 8-разрядных строк в формате UTF-8 без параметров |
| Cy | Отображение 16-разрядных строк в кавычках UTF-16 |
| Sub | Отображение 16-разрядных строк в формате UTF-16 unqouted |
| ,! | Отображение объектов только в необработанном режиме (например, нет NatVis) |
| , # | Укажите длину указателя/ массива или контейнера в качестве литерального значения # (замените числовым значением) |
| ,[<выражение>] | Укажите длину указателя, массива или контейнера в качестве выражения выражения <> |
| Nd | Не найдите производный (runtype) тип объекта. Отображение только статического значения |

**dx-?**

Отображение справки командной строки.

**Dx-h** Отображает справку по объектам, доступным в отладчике.

**dx-id**

Только для внутреннего использования в корпорации Майкрософт. Используется для выполнения ссылок модели данных в выходных данных команды.

# Пример использования командной строки

Команда параметров .dx может использоваться для отображения сведений об объекте Отладки Параметры. Дополнительные сведения о объектах параметров

отладки см. в разделе .settings.

dbgcmd

```
kd> dx -r1 Debugger.Settings
Debugger.Settings
    Debug
    Display
    EngineInitialization
    Extensions
    Input
    Sources
    Symbols
    AutoSaveSettings : false
```

Используйте параметр рекурсии -r1, чтобы просмотреть другие объекты отладчика — сеансы, Параметры и состояние.

dbgcmd

```
kd> dx -r1 Debugger
Debugger
  Sessions
  Settings
  State
  Utility
  LastEvent
```

Укажите объект Debugger.Session с параметром рекурсии -r3 для дальнейшего перемещения по цепочке объектов.

dbgcmd

```
kd> dx -r3 Debugger.Sessions
Debugger.Sessions
  [0]              : Remote KD:
KdSrv:Server=@{<Local>},Trans=@{1394:Channel=0}
    Processes
      [0]              : <Unknown Image>
      [4]              : <Unknown Image>
      [304]            : smss.exe
      [388]            : csrss.exe
      [456]            : wininit.exe
      [468]            : csrss.exe
      [528]            : services.exe
      [536]            : lsass.exe
      [544]            : winlogon.exe
      [620]            : svchost.exe
       ...               ...
```

Добавьте описатель формата x, чтобы отобразить порядковые значения в шестнадцатеричном формате.

dbgcmd

```
kd> dx -r3 Debugger.Sessions,x
Debugger.Sessions,x
  [0x0]            : Remote KD:
KdSrv:Server=@{<Local>},Trans=@{1394:Channel=0}
    Processes
      [0x0]            : <Unknown Image>
      [0x4]            : <Unknown Image>
      [0x130]          : smss.exe
      [0x184]          : csrss.exe
      [0x1c8]          : wininit.exe
      [0x1d4]          : csrss.exe
      [0x210]          : services.exe
      [0x218]          : lsass.exe
      [0x220]          : winlogon.exe
      [0x26c]          : svchost.exe
      [0x298]          : svchost.exe
      [0x308]          : dwm.exe
      [0x34c]          : nvvsvc.exe
      [0x37c]          : nvvsvc.exe
      [0x384]          : svchost.exe
       ...               ...
```

В этом примере используется активный сеанс отладки для перечисления стека вызовов первого потока в первом процессе.

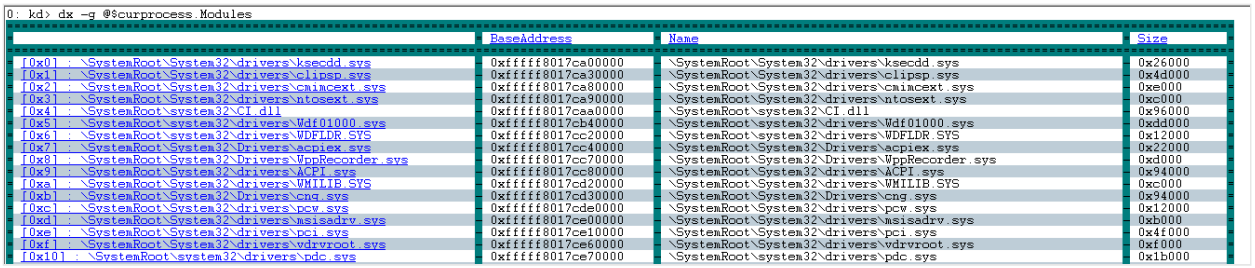
dbgcmd

```
kd> dx -r1
Debugger.Sessions.First().Processes.First().Threads.First().Stack.Frames
Debugger.Sessions.First().Processes.First().Threads.First().Stack.Frames
    [0x0]            : nt!RtlpBreakWithStatusInstruction
    [0x1]            : nt!KdCheckForDebugBreak + 0x7a006
    [0x2]            : nt!KiUpdateRunTime + 0x42
    [0x3]            : nt!KiUpdateTime + 0x129
    [0x4]            : nt!KeClockInterruptNotify + 0x1c3
    [0x5]            : hal!HalpTimerClockInterruptEpilogCommon + 0xa
    [0x6]            : hal!HalpTimerClockInterruptCommon + 0x3e
    [0x7]            : hal!HalpTimerClockInterrupt + 0x1cb
    [0x8]            : nt!KiIdleLoop + 0x1a
```

Используйте параметр -g для отображения выходных данных в виде сетки данных. Выберите столбец для сортировки.

dbgcmd

```
kd> dx -g @$curprocess.Modules
```




Используйте параметр -h для отображения сведений об объектах.

dbgcmd

```
kd>  dx -h Debugger.State
Debugger.State   [State pertaining to the current execution of the debugger
(e.g.: user variables)]
    DebuggerVariables [Debugger variables which are owned by the debugger
and can be referenced by a pseudo-register prefix of @$]
    PseudoRegisters   [Categorizied debugger managed pseudo-registers which
can be referenced by a pseudo-register prefix of @$]
    UserVariables     [User variables which are maintained by the debugger
and can be referenced by a pseudo-register prefix of @$]
```

# Отображение сведений TEB и PEB с помощью объекта Environment

Используйте объект Environment для отображения сведений TEB и PEB, связанных с потоком и процессом.

Чтобы отобразить TEB, связанный с текущим потоком, используйте эту команду.

dbgcmd

```
0: kd> dx -r2 @$curthread.Environment
@$curthread.Environment
    EnvironmentBlock [Type: _TEB]
        [+0x000] NtTib            [Type: _NT_TIB]
        [+0x038] EnvironmentPointer : Unable to read memory at Address 0x38
        [+0x040] ClientId         [Type: _CLIENT_ID]
        [+0x050] ActiveRpcHandle  : Unable to read memory at Address 0x50
        [+0x058] ThreadLocalStoragePointer : Unable to read memory at
Address 0x58
        [+0x060] ProcessEnvironmentBlock : Unable to read memory at Address
0x60
        [+0x068] LastErrorValue   : Unable to read memory at Address 0x68
        [+0x06c] CountOfOwnedCriticalSections : Unable to read memory at
```

```
Address 0x6c
        [+0x070] CsrClientThread  : Unable to read memory at Address 0x70
        [+0x078] Win32ThreadInfo  : Unable to read memory at Address 0x78
        [+0x080] User32Reserved   [Type: unsigned long [26]]
        [+0x0e8] UserReserved     [Type: unsigned long [5]]
        [+0x100] WOW32Reserved    : Unable to read memory at Address 0x100
        [+0x108] CurrentLocale    : Unable to read memory at Address 0x108
        [+0x10c] FpSoftwareStatusRegister : Unable to read memory at Address
0x10c
         ...
```

Для отображения PEB, связанного с текущим процессом, используйте эту команду.

dbgcmd

```
0: kd> dx -r2 @$curprocess.Environment
@$curprocess.Environment
    EnvironmentBlock [Type: _PEB]
        [+0x000] InheritedAddressSpace : Unable to read memory at Address
0x0
        [+0x001] ReadImageFileExecOptions : Unable to read memory at Address
0x1
        [+0x002] BeingDebugged    : Unable to read memory at Address 0x2
        [+0x003] BitField         : Unable to read memory at Address 0x3
        [+0x003 ( 0: 0)] ImageUsesLargePages : Unable to read memory at
Address 0x3
        [+0x003 ( 1: 1)] IsProtectedProcess : Unable to read memory at
Address 0x3
        [+0x003 ( 2: 2)] IsImageDynamicallyRelocated : Unable to read memory
at Address 0x3
        [+0x003 ( 3: 3)] SkipPatchingUser32Forwarders : Unable to read
memory at Address 0x3
        [+0x003 ( 4: 4)] IsPackagedProcess : Unable to read memory at
Address 0x3
        [+0x003 ( 5: 5)] IsAppContainer   : Unable to read memory at Address
0x3
        [+0x003 ( 6: 6)] IsProtectedProcessLight : Unable to read memory at
Address 0x3
        [+0x003 ( 7: 7)] IsLongPathAwareProcess : Unable to read memory at
Address 0x3
        [+0x004] Padding0         [Type: unsigned char [4]]
        [+0x008] Mutant           : Unable to read memory at Address 0x8
        [+0x010] ImageBaseAddress : Unable to read memory at Address 0x10
        [+0x018] Ldr              : Unable to read memory at Address 0x18
        [+0x020] ProcessParameters : Unable to read memory at Address 0x20
        ...
```

# Объект Kernel Io.Handles

Используйте текущий объект Io.Handles для отображения сведений об дескрипторе ядра.

dbgcmd

```
0: kd> dx -r1 @$curprocess.Io.Handles
@$curprocess.Io.Handles
    [0x8]
    [0xc]
    [0x10]
    [0x14]
    [0x18]
    ...
```

Используйте . Функция First() для отображения сведений о первом дескрипторе.

dbgcmd

```
0: kd> dx -r2 @$curprocess.Io.Handles.First()
@$curprocess.Io.Handles.First()
    Handle           : 0x8
    Type             : Unexpected failure to dereference object
    GrantedAccess    : Unexpected failure to dereference object
    Object           [Type: _OBJECT_HEADER]
        [+0x000] PointerCount     : 228806 [Type: __int64]
        [+0x008] HandleCount      : 6 [Type: __int64]
        [+0x008] NextToFree       : 0x6 [Type: void *]
        [+0x010] Lock             [Type: _EX_PUSH_LOCK]
        [+0x018] TypeIndex        : 0xf2 [Type: unsigned char]
        [+0x019] TraceFlags       : 0x0 [Type: unsigned char]
        [+0x019 ( 0: 0)] DbgRefTrace      : 0x0 [Type: unsigned char]
        [+0x019 ( 1: 1)] DbgTracePermanent : 0x0 [Type: unsigned char]
        [+0x01a] InfoMask         : 0x0 [Type: unsigned char]
        [+0x01b] Flags            : 0x2 [Type: unsigned char]
        [+0x01b ( 0: 0)] NewObject        : 0x0 [Type: unsigned char]
        [+0x01b ( 1: 1)] KernelObject     : 0x1 [Type: unsigned char]
        [+0x01b ( 2: 2)] KernelOnlyAccess : 0x0 [Type: unsigned char]
        [+0x01b ( 3: 3)] ExclusiveObject  : 0x0 [Type: unsigned char]
        [+0x01b ( 4: 4)] PermanentObject  : 0x0 [Type: unsigned char]
        [+0x01b ( 5: 5)] DefaultSecurityQuota : 0x0 [Type: unsigned char]
        [+0x01b ( 6: 6)] SingleHandleEntry : 0x0 [Type: unsigned char]
        [+0x01b ( 7: 7)] DeletedInline    : 0x0 [Type: unsigned char]
        [+0x01c] Reserved         : 0x0 [Type: unsigned long]
        [+0x020] ObjectCreateInfo : 0xfffff801f6d9c6c0 [Type:
_OBJECT_CREATE_INFORMATION *]
        [+0x020] QuotaBlockCharged : 0xfffff801f6d9c6c0 [Type: void *]
        [+0x028] SecurityDescriptor : 0xffffb984aa815d06 [Type: void *]
        [+0x030] Body             [Type: _QUAD]
        ObjectType       : Unexpected failure to dereference object
        UnderlyingObject : Unexpected failure to dereference object
```

Обратите внимание, что объект Io.Handles является единственным объектом ядра.

## Обход ограничений файлов символов с приведением

При отображении сведений о различных системных переменных Windows возникает время, когда не все сведения о типе доступны в общедоступных символах. Этот пример иллюстрирует эту ситуацию.

dbgcmd

```
0: kd> dx nt!PsIdleProcess
Error: No type (or void) for object at Address 0xfffff800e1d50128
```

Команда dx поддерживает возможность ссылаться на адрес переменной, которая не содержит сведений о типе. Такие "адреса" ссылок рассматриваются как "void*" и могут быть приведены как такие. Это означает, что если тип данных известен, для отображения сведений о типе переменной можно использовать следующий синтаксис.

dbgcmd

```
dx (Datatype *)&VariableName
```

Например, для nt! PsIdleProcess, имеющий тип данных nt!_EPROCESS, используйте эту команду.

dbgcmd

```
dx (nt!_EPROCESS *)&nt!PsIdleProcess
(nt!_EPROCESS *)&nt!PsIdleProcess                 : 0xfffff800e1d50128
[Type: _EPROCESS *]
    [+0x000] Pcb              [Type: _KPROCESS]
    [+0x2c8] ProcessLock      [Type: _EX_PUSH_LOCK]
    [+0x2d0] CreateTime       : {4160749568} [Type: _LARGE_INTEGER]
    [+0x2d8] RundownProtect   [Type: _EX_RUNDOWN_REF]
    [+0x2e0] UniqueProcessId  : 0x1000 [Type: void *]
    [+0x2e8] ActiveProcessLinks [Type: _LIST_ENTRY]
    [+0x2f8] Flags2           : 0x218230 [Type: unsigned long]
    [+0x2f8 ( 0: 0)] JobNotReallyActive : 0x0 [Type: unsigned long]
```

Команда dx не поддерживает переключение вычислителей выражений с синтаксисом @@ MASM. Дополнительные сведения об вычислителях выражений см. в разделе "Оценка выражений".

# Использование LINQ С объектами отладчика

Синтаксис LINQ можно использовать с объектами отладчика для поиска и управления данными. LINQ концептуально похож на язык SQL (SQL), который используется для запроса баз данных. Для поиска, фильтрации и анализа данных отладки можно использовать ряд методов LINQ. Сведения об использовании LINQ с объектами отладчика см. в разделе "Использование LINQ С объектами отладчика".

# Использование объектов отладчика с NatVis и JavaScript

Сведения об использовании объектов отладчика с NatVis см. в разделе "Собственные объекты отладчика" в NatVis.

Сведения об использовании объектов отладчика с JavaScript см. в разделе "Собственные объекты отладки" в расширениях JavaScript.

# См. также

Использование LINQ С объектами отладчика

Собственные объекты отладчика в NatVis

Собственные объекты отладчика в расширениях JavaScript

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# e, ea, eb, ed, eD, ef, ep, eq, eu, ew, eza (ввод значений)

Статья • 08.07.2024

Команды `*e*` введите в память указанные значения.

Эта команда не должна быть путана с квалификатором **~E (команда** для конкретного потока).

dbgcmd

```
e{b|d|D|f|p|q|w} Address [Values]
e{a|u|za|zu} Address "String"
e Address [Values]
```

Только режим ядра — физический адрес

dbgcmd

```
/p {[c]|[uc]|[wc]}
```

Дополнительные сведения см. в разделе "Физические адреса в режиме ядра" далее в этом разделе.

# Параметры

## Синтаксис

*Адрес*
Указывает начальный адрес, в котором нужно ввести значения. Отладчик заменяет значение по *адресу и* каждому последующему расположению памяти до тех пор, пока не будут использованы все *значения* .

*Значения*
Указывает одно или несколько значений для ввода в память. Несколько числовых значений должны быть разделены пробелами. Если вы не указываете никаких значений, будет отображаться текущий адрес и значение по указанному адресу, и вам будет предложено ввести данные.

*Струна*
Указывает строку для ввода в память. Команды **ea** и **eza** записывают это в память в виде строки ASCII; **команды eu** и **ezu** записывают это в память в виде строки Юникода. Команды **eza** и **ezu** записывают терминал **NULL**; **команды ea** и **eu** не выполняются. *Строка* должна быть заключена в кавычки.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

Эта команда существует в следующих формах. Второй символ **команд ed** и **eD** учитывает регистр.

Развернуть таблицу

| Команда | Введите |
|---|---|
| **e** | Он вводит данные в том же формате, что и последняя **команда e**. *(Если последние* команда e ea, eza**, eu** или **ezu**, окончательный параметр будет *строкой* и не может быть опущен.) |
| **ea** | Строка ASCII (не завершающаяся значением NULL). |
| **eb** | Значения байтов. |
| **Эд** | Значения двойного слова (4 байта). |

| Команда | Введите |
| --- | --- |
| **Эд** | Числа с плавающей запятой двойной точности (8 байт). |
| **ef** | Числа с плавающей запятой с одной точностью (4 байта). |
| **ep** | Значения размера указателя. Эта команда эквивалентна **ed** или **eq** в зависимости от того, является ли архитектура процессора целевого компьютера 32-разрядной или 64-разрядной соответственно. |
| **eq** | Значения четырех слов (8 байт). |
| **eu** | Строка Юникода (не завершающаяся null). |
| **РЭБ** | Значения Word (2 байта). |
| **eza** | Строка ASCII с завершением NULL. |
| **ezu** | Строка Юникода с завершением NULL. |

Числовые значения интерпретируются как числа в текущем радиксе (16, 10 или 8). Чтобы изменить радикс по умолчанию, используйте **команду n (Set Number Base).** Радикс по умолчанию можно переопределить, указав **префикс 0x** (шестнадцатеричный), **префикс 0n** (десятичный **), префикс 0t** (восьмерику) или **0y** префикс (двоичный).

**Обратите внимание** , что радикс по умолчанию работает по-разному при использовании выражений C++. Дополнительные сведения см. в статье об оценке выражений .

При вводе **байтовых значений с помощью команды EB** можно использовать одинарные кавычки для указания символов. Если требуется включить несколько символов, каждый из них должен быть окружен собственными кавычками. Это позволяет ввести символьную строку, которая не завершается пустым символом. Например:

dbgcmd

```
eb 'h' 'e' 'l' 'l' 'o'
```

Экранные символы в стиле C (например `\\0` , или `\\n`) не могут использоваться с этими командами.

Если параметр "Значения" опущен, вам будет предложено ввести данные. Отобразится адрес и его текущее содержимое, и **появится запрос ввода>** . Затем можно выполнить любое из следующих действий:

- Введите новое значение, введя значение и нажав клавишу ВВОД.
- Сохраните текущее значение в памяти, нажав клавишу ПРОБЕЛ, а затем ВВОД.
- Выполните выход из команды, нажав клавишу ВВОД.

## Физические адреса в режиме ядра

Команды WinDbg версии 1.2402.24001.0 и более поздних `e` версий поддерживают физические адреса, такие как d, da, db, dc, dd, dd, df, dp, dq, du, dw, dW, dyb, dyd (память). Эти параметры поддерживаются только в режиме ядра.

Развернуть таблицу

| Вариант | Описание |
|---|---|
| `/p` | Использует адреса физической памяти для отображения. Диапазон, указанный диапазоном, будет взят из физической памяти, а не виртуальной памяти. |
| `/p[c]` | Аналогично тому, что `/p`кэшированная память будет считываться. Квадратные скобки вокруг с должны быть включены. |
| `/p[uc]` | То же самое `/p`, что, за исключением того, что некэшированная память будет считываться. Квадратные скобки вокруг uc должны быть включены. |
| `/p[wc]` | Так же, как `/p`и в случае с памятью, объединенной записью, будет считываться. Квадратные скобки вокруг wc должны быть включены. |

## См. также

Чтение и запись памяти

d, da, db, dc, dd, dD, df, dp, dq, du, dw, dW, dyb, dyd (память отображения)

Вычисление выражений

---

## Обратная связь

**Были ли сведения на этой странице полезными?**

Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# f, fp (заполнение памяти)

Статья • 08.07.2024

Команды **f** и **fp** заполняют указанный диапазон памяти повторяющимся шаблоном.

Эти команды не следует путать с командой **~F (Замораживание потока** ).

dbgcmd

```
f Range Pattern
fp [MemoryType] PhysicalRange Pattern
```

## Параметры

*Диапазон*
Задает диапазон в виртуальной памяти для заполнения. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес".

*PhysicalRange*
(только режим ядра) Задает диапазон в физической памяти для заполнения. Синтаксис PhysicalRange *совпадает с синтаксисом* диапазона виртуальной памяти, за исключением того, что имена символов не разрешены.

*MemoryType*
(только режим ядра) Указывает тип физической памяти, который может быть одним из следующих вариантов:

**[c]**
Кэшированная память.

**[uc]**
Некшированная память.

**[wc]**
Объединенная память записи.

*Рисунок*
Указывает одно или несколько байтовых значений, с помощью которых заполняется память.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | **f: пользовательский режим, режим ядра fp:** только режим ядра |
| **Целевые объекты** | live, аварийное дампа |
| **Платформы** | all |

# Дополнительная информация

Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

Эта команда заполняет область памяти, указанную диапазоном, заданным заданным *шаблоном*, повторяемым столько раз, сколько необходимо.

Параметр *шаблона* должен быть входным в виде ряда байтов. Их можно ввести как числовые или как символы ASCII.

Числовые значения интерпретируются как числа в текущем радиксе (16, 10 или 8). Чтобы изменить радикс по умолчанию, используйте **команду n (Set Number Base).** Радикс по умолчанию можно переопределить, указав **префикс 0x** (шестнадцатеричный), **префикс 0n** (десятичный **), префикс 0t** (восьмерику) или **0y** префикс (двоичный).

**Обратите внимание** , что радикс по умолчанию работает по-разному при использовании выражений C++. Дополнительные сведения см. в разделе "Оценка выражений ".

Если используются символы ASCII, каждый символ должен быть заключен в одну прямую кавычки. Экранные символы в стиле C (например, \0 или \n) не могут использоваться.

Если задано несколько байтов, они должны быть разделены пробелами.

Если *шаблон* имеет больше значений, чем число байтов в диапазоне, отладчик игнорирует дополнительные значения.

Ниже приведено несколько примеров. Если текущий радикс равен 16, следующая команда заполняет расположения памяти 0012FF40 до 0012FF5F с шаблоном ABC, повторяющиеся несколько раз:

dbgcmd

```
0:000> f 0012ff40 L20 'A' 'B' 'C'
```

Следующая команда имеет тот же эффект:

dbgcmd

```
0:000> f 0012ff40 L20 41 42 43
```

В следующих примерах показано, как использовать типы физической памяти (**c**, **uc** и **wc**) с **командой fp** в режиме ядра:

dbgcmd

```
kd> fp [c] 0012ff40 L20 'A' 'B' 'C'
```

dbgcmd

```
kd> fp [uc] 0012ff40 L20 'A' 'B' 'C'
```

dbgcmd

```
kd> fp [wc] 0012ff40 L20 'A' 'B' 'C'
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# g (Go)

Статья • 26.10.2023

Команда **g** начинает выполнение заданного процесса или потока. Выполнение будет остановлено в конце программы, при *попадании BreakAddress* или при другом событии, которое приводит к остановке отладчика.

Синтаксис User-Mode

```
dbgcmd
```

```
[~Thread] g[a] [= StartAddress] [BreakAddress ... [; BreakCommands]]
```

Синтаксис Kernel-Mode

```
dbgcmd
```

```
g[a] [= StartAddress] [BreakAddress ... [; BreakCommands]]
```

# Параметры

*Поток*

(Только в пользовательском режиме) Указывает поток для выполнения. Дополнительные сведения о синтаксисе см. в разделе Синтаксис потока.

**a**

Указывает, что любая точка останова, созданная этой командой, будет точкой останова процессора (например, точкой останова, созданной **ba**), а не точкой останова программного обеспечения (например, точками останова, созданными **bp** и **bm**). Если *параметр BreakAddress* не указан, точка останова не создается, **и флаг не** действует.

*StartAddress*

Указывает адрес, с которого должно начаться выполнение. Если этот параметр не указан, отладчик передает выполнение по адресу, указанному текущим значением указателя инструкции. Дополнительные сведения о синтаксисе см. в разделе Синтаксис адресов и диапазонов адресов.

*BreakAddress*

Указывает адрес точки останова. Если указан *параметр BreakAddress* , необходимо указать адрес инструкции (то есть адрес должен содержать первый байт инструкции). Одновременно можно указать до десяти адресов останова в любом порядке. Если не удается разрешить *BreakAddress* , он сохраняется как неразрешенная точка останова. Дополнительные сведения о синтаксисе см. в разделе Синтаксис адресов и диапазонов адресов.

*BreakCommands*

Указывает одну или несколько команд, которые будут автоматически выполняться при достижении точки останова, заданной *параметром BreakAddress* . *Параметру BreakCommands* должна предшествовать точка с запятой. Если указано несколько значений *BreakAddress* , *BreakCommands* применяется ко всем из них.

**Примечание** Параметр *BreakCommands* доступен только при внедрении этой команды в командную строку, используемую другой командой, например в другой команде точки останова или в параметре или параметре события . В командной строке точка с запятой завершает команду **g** , а все дополнительные команды, перечисленные после точки с запятой, будут выполняться сразу после выполнения команды **g** .

## Среда

| Элемент | Описание |
|---|---|
| Режимы | пользовательский режим, режим ядра |
| Цели | Только динамическая отладка |
| Платформы | все |

## Дополнительные сведения

Другие методы выполнения этой команды и общие сведения о связанных командах см. в разделе Управление целевым объектом.

# Комментарии

Если указан *поток* , команда **g** выполняется с указанным потоком, размороженным и всеми остальными. Например, если указана команда **~123g**, **~#g** или **~*g** , указанные потоки будут размерожены, а все остальные — заморожены.

# См. также раздел

gu (Go Up)

gh (Go с обработанной обработкой исключения)

# gc (перейти из условной точки останова)

Статья • 26.10.2023

Команда **gc** возобновляет выполнение из условной точки останова тем же образом, который использовался для попадания в точку останова (пошаговое выполнение, трассировка или свободное выполнение). Это относится только к более старому стилю условных точек останова с использованием "j (условие) ..." выражение style, а не простая условная точка останова в стиле "/w". Дополнительные сведения см. в статье Настройка условной точки останова.

```
dbgcmd
```

```
gc
```

Хотя эта команда больше не так полезна для условных точек останова, ее по-прежнему можно использовать для точек останова, которые выполняют ведение журнала или другие действия без взлома отладчика. Например, можно написать точку останова, которая выглядит следующим образом:

```
dbgcmd
```

```
bp module!myFunction ".echo myFunction executed; gc"
```

Если вместо этого использовалась обычная команда "g", программа продолжит выполнение при пошаговом выполнении "myFunction", а не просто печатает сообщение и продолжает операцию шага.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | пользовательский режим, режим ядра |
| Цели | Только динамическая отладка |
| Платформы | все |

## Дополнительные сведения

Общие сведения о связанных командах см. в разделе Управление целевым объектом.

# Комментарии

При условной точке останова с использованием "j (условие) ..." expression включает в себя команду выполнения в конце, это должна быть команда **gc** .

Например, ниже приведен пример условной точки останова.

dbgcmd

```
0:000> bp Address "j (Condition) 'OptionalCommands'; 'gc' "
```

Если эта точка останова обнаружена и выражение имеет значение false, выполнение будет возобновлено с использованием того же типа выполнения, который использовался ранее. Например, если вы использовали команду **g (Go)** для достижения этой точки останова, выполнение будет свободно возобновлено. Но если вы достигли этой точки останова при пошаговом выполнении или трассировке, выполнение будет возобновлено с шагом или трассировкой.

С другой стороны, ниже приведена неправильная формулировка точки останова, так как выполнение всегда будет свободно возобновляться, даже если вы были шагами перед достижением точки останова:

dbgcmd

```
0:000> bp Address "j (Condition) 'OptionalCommands'; 'g' "
```

# gh (Go с обработкой исключений)

Статья • 08.07.2024

Команда **gh** помечает исключение данного потока как обработанное и позволяет потоку перезапустить выполнение по инструкции, вызвавшей исключение.

Синтаксис пользовательского режима

```
dbgcmd
```

```
[~Thread] gh[a] [= StartAddress] [BreakAddress ... [; BreakCommands]]
```

Синтаксис режима ядра

```
dbgcmd
```

```
gh[a] [= StartAddress] [BreakAddress ... [; BreakCommands]]
```

## Параметры

*Нить*
(Только в пользовательском режиме) Указывает поток для выполнения. Этот поток должен быть остановлен исключением. Сведения о синтаксисе см. в разделе "Синтаксис потока".

**a**
Вызывает любую точку останова, созданную этой командой, в точку останова процессора (например, созданную ba), а не точку останова программного обеспечения (например, созданную с помощью **bp** и **bm).** Если *BreakAddress* не указан, точка останова не создается, а флаг не действует.

*StartAddress*
Указывает адрес, с которого должен начинаться выполнение. Если это не указано, отладчик передает выполнение адресу, в котором произошло исключение. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес".

*BreakAddress*
Задает адрес точки останова. Если *указан параметр BreakAddress* , он должен указать адрес инструкции (т. е. адрес должен содержать первый байт инструкции). В любом порядке можно указать до десяти адресов останова. Если *не удается*

*разрешить BreakAddress* , он сохраняется как неразрешенная точка останова. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес".

*BreakCommands*
Указывает одну или несколько команд, которые будут автоматически выполняться при нажатии точки останова, *указанной в BreakAddress* . Параметр *BreakCommands* должен предшествовать точке с запятой. Если задано несколько *значений BreakAddress* , *к ним применяется BreakCommands* .

**Обратите внимание, *что* *параметр BreakCommands*** доступен только при внедрении этой команды в строку команды, используемой другой командой, например в другой команде точки останова или в пределах другого параметра или события. В командной строке точка с запятой завершит **команду gh, и все дополнительные команды, перечисленные после того, как точка с запятой будет выполнена сразу после выполнения команды gh**.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Другие методы выдачи этой команды и общие сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Если для задания точки останова используется *параметр BreakAddress* , эта новая точка останова будет активирована только текущим потоком. Другие потоки, выполняющие код в этом расположении, не будут остановлены.

Если *задан поток* , **команда gh** выполняется с указанным потоком unfrozen и всеми остальными замороженными. Например, если **указана команда ~123gh**,

**~#gh** или **~*gh** , указанные потоки hefrozen и все остальные заморожены.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# gn, gN (Go с исключением не обрабатывается)

Статья • 08.07.2024

**Команды gn** и **gN** продолжают выполнение заданного потока без маркировки исключения как обработанного. Это позволяет обработчику исключений приложения обрабатывать исключение.

Синтаксис пользовательского режима

```
dbgcmd
```

```
[~Thread] gn[a] [= StartAddress] [BreakAddress ... [; BreakCommands]]
[~Thread] gN[a] [= StartAddress] [BreakAddress ... [; BreakCommands]]
```

Синтаксис режима ядра

```
dbgcmd
```

```
gn[a] [= StartAddress] [BreakAddress ... [; BreakCommands]]
gN[a] [= StartAddress] [BreakAddress ... [; BreakCommands]]
```

# Параметры

*Нить*
(Только в пользовательском режиме) Указывает поток для выполнения. Этот поток должен быть остановлен исключением. Сведения о синтаксисе см. в разделе "Синтаксис потока".

**a**
Вызывает любую точку останова, созданную этой командой, в точку останова процессора (например, созданную ba), а не точку останова программного обеспечения (например, созданную с помощью **bp** и **bm).** Если *BreakAddress* не указан, точка останова не создается, а флаг не действует.

*StartAddress*
Указывает адрес, в котором должно начинаться выполнение. Если это не указано, отладчик передает выполнение адресу, в котором произошло исключение. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес".

*BreakAddress*
Задает адрес точки останова. Если *указан параметр BreakAddress* , он должен указать адрес инструкции (т. е. адрес должен содержать первый байт инструкции). В любом порядке можно указать до десяти адресов останова. Если *не удается разрешить BreakAddress* , он сохраняется как неразрешенная точка останова. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис диапазона адресов" и "Адрес".

*BreakCommands*
Указывает одну или несколько команд, которые будут автоматически выполняться при нажатии точки останова, *указанной в BreakAddress* . Параметр *BreakCommands* должен предшествовать точке с запятой. Если задано несколько *значений BreakAddress* , *к ним применяется BreakCommands* .

**Обратите внимание, *что* *параметр BreakCommands*** доступен только при внедрении этой команды в строку команды, используемой другой командой, например в другой команде точки останова или в пределах другого параметра или события. В командной строке точка с запятой завершит команду, и все дополнительные команды, перечисленные после того, как точка с запятой будет выполнена сразу после **выполнения команды gn** или **gN** .

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Другие методы выдачи этой команды и общие сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Если отладчик не остановлен в точке останова, **gn** и **gN** работают одинаково. Если отладчик остановлен в точке останова, **gn** не будет работать. Для выполнения этой команды необходимо прописной буквой "N". Это мера предосторожности, так как редко мудро продолжать точку останова необработанным.

Если для задания точки останова используется *параметр BreakAddress* , эта новая точка останова будет активирована только текущим потоком. Другие потоки, выполняющие код в этом расположении, не будут остановлены.

Если *задан поток* , **команда gn** выполняется с указанным потоком unfrozen и всеми остальными замороженными. Например, если **указана команда ~123gn**, **~#gn** или **~*gn** , указанные потоки не переключаются и все остальные заморожены.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# gu (Go Up)

Статья • 08.07.2024

Команда **gu** приводит к выполнению целевого объекта до завершения текущей функции.

Синтаксис пользовательского режима

```
dbgcmd
```

```
[~Thread] gu
```

Синтаксис режима ядра

```
dbgcmd
```

```
gu
```

## Параметры

*Нить*
(Только в пользовательском режиме) Указывает поток для выполнения. Этот поток должен быть остановлен исключением. Сведения о синтаксисе см. в разделе "Синтаксис потока".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Другие методы выдачи этой команды и общие сведения о связанных командах см. в разделе "Управление целевым объектом".

## Замечания

Команда **gu** выполняет целевой объект до тех пор, пока текущий вызов функции не возвращается.

Если текущая функция вызывается рекурсивно, команда gu **не остановит выполнение до тех пор,** пока *текущий экземпляр* текущей функции не возвращается. Таким образом, **gu** отличается от **g @$ra**, что будет остановлено в любой момент, когда возвращается адрес этой функции.

**Обратите внимание** , что **команда gu** различает различные экземпляры функции, измеряя глубину стека вызовов. Выполнение этой команды в режиме сборки после отправки аргументов в стек и непосредственно перед вызовом может привести к неправильному измерению. Возвращаемые функцией, оптимизированные компилятором, могут также привести к остановке этой команды в неправильном экземпляре этого возвращаемого значения. Эти ошибки редки и могут возникать только во время рекурсивных вызовов функций.

Если *задан поток* , **команда gu** выполняется с указанным потоком unfrozen и всеми остальными замороженными. Например, если **указана команда ~123gu**, **~#gu** или **~*gu** , указанные потоки не раскроются и все остальные заморожены.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ib, iw, id (входные данные из порта)

Статья • 08.07.2024

**Команды ib**, **iw** и **id** считывают и отображают байт, слово или двойное слово из выбранного порта.

dbgcmd

```
ib Address
iw Address
id Address
```

## Параметры

*Адрес*
Адрес порта.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Только компьютер на основе x86 |

## Замечания

Команда **IB** считывает один байт, **команда iw** считывает слово, а **команда id** считывает двойное слово.

Убедитесь, что чтение порта ввода-вывода не влияет на поведение устройства, которое вы читаете. Некоторые устройства изменяют состояние после чтения порта только для чтения. Кроме того, не следует пытаться читать слово или двойное слово из порта, который не разрешает значения этой длины.

## См. также

**ob, od, ow (выходные данные в порт)**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# j (Execute If - Else)

Статья • 08.07.2024

Команда **j** условно выполняет одну из указанных команд в зависимости от оценки заданного выражения.

dbgcmd

```
j Expression Command1 ; Command2
j Expression 'Command1' ; 'Command2'
```

## Параметры

*Выражение*
Выражение для вычисления. Если это выражение оценивается как ненулевое значение, *команда1* выполняется. Если это выражение равно нулю, *выполняется command2* . Дополнительные сведения о синтаксисе этого выражения см. в разделе "Синтаксис числовых выражений".

*Command1*
Командная строка, выполняемая, если выражение в *выражении* вычисляется как ненулевое значение (TRUE). Можно объединить несколько команд, окружив строку команды одними прямыми кавычками ( ' ) и разделив команды с запятой. Если строка команды является одной командой, одинарные кавычки являются необязательными.

*Command2*
Командная строка, выполняемая, если выражение в *выражении* вычисляется до нуля (FALSE). Можно объединить несколько команд, окружив строку команды одними прямыми кавычками ( ' ) и разделив команды с запятой. Если строка команды является одной командой, одинарные кавычки являются необязательными.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |

| Позиция | Description |
| --- | --- |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Невозможно добавить точку с запятой или дополнительные команды после **команды j** . Если точка с запятой отображается после *Command2*, все после запятой игнорируется.

Следующая команда отображает значение **eax,** если **MySymbol** равно нулю и отображает значения **ebx и ecx** в противном случае.

dbgcmd

```
0:000> j (MySymbol=0) 'r eax'; 'r ebx; r ecx'
```

Вы можете пропустить одинарные кавычки вокруг **rax**, но они упрощают чтение команды. Если вы хотите пропустить одну из команд, можно включить пустые кавычки или опустить параметр для этой команды, как показано в следующих командах.

dbgcmd

```
0:000> j (MySymbol=0) ''; 'r ebx; r ecx'
0:000> j (MySymbol=0)  ; 'r ebx; r ecx'
```

Вы также можете использовать **команду j** в других командах. Например, можно использовать **команду j** для создания условных точек останова.

dbgcmd

```
0:000> bp `mysource.cpp:143` "j (poi(MyVar)>0n20) ''; 'gc' "
```

Этот стиль условной точки останова больше не рекомендуется, так как более простая форма теперь доступна в отладчике. Дополнительные сведения о синтаксисе условных точек останова см. в разделе "Настройка условной точки останова".

# См. также

z (выполнение в то время как)

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# k, kb, kc, kd, kp, kP, kv (обратная трассировка стека отображения)

Статья • 26.10.2023

Команды **k*** отображают кадр стека заданного потока со связанными сведениями.

Пользовательский режим, процессор x86

dbgcmd

```
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] [FrameCount]
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] = BasePtr [FrameCount]
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] = BasePtr StackPtr InstructionPtr
[~Thread] kd [WordCount]
```

Режим ядра, процессор x86

dbgcmd

```
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] [FrameCount]
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr FrameCount
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] = BasePtr StackPtr InstructionPtr
[Processor] kd [WordCount]
```

Пользовательский режим, процессор x64

dbgcmd

```
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] [FrameCount]
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr FrameCount
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr InstructionPtr
FrameCount
[~Thread] kd [WordCount]
```

Режим ядра, процессор x64

dbgcmd

```
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] [FrameCount]
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr FrameCount
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr InstructionPtr
FrameCount
[Processor] kd [WordCount]
```

Пользовательский режим, процессор ARM

```
dbgcmd
```

```
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] [FrameCount]
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr FrameCount
[~Thread] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr InstructionPtr
FrameCount
[~Thread] kd [WordCount]
```

Режим ядра, процессор ARM

```
dbgcmd
```

```
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] [FrameCount]
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr FrameCount
[Processor] k[b|p|P|v] [c] [n] [f] [L] [M] = StackPtr InstructionPtr
FrameCount
[Processor] kd [WordCount]
```

# Параметры

*Поток*
Указывает отображаемый стек потоков. Если этот параметр не указан, отобразится стек текущего потока. Дополнительные сведения о синтаксисе потоков см. в разделе Синтаксис потока. Потоки можно указать только в пользовательском режиме.

*Процессор*
Указывает процессор, стек которого должен отображаться. Дополнительные сведения о синтаксисе процессора см. в разделе Синтаксис нескольких процессоров.

b
Отображает первые три параметра, которые передаются каждой функции в трассировке стека.

c
Отображает трассировку чистого стека. Каждая отображаемая строка содержит только имя модуля и имя функции.

p
Отображает все параметры для каждой функции, вызываемой в трассировке стека. Список параметров содержит тип данных, имя и значение каждого параметра.

Параметр `p` учитывает регистр. Для этого параметра требуются полные сведения о символах.

P
Отображает все параметры для каждой функции, вызываемой в трассировке стека, например `p` параметр . Однако для `P` параметры функции печатаются во второй строке отображения, а не в той же строке, что и остальные данные.

v
Отображает сведения об упущении указателя кадра (FPO). На процессорах на базе x86 дисплей также содержит сведения о соглашении о вызовах.

n
Отображает номера кадров.

f
Отображает расстояние между смежными кадрами. Это расстояние — количество байтов, разделяющих кадры в фактическом стеке.

L
Скрывает строки источника на экране. `L` с учетом регистра.

M
Отображает выходные данные с помощью языка разметки отладчика. Каждый номер кадра на экране — это ссылка, которую можно выбрать, чтобы задать локальный контекст и отобразить локальные переменные. Сведения о локальном контексте см. в разделе .frame.

*FrameCount*
Указывает количество отображаемых кадров стека. Это число следует указывать в шестнадцатеричном формате, если только вы не изменили радикс с помощью команды n (задать число). Используйте команду .kframes (задать длину стека), чтобы отобразить значение по умолчанию и изменить значение.

*BasePtr*
Указывает базовый указатель для трассировки стека. Параметр `BasePtr` доступен только при наличии знака равенства (=) после команды.

*StackPtr*
Указывает указатель стека для трассировки стека. Если опустить `StackPtr` и `InstructionPtr`, команда использует указатель стека, который указывает регистр rsp (или esp), и указатель инструкции, который указывает регистр rip (или eip).

*InstructionPtr*
Указывает указатель инструкции для трассировки стека. Если опустить `StackPtr` и `InstructionPtr`, команда использует указатель стека, который указывает регистр rsp (или esp), и указатель инструкции, который указывает регистр rip (или eip).

*WordCount*
Указывает количество DWORD_PTR значений в стеке для дампа.

| **Среда** | |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

При выполнении `k` команд , `kb`, `kp`, `kP` или `kv` трассировка стека отображается в табличном формате. Если загрузка строк включена, также отображаются исходные модули и номера строк.

Трассировка стека включает базовый указатель для кадра стека, обратный адрес и имена функций.

При использовании `kp` команд или `kP` отображаются полные параметры для каждой функции, вызываемой в трассировке стека. Список параметров содержит тип данных, имя и значение каждого параметра.

Команда может выполняться медленно. Например, при `MyFunction1` вызове `MyFunction2` отладчик должен иметь полные сведения о символах для `MyFunction1` отображения параметров, передаваемых в этом вызове. Эта команда не полностью отображает внутренние подпрограммы Microsoft Windows, которые не представлены в открытых символах.

При использовании `kb` команд или `kv` отображаются первые три параметра, которые передаются в каждую функцию. При использовании `kv` команды также отображаются данные FPO.

На процессоре x86 команда также отображает сведения о соглашении `kv` о вызовах.

При использовании `kv` команды сведения о FPO добавляются в конце строки в следующем формате.

| Текст FPO | Значение |
|---|---|
| FPO: [не Fpo] | Нет данных FPO для кадра. |
| FPO: [N1,N2,N3] | *N1* — общее количество параметров.<br><br>*N2* — количество значений DWORD для локальных переменных.<br><br>*N3* — количество сохраненных регистров. |
| FPO: [N1,N2] TrapFrame @ Address | *N1* — общее количество параметров.<br><br>*N2* — количество значений DWORD для локальных.<br><br>*Address* — это адрес кадра ловушки. |
| FPO: TaskGate Segment:0 | *Segment* — это селектор сегментов для шлюза задачи. |
| FPO: [EBP 0xBase] | *Base* — это базовый указатель для кадра. |

Команда `kd` отображает необработанные данные стека. Каждое значение DWORD отображается в отдельной строке. Сведения о символах отображаются для этих строк вместе со связанными символами. Этот формат создает более подробный список, чем другие `k*` команды. Команда `kd` эквивалентна команде dds (отображаемая память), которая использует адрес стека в качестве параметра.

Если вы используете `k` команду в начале функции (до выполнения пролога функции), вы получите неверные результаты. Отладчик использует регистр кадра для вычисления текущей обратной трассировки, и этот регистр неправильно задан для функции, пока не будет выполнен ее пролог.

В пользовательском режиме трассировка стека основана на стеке текущего потока. Дополнительные сведения о потоках см. в разделе Управление процессами и потоками.

В режиме ядра трассировка стека основана на текущем контексте регистра. Контекст регистра можно задать в соответствии с конкретным потоком, записью контекста или кадром ловушки.

# Дополнительные сведения

Дополнительные сведения о контексте регистрации и других параметрах контекста см. в разделе Изменение контекстов.

# l+, l- (установка параметров источника)

Статья • 26.10.2023

Команды **l+** и **l-** задают параметры исходной строки, управляющие отображением источника и параметрами пошагового выполнения программы.

dbgcmd

```
l+Option
l-Option
l{+|-}
```

# Параметры

**+** или **-**
Указывает, должен ли данный параметр быть включен (знак "плюс" [+]) или отключен (знак "минус" [-]).

*Параметр*

Один из следующих вариантов. Параметры должны быть строчными буквами.

**l**

Отображает номера строк источника в командной строке. Вы можете отключить отображение строки источника с помощью l-ls или .prompt_allow -src. Чтобы номера строк источника отображались, необходимо включить отображение исходной строки с помощью обоих механизмов.

**o**

Скрывает все сообщения (кроме исходной строки и номера строки) при пошаговом выполнении кода. (Параметр **s** также должен быть активным, чтобы параметр **o** был каким-либо эффектом.)

**s**
Отображает исходные строки и номера исходных строк в командной строке.

**t**

Запускает исходный режим. Если этот режим не задан, отладчик находится в режиме сборки.

*

Включает или отключает все параметры.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения об отладке исходного кода и связанных командах см. в разделе Отладка в исходном режиме. Дополнительные сведения об отладке сборок и связанных командах см. в разделе Отладка в режиме сборки.

## Комментарии

Если *параметр не* указан, отображаются ранее заданные параметры. В этом случае команды **l+** и **l-** имеют одинаковые эффекты. Однако для работы команды **l** необходимо добавить знак "плюс" (+) или "минус" (-).

При каждом запуске этой команды можно включать только один *параметр* . Если вы перечисляете несколько вариантов, обнаруживается только первый вариант. Однако, многократно выполняя эту команду, можно включить или отключить любое количество параметров. (Другими словами, **l+lst** не работает, но **l+l; l+s; l+t** обеспечивает нужный эффект.)

При указании параметра **s** исходные строки и номера строк отображаются при пошаговом выполнении кода независимо от того, указан ли параметр **l** . Параметр **o** не действует, если не указан параметр **s** .

Параметры исходной строки не вступают в силу, если только вы не включите загрузку номеров строк с помощью команды **.lines (toggle Source Line Support)** или параметра командной строки -lines. По умолчанию, если вы не использовали эти команды, WinDbg включает поддержку исходной строки, а CDB отключает ее.

# ld (символы загрузки)

Статья • 08.07.2024

Команда **ld** загружает символы для указанного модуля и обновляет все сведения о модуле.

```
dbgcmd
```

```
ld ModuleName [/f FileName]
```

## Параметры

*Имя модуля*
Указывает имя модуля, символы которого необходимо загрузить. Имя модуля может содержать различные подстановочные знаки и описатели.

**/f** *Имя файла*
Изменяет имя, выбранное для совпадения. По умолчанию имя модуля совпадает, но при **использовании /f** имя файла совпадает вместо имени модуля. *FileName* может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе подстановочных знаков и описателей см. в разделе "Синтаксис подстановочных знаков строки".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Поведение отладчика по умолчанию — использовать *отложенную загрузку* символов (также называемую отложенной загрузкой символов). Это означает, что символы фактически не загружаются до тех пор, пока они не потребуются.

С **другой стороны, команда ld** принудительно загружает все символы для указанного модуля.

## Дополнительная информация

Дополнительные сведения об отложенной (отложенной) загрузке символов см. в разделе "Отложенная загрузка символов". Дополнительные сведения о других параметрах символов см. в разделе "Параметры символов".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# lm (перечисление загруженных модулей)

Статья • 26.10.2023

Команда **lm** отображает указанные загруженные модули. Выходные данные включают состояние и путь к модулю.

```
dbgcmd
```

```
lm Options [a Address] [m Pattern | M Pattern]
```

## Параметры

*Параметры*

Любое сочетание следующих параметров:

D

Отображает выходные данные с помощью языка разметки отладчика.

o

Отображает только загруженные модули.

l

Отображает только модули, сведения о символах которых были загружены.

v

Приводит к тому, что отображение будет подробным. Отображение включает имя файла символов, имя файла изображения, сведения о контрольной сумме, сведения о версии, метки дат, метки времени и сведения о том, является ли модуль управляемым кодом (CLR). Эти сведения не отображаются, если соответствующие заголовки отсутствуют или выводятся на страницу.

u

(Только в режиме ядра) Отображает только сведения о символах пользовательского режима.

k

(Только в режиме ядра) Отображает только сведения о символах режима ядра.

й

Отображает только модули с проблемой символов. К этим символам относятся модули без символов и модули с состоянием символов C, T, #, M или Export. Дополнительные сведения об этих нотациях см. в разделе Сокращение состояния символов.

c

Отображает данные контрольной суммы.

1 мин

Уменьшает выходные данные, чтобы ничего не включались, кроме имен модулей. Этот параметр полезен, если вы используете маркер **FOREACH** для передачи выходных данных команды в входные данные другой команды.

sm

Сортирует отображение по имени модуля, а не по начальнму адресу.

Кроме того, можно включить только один из следующих вариантов. Если вы не включили ни один из этих параметров, отображается имя файла символов.

i

Отображает имя файла изображения.

f
Отображает полный путь к изображению. (Этот путь всегда соответствует пути, который отображается в уведомлении о начальной загрузке, если вы не выполнили команду **.reload -s** .) При использовании f сведения о типе символов не отображаются.

n

Отображает имя изображения. При использовании n сведения о типе символов не отображаются.

p
Отображает имя сопоставленного изображения. При использовании p сведения о типе символов не отображаются.

t

Отображает метки времени файла. При использовании t сведения о типе символов не отображаются.

*Адрес*

Указывает адрес, содержащийся в этом модуле. Отображается только модуль, содержащий этот адрес. Если Address содержит выражение, оно должно быть заключено в круглые скобки.

*Шаблон* m

Указывает шаблон, которому должно соответствовать имя модуля. Шаблон может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе этих сведений см. в разделе Синтаксис строковых подстановочных знаков.

В большинстве случаев имя модуля — это имя файла без расширения имени файла. Например, если вы хотите отобразить сведения о драйвере Flpydisk.sys, используйте команду lm mflpydisk, а не lm mflpydisk.sys. В некоторых случаях имя модуля значительно отличается от имени файла.

*Шаблон* M

Указывает шаблон, которому должен соответствовать путь к изображению. Шаблон может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе этих сведений см. в разделе Синтаксис строковых подстановочных знаков.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

Команда **lm** выводит список всех модулей и состояние символов для каждого модуля.

Windows поддерживает выгруженный список модулей для процессов в пользовательском режиме. При отладке процесса или файла дампа пользовательского режима команда **lm** также отображает эти выгруженные модули.

Отображаемые модули зависят от того, как выполняется отладка, например в пользовательском режиме или режиме ядра, а также от контекста, на который вы смотрите. Дополнительные сведения о контексте процесса и других параметрах контекста см. в разделах Изменение контекстов и Управление процессами и потоками.

Эта команда показывает несколько столбцов или полей, каждое из которых имеет свой заголовок. Некоторые из этих названий имеют определенное значение:

- *Имя модуля* обычно является именем файла без расширения имени файла. В некоторых случаях имя модуля значительно отличается от имени файла.
- Тип символа сразу же следует за именем модуля. Этот столбец не помечен. Дополнительные сведения о различных значениях состояния см. в разделе Сокращение состояния символов. Если вы загрузили символы, имя файла символов следует за этим столбцом.
- Первый адрес в модуле отображается как начальный. Первый адрес после окончания модуля отображается как конец. Например, если start имеет значение "faab4000", а end — "faab8000", модуль расширяется от 0xFAAB4000 до 0xFAAB7FFF включительно.
- Только **lmv**: в столбце путь к изображению отображается имя исполняемого файла, включая расширение имени файла. Как правило, полный путь включается в пользовательском режиме, но не в режиме ядра.
- Только **lmv**: значение загруженного файла изображения символов совпадает с именем изображения, если только не присутствуют символы Microsoft CodeView.
- Только **lmv**: значение файла сопоставленного образа памяти обычно не используется. Если отладчик сопоставляет файл изображения (например, во время отладки минидампа), это значение является именем сопоставленного образа.

В следующем примере кода показана команда **lm** с использованием параметров m и s*, поэтому отображаются только модули, начинающиеся с "s".

```
dbgcmd
```

```
kd> lm m s*
start    end        module name
f9f73000 f9f7fd80   sysaudio     (deferred)
fa04b000 fa09b400   srv          (deferred)
faab7000 faac8500   sr           (deferred)
facac000 facbae00   serial       (deferred)
fb008000 fb00ba80   serenum      e:\mysymbols\SereEnum.pdb\.......
fb24f000 fb250000   swenum       (deferred)

Unloaded modules:
f9f53000 f9f61000   swmidi.sys
fb0ae000 fb0b0000   splitter.sys
fb040000 fb043000   Sfloppy.SYS
```

# Примеры

В следующих двух примерах команда **lm** показана один раз без каких-либо параметров и один раз с параметром sm. Сравните порядок сортировки в двух примерах.

Пример 1:

dbgcmd

```
0:000> lm
start    end        module name
01000000 0100d000   stst       (deferred)
77c10000 77c68000   msvcrt     (deferred)
77dd0000 77e6b000   ADVAPI32   (deferred)
77e70000 77f01000   RPCRT4     (deferred)
7c800000 7c8f4000   kernel32   (deferred)
7c900000 7c9b0000   ntdll      (private pdb symbols) c:\db20sym\ntdll.pdb
```

Пример 2.

dbgcmd

```
0:000> lm sm
start    end        module name
77dd0000 77e6b000   ADVAPI32   (deferred)
7c800000 7c8f4000   kernel32   (deferred)
77c10000 77c68000   msvcrt     (deferred)
7c900000 7c9b0000   ntdll      (private pdb symbols)  c:\db20sym\ntdll.pdb
77e70000 77f01000   RPCRT4     (deferred)
01000000 0100d000   stst       (deferred)
```

# См. также раздел

- Изменение контекстов
- Управление процессами и потоками.
- Стандартные методы отладки
- Начало работы с отладкой Windows

# ln (список ближайших символов)

Статья • 08.07.2024

Команда **ln** отображает символы по указанному адресу или рядом с ним.

```
dbgcmd
```

```
ln Address
ln /D Address
```

## Параметры

*Адрес*
Указывает адрес, в котором отладчик должен начать поиск символов. Отображаются ближайшие символы до или после *адреса*. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

**/D**
Указывает, что выходные данные отображаются с помощью языка разметки отладчика (DML). Выходные данные DML включают ссылку, которую можно использовать для изучения модуля, содержащего ближайший символ. Она также содержит ссылку, которую можно использовать для задания точки останова.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

С помощью **команды ln** можно определить, на что указывает указатель. Эта команда также может быть полезной при просмотре поврежденного стека, чтобы определить, какая процедура сделала вызов.

Если доступны сведения о исходной строке, отображается также имя исходного файла и сведения о номере строки.

При использовании исходного сервера **команда ln** отображает сведения, связанные с исходным сервером.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ls, lsa (список исходных строк)

Статья • 08.07.2024

Команды **ls и lsa** отображают ряд строк из текущего исходного файла и перенастроите текущий номер исходной строки.

dbgcmd

```
ls [.] [first] [, count]
lsa [.] address [, first [, count]]
```

## Параметры

.
Вызывает поиск исходного файла, используемого обработчиком отладчика или командой **srcpath (Set Source Path).** Если период (**.**) **не включен, ls использует файл, который был недавно загружен командой lsf (load Source File).**

*адрес*
Указывает адрес, в котором начинается отображение источника.

Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*первый*
Указывает первую строку для отображения. Значение по умолчанию — текущая строка.

*считать*
Указывает количество отображаемых строк. Значение по умолчанию — 20 (0x14), если вы не изменили значение по умолчанию с помощью **команды lsp -a** .

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |

| Позиция | Description |
| --- | --- |
| Платформы | Все |

## Замечания

После выполнения **команды ls или lsa** текущая строка переопределена как окончательная строка, отображаемая плюс одна. Текущая строка используется в будущих **командах lsa, lsa** и lsc.

## См. также

lsc (список текущего источника)

lsf, lsf- (загрузка или выгрузка исходного файла)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# lsc (перечисление текущего источника)

Статья • 26.10.2023

Команда **lsc** отображает текущее имя исходного файла и номер строки.

dbgcmd

```
lsc
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## См. также

**ls, lsa (перечисление строк источника)**

**lsf, lsf- (загрузка или выгрузка исходного файла)**

# lse (запуск редактора исходного кода)

Статья • 26.10.2023

Команда **lse** открывает редактор для текущего исходного файла.

```
dbgcmd
```

```
lse
```

## Среда

| | |
|---|---|
| **Режимы** | Пользовательский режим, режим ядра |
| **Целевые объекты** | Динамический аварийный дамп |
| **Платформы** | Все |

# Комментарии

Команда **lse** открывает редактор для текущего исходного файла. Эта команда эквивалентна нажатию кнопки **Изменить этот файл** в контекстном меню окна Источник в WinDbg.

Редактор открывается на компьютере, на котором запущен целевой объект, поэтому вы не можете использовать команду **lse** из удаленного клиента.

Сведения о реестре редактора WinDiff или значение переменной среды WINDBG_INVOKE_EDITOR определяют, какой редактор открывается. Например, рассмотрим следующее значение WINDBG_INVOKE_EDITOR.

```
reg
```

```
c:\my\path\myeditor.exe -file %f -line %l
```

Это значение указывает, что Myeditor.exe открывается с одним номером строки текущего исходного файла. Параметр **%l** указывает, что номера строк должны считываться как однонационные, а **%f** указывает, что следует использовать текущий исходный файл. Можно также включить **%L** , чтобы указать, что номера строк отсчитываются от нуля, или **%p** , чтобы указать, что следует использовать текущий исходный файл.

# lsf, lsf- (загрузка или выгрузка исходного файла)

Статья • 08.07.2024

Команды **lsf и lsf-** загружают или выгружают исходный файл.

```
dbgcmd
```

```
lsf Filename
lsf- Filename
```

## Параметры

*Имя файла*
Указывает файл для загрузки или выгрузки. Если этот файл не расположен в каталоге, из которого был открыт отладчик, необходимо включить абсолютный или относительный путь. Имя файла должно соответствовать соглашениям о имени файла Microsoft Windows.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Команда **lsf** загружает исходный файл.

Команда **lsf-** выгрузит исходный файл. Эту команду можно использовать для выгрузки файлов, загруженных ранее с **lsf** или автоматически загруженных исходных файлов. Вы не можете использовать **lsf-** для выгрузки файлов, загруженных с помощью файла WinDbg | **Команда файла с открытым кодом** или файлы, загруженные рабочей областью WinDbg.

В CDB или KD можно просмотреть исходные файлы в окне команды отладчика. В WinDbg исходные файлы загружаются в качестве новых окон источника.

Дополнительные сведения о исходных файлах, путях к источнику и других способах загрузки исходных файлов см. в разделе "Путь к источнику".

## См. также

**ls, lsa (список исходных строк)**

**lsc (список текущего источника)**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# lsp (число исходных строк)

Статья • 08.07.2024

Команда **lsp** управляет количеством исходных строк при выполнении или выполнении кода или использовании **команд** lsa.

dbgcmd

```
lsp [-a] LeadingLines TrailingLines
lsp [-a] TotalLines
lsp [-a]
```

## Параметры

**-a**
Задает или отображает количество строк, которые **отображаются ls** и **lsa** . Если опущено **значение -a**, **lsp** задает или отображает количество строк, отображаемых при выполнении и выполнении кода.

*Ведущие линии*
Указывает количество строк, отображаемых до текущей строки.

*Конечные линии*
Указывает количество строк, отображаемых после текущей строки.

*TotalLines*
Указывает общее количество строк для отображения. Это число распределяется равномерно между начальными и конечными линиями. (Если это число нечетно, отображаются более конечные строки.)

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

При использовании **команды lsp вместе с параметрами lsp** отображаются текущие начальные и конечные значения строк, которые использовались при пошаговом выполнении. При использовании этой команды вместе с параметром **-a** lsp **отображаются значения,** используемые при пошаговом выполнении, а также для **команд** ls и lsa.

При пошаговом переходе по программе или разрыве после выполнения программы предыдущая **команда lsp** определяет количество отображаемых начальных и конечных строк. При использовании **lsa** предыдущая **команда lsp** определяет количество отображаемых начальных и конечных строк. При использовании **ls** все строки отображаются как один блок, поэтому предыдущая **команда lsp** определяет общее количество отображаемых строк.

# Дополнительная информация

Дополнительные сведения об отладке источников и связанных командах см. в разделе "Отладка в режиме источника".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# m (перемещение памяти)

Статья • 08.07.2024

Команда **m** копирует содержимое памяти из одного расположения в другое.

Не путайте эту команду с командой **~m (Resume Thread).**

```
dbgcmd
```

```
m Range Address
```

## Параметры

*Диапазон*
Указывает область памяти для копирования. Дополнительные сведения о синтаксисе этого параметра см. в разделе "Синтаксис адреса и диапазона адресов".

*Адрес*
Задает начальный адрес целевой области памяти. Дополнительные сведения о синтаксисе этого параметра см. в разделе "Синтаксис адреса и диапазона адресов".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о манипуляции с памятью и описании других команд, связанных с памятью, см. в разделе "Чтение и запись памяти".

## Замечания

Область памяти, указываемая адресом, может быть частью области памяти, *указанной диапазоном*. Перекрывающиеся перемещения обрабатываются правильно.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# n (установка базы номеров)

Статья • 08.07.2024

Команда **n** задает для базы номеров по умолчанию (radix) указанное значение или отображает текущую базу чисел.

Не путайте эту команду с командой **~n (Приостановка потока).**

dbgcmd

```
n [Radix]
```

## Параметры

*Корень*
Указывает базу номеров по умолчанию, которая используется для числового отображения и записи. Можно использовать одно из следующих значений.

Развернуть таблицу

| значение | Описание |
|---|---|
| 8 | Восьмеричное |
| 10 | Десятичное число |
| 16 | Шестнадцатеричное |

Если опустить *Radix*, отображается текущая база номеров по умолчанию.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Текущий радикс влияет на входные и выходные данные выражений MASM. Это не влияет на входные или выходные данные выражений C++. Дополнительные сведения об этих выражениях см. в разделе "Оценка выражений".

По умолчанию радикс имеет значение 16 при запуске отладчика.

Во всех выражениях MASM числовые значения интерпретируются как числа в текущем радиксе (16, 10 или 8). Можно переопределить радикс по умолчанию, указав **префикс 0x** (шестнадцатеричный), **префикс 0n** (десятичный **), префикс 0t** (восьмерично) или **0y** префикс (двоичный).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ob, ow, od (выходные данные в порт)

Статья • 08.07.2024

Команды **ob**, **ow** и **od** отправляют байт, слово или двойное слово в выбранный порт.

dbgcmd

```
ob Address Value
ow Address Value
od Address Value
```

## Параметры

*Адрес*
Указывает адрес порта.

*Value*
Указывает шестнадцатеричное значение для записи в порт.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Только на основе x86 |

## Замечания

Команда **ob** записывает один байт, **команда ow** записывает слово, а **команда od** записывает двойное слово.

Убедитесь, что вы не отправляете слово или двойное слово в порт, который не поддерживает этот размер.

## См. также

ib, id, iw (входные данные из порта)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# p (шаг)

Статья • 05.04.2024

Команда **p** выполняет одну инструкцию или исходную строку и при необходимости отображает полученные значения всех регистров и флагов. При возникновении вызовов или прерываний вложенных вызовов они рассматриваются как один шаг.

Режим пользователя

```
dbgcmd
```

```
[~Thread] p[r] [= StartAddress] [Count] ["Command"]
```

Режим ядра

```
dbgcmd
```

```
p[r] [= StartAddress] [Count] ["Command"]
```

# Параметры

*Поток*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд pr**, **tr** или .prompt_allow -reg. Все три из этих команд управляют одинаковым параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других трех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором должно начинаться выполнение. Если вы не используете *StartAddress*, выполнение начинается с инструкции, на которую

указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Численность*
Указывает количество инструкций или исходных строк, которые необходимо выполнить перед остановкой. Каждый шаг отображается как отдельное действие в окне команды отладчика. Значение по умолчанию равно единице.

*Command*
Указывает команду отладчика, выполняемую после выполнения шага. Эта команда выполняется до отображения стандартных **результатов p** . Если вы также используете *count*, указанная команда выполняется после завершения всех шагов (но до отображения результатов последнего шага).

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о выпуске **команды p** и обзоре связанных команд см. в разделе "Управление целевым объектом".

# Замечания

При указании *count* каждая инструкция отображается так, как она выполняется.

Если отладчик сталкивается **с инструкцией вызова** или прерыванием при пошаговом выполнении, вызываемая подпрограмма будет выполняться полностью, если точка останова не обнаружена. Элемент управления возвращается отладчику по следующей инструкции после вызова или прерывания.

Каждый шаг выполняет одну инструкцию сборки или одну исходную строку в зависимости от того, находится ли отладчик в режиме сборки или в режиме

источника. **Используйте команды l+t и l-t** или кнопки на панели инструментов WinDbg, чтобы переключаться между этими режимами.

При быстром выполнении шагов во многих случаях в WinDbg окна сведений об отладке обновляются после каждого шага. Если это обновление приводит к замедлению времени отклика, используйте **.suspend_ui (приостановка интерфейса WinDbg)** для временной приостановки обновления этих окон.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# pa (шаг к адресу)

Статья • 08.07.2024

Команда **pa** выполняет программу до достижения указанного адреса, отображая каждый шаг.

Режим пользователя

```
dbgcmd
```

```
[~Thread] pa [r] [= StartAddress] StopAddress ["Command"]
```

Режим ядра

```
dbgcmd
```

```
pa [r] [= StartAddress] StopAddress ["Command"]
```

## Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд par**, pr**,** **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других трех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. В противном случае отладчик начинается с инструкции, на которую указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*StopAddress*
Указывает адрес, в котором будет останавливаться выполнение. Этот адрес должен соответствовать точному адресу инструкции.

*Команда*
Указывает команду отладчика, выполняемую после выполнения шага. Эта команда выполняется до отображения стандартных **результатов ра** . Если вы также используете *StopAddress, указанная команда выполняется после достижения StopAddress* (но до отображения результатов последнего шага).

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **ра** приводит к началу выполнения целевого объекта. Это выполнение продолжается до достижения указанной инструкции или точки останова.

**Обратите внимание** , что если эта команда используется в режиме ядра, выполнение останавливается при обнаружении инструкции по указанному виртуальному адресу в любом виртуальном адресном пространстве.

Во время этого выполнения все шаги отображаются явным образом. Называемые функции рассматриваются как единая единица. Таким образом, отображение этой команды похоже на то, что отображается при повторном выполнении **p (шаг)** до тех пор, пока счетчик программы не достигнет указанного адреса.

Например, следующая команда явным образом выполняет шаги по целевому коду до достижения возвращаемого адреса текущей функции.

```
dbgcmd
```

```
0:000> pa @$ra
```

В следующем примере показано использование **команды pa** вместе с **командой kb** для отображения трассировки стека:

```
dbgcmd
```

```
0:000> pa 70b5d2f1 "kb"
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# pc (шаг к следующему вызову)

Статья • 08.07.2024

Команда **пк** выполняет программу, пока не будет достигнута инструкция вызова.

Режим пользователя

```
dbgcmd
```

```
[~Thread] pc [r] [= StartAddress] [Count]
```

Режим ядра

```
dbgcmd
```

```
pc [r] [= StartAddress] [Count]
```

# Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд pcr**, pr, **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других трех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. В противном случае отладчик начинается с инструкции, на которую указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает количество инструкций по вызову**, с которым отладчик должен столкнуться для остановки** этой команды. Значение по умолчанию равно единице.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **компьютера** приводит к началу выполнения целевого объекта. Это выполнение продолжается до тех пор, **пока не будет достигнута инструкция вызова** или не будет обнаружена точка останова.

Если счетчик программы уже находится в инструкции **вызова** , выполняется весь вызов. После возвращения этого вызова выполнение продолжается до достижения другого **вызова** . Это выполнение, а не трассировка вызова является единственной разницей между **pc** и **tc (трассировка до следующего вызова).**

В исходном режиме можно связать одну исходную строку с несколькими инструкциями по сборке. Команда **пк** не останавливается в инструкции вызова**, связанной** с текущей исходной строкой.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# pct (шаг к следующему вызову или возврату)

Статья • 08.07.2024

Команда **pct** выполняет программу до тех пор, пока она не достигнет инструкции вызова или возвращаемой инструкции.

Режим пользователя

```
dbgcmd
```

```
[~Thread] pct [r] [= StartAddress] [Count]
```

Режим ядра

```
dbgcmd
```

```
pct [r] [= StartAddress] [Count]
```

## Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистра **с помощью команд pctr**, pr, **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других трех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. В противном случае отладчик начинается с инструкции, на которую указывает указатель

инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает количество вызовов **или инструкций возврата, которые должны быть обнаружены для остановки** этой команды. Значение по умолчанию равно единице.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **pct** приводит к началу выполнения целевого объекта. Это выполнение продолжается до тех пор, пока не будет достигнута инструкция вызова или возврата **или** не будет обнаружена точка останова.

Если счетчик программы уже находится на вызове **или возврате инструкции, выполняется весь вызов или** возврат. После возвращения этого вызова или возврата выполнение продолжается до тех пор, пока не будет достигнут другой **вызов** или **возврат** . Это выполнение, а не трассировка вызова является единственной разницей между **pct** и **tct (трассировка до следующего вызова или возврата)**.

В исходном режиме можно связать одну исходную строку с несколькими инструкциями по сборке. Команда **pct** не останавливается при вызове **или возврате инструкции, связанной** с текущей исходной строкой.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ph (шаг к следующей ветвления инструкции)

Статья • 08.07.2024

Команда **ph** выполняет программу до тех пор, пока не будет достигнута любая инструкция ветвления, включая условные или безусловные ветви, вызовы, возвраты и системные вызовы.

Режим пользователя

```
dbgcmd
```

```
[~Thread] ph [r] [= StartAddress] [Count]
```

Режим ядра

```
dbgcmd
```

```
ph [r] [= StartAddress] [Count]
```

## Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд phr**, pr, **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других трех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. В противном

случае отладчик начинается с инструкции, на которую указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает количество ветвей инструкций, которые должны быть обнаружены для остановки этой команды. Значение по умолчанию равно единице.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **ph** приводит к началу выполнения целевого объекта. Это выполнение продолжается до достижения инструкции ветвления или точки останова.

Если счетчик программы уже находится в инструкции ветвления, выполняется ветвление всей инструкции. После возврата этой инструкции ветвления выполнение продолжается до тех пор, пока не будет достигнута другая инструкция ветвления. Это выполнение, а не трассировка вызова является единственной разницей между **ph** и **th (трассировка до следующей ветви инструкции)**.

В исходном режиме можно связать одну исходную строку с несколькими инструкциями по сборке. Команда **ph** не останавливается в инструкции ветвления, связанной с текущей исходной строкой.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# pt (шаг к следующему возврату)

Статья • 08.07.2024

Команда **pt** выполняет программу до достижения инструкции возврата.

Режим пользователя

```
dbgcmd
```

```
[~Thread] pt [r] [= StartAddress] [Count] ["Command"]
```

Режим ядра

```
dbgcmd
```

```
pt [r] [= StartAddress] [Count] ["Command"]
```

# Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд ptr**, pr, **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других трех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. В противном случае отладчик начинается с инструкции, на которую указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает количество инструкций возврата, **которые должны быть обнаружены для остановки** этой команды. Значение по умолчанию равно единице.

*Команда*
Указывает команду отладчика, выполняемую после выполнения шага. Эта команда выполняется до отображения стандартных **результатов pt** . Если вы также используете *count*, указанная команда выполняется после завершения всех шагов (но до отображения результатов последнего шага).

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **pt** приводит к началу выполнения целевого объекта. Это выполнение продолжается до тех пор, пока **не будет достигнута инструкция возврата** или не будет обнаружена точка останова.

Если счетчик программы уже находится в инструкции **возврата** , выполняется весь возврат. После возврата этого возврата выполнение продолжается до тех пор, пока не будет достигнут другой **возврат** . Это выполнение, а не трассировка вызова является единственной разницей между **pt** и **tt (трассировка до следующего возврата)**.

В исходном режиме можно связать одну исходную строку с несколькими инструкциями по сборке. Команда **pt** не останавливается на инструкции возврата, **связанной с текущей** исходной строкой.

В следующем примере показано использование **команды pt** вместе с **командой kb** для отображения трассировки стека:

```
dbgcmd
```

```
0:000> pt "kb"
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# q, qq (Quit)

Статья • 26.10.2023

Команды **q** и **qq** завершают сеанс отладки. (В CDB и KD эта команда также завершает сам отладчик. В WinDbg эта команда возвращает отладчик в неактивный режим.)

```
dbgcmd
```

```
q
qq
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

При отладке в пользовательском режиме команда **q** завершает сеанс отладки и закрывает целевое приложение.

При отладке в режиме ядра команда **q** сохраняет файл журнала и завершает сеанс отладки. Целевой компьютер остается заблокированным.

Если эта команда не работает в KD, нажмите **клавиши CTRL+R+ВВОД** на клавиатуре отладчика, а затем повторите команду **q** . Если это действие не работает, нажмите клавиши CTRL+B+ВВОД, чтобы выйти из отладчика.

Команда **qq** ведет себя точно так же, как и команда **q** , если вы не выполняете удаленную отладку. Во время удаленной отладки команда **q** не оказывает никакого влияния, но команда **qq** завершает сервер отладки. Дополнительные сведения об этом эффекте см. в разделе Удаленная отладка с помощью отладчика.

# qd (выход и отключение)

Статья • 26.10.2023

Команда **qd** завершает сеанс отладки и оставляет все целевые приложения в пользовательском режиме запущенными. (В CDB и KD эта команда также завершает работу отладчика. В WinDbg эта команда возвращает отладчик в неактивный режим.)

```
dbgcmd
```

```
qd
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Только пользовательский режим |
| Цели | Только динамическая отладка |
| Платформы | все |

# Комментарии

Команда **qd** отсоединяется от целевого приложения и завершает сеанс отладки, оставляя целевой объект по-прежнему запущенным. Однако эта команда поддерживается только в Microsoft Windows XP и более поздних версиях Windows. В Windows 2000 **qd** создает предупреждающее сообщение и не оказывает никакого влияния.

При выполнении удаленной отладки с помощью отладчика нельзя использовать команду **qd** клиента отладки.

# r (Registers)

Статья • 08.07.2024

Команда **r** отображает или изменяет регистры, регистры с плавающей запятой, флаги, псевдорегистры и псевдонимы с фиксированным именем.

Режим пользователя

```
dbgcmd
```

```
[~Thread] r[M Mask|F|X|?] [ Register[:[Num]Type] [= [Value]] ]
r.
```

Режим ядра

```
dbgcmd
```

```
[Processor] r[M Mask|F|X|Y|YI|?] [ Register[:[Num]Type] [= [Value]] ]
r.
```

## Параметры

*Процессор*
Указывает процессор, из которых считываются регистры. Значение по умолчанию равно нулю. Если указать *обработчик*, вы не можете включить *параметр Register* - все регистры отображаются. Дополнительные сведения о синтаксисе см. в разделе "Многопроцессорный синтаксис". Вы можете указать процессоры только в режиме ядра.

*Нить*
Указывает поток, из которых считываются регистры. Если поток не указан, используется текущий поток. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**M** *Маска*
Указывает маску, используемую при отображении регистров отладчиком. Буква "M" должна быть прописной буквой. *Маска* — это сумма битов, указывающих что-то о отображении регистра. Значение битов зависит от процессора и режима (дополнительные сведения см. в таблицах в следующем разделе примечаний). Если не указано **M**, используется маска по умолчанию. Вы можете задать или отобразить маску по умолчанию с помощью **команды Rm (Register Mask).**

**F**
Отображает регистры с плавающей запятой. Буква "F" должна быть прописной буквой. Этот параметр эквивалентен **0x4** M.

**X**
Отображает регистры XMM SSE. Этот параметр эквивалентен **0x40** M.

**Y**
Отображает регистры AVX YMM. Этот параметр эквивалентен **0x200** M.

**YI**
Отображает целые числа AVX YMM. Этот параметр эквивалентен **0x400** M.

**Z**
Отображает регистры AVX-512 YMM (zmm0-zmm31) в формате с плавающей запятой.

**ZI**
Отображает регистры AVX-512 YMM (zmm0-zmm31) в целочисленном формате.

**K**
Отображение регистров предиката AVX-512 Opmask (K0-K7).

**?**
(Только псевдорегистрируемое назначение) Вызывает псевдорегистрированную информацию. Разрешен любой тип. Дополнительные сведения о синтаксисе **r?** см. в примерах программы команд отладчика.

*Регистрировать*
Задает псевдоним регистра, флага, псевдорегистрирующего или фиксированного имени для отображения или изменения. Этот параметр не должен предшествовать знаку (@). Дополнительные сведения о синтаксисе см. в разделе "Регистр синтаксис".

*Число*
Указывает количество отображаемых элементов. Если этот параметр не указан , *но включен тип*, отображается полная длина регистра.

*Тип*
Указывает формат данных для отображения каждого элемента регистрации в. Тип можно использовать только с 64-разрядными и 128-разрядными регистрами векторов. Можно указать несколько типов.

Можно указать одно или несколько следующих значений.

Развернуть таблицу

| *Тип* | **Формат отображения** |
|---|---|
| **ib** | Подписанный байт |
| **ub** | Неподписанный байт |
| **iw** | Подписанное слово |
| **uw** | Слово без знака |
| **id** | Подписанный DWORD |
| **ud** | Неподписанный DWORD |
| **iq** | Подписанный четыре слова |
| **uq** | Unsigned quad-word |
| **f** | 32-разрядная плавающая точка |
| **дн.** | 64-разрядная плавающая точка |

*Value*
Указывает значение, которое нужно назначить регистру. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис числовых выражений".

.
Отображает регистры, используемые в текущей инструкции. Если регистры не используются, выходные данные не отображаются.

# Среда

Развернуть таблицу

| **Позиция** | **Description** |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения о контексте регистрации и других параметрах контекста см. в разделе "Изменение контекстов".

# Замечания

Если вы не указываете *register*, **команда r** отображает все регистры, отличные от плавающей запятой, и **команда rF** отображает все регистры с плавающей запятой. Это поведение можно изменить с помощью **команды rm (Register Mask).**

Если указать register, *но не указан* знак равенства (=) и *параметр Value*, команда отображает текущее значение регистра.

Если указать *регистр* и знак равенства (=), но не указано *значение*, команда отображает текущее значение регистра и запрашивает новое значение.

Если указать *register*, знак равенства (=) и *Значение*, команда изменяет регистр на значение. (Если *тихий режим* активен, вы можете опустить знак равенства. Вы можете включить тихий **режим с помощью команды sq (Set Quiet Mode).** В режиме ядра можно также включить спокойный режим с помощью переменной среды KDQUIET.)

Можно указать несколько регистров, разделенных запятыми.

В пользовательском режиме **команда r** отображает регистры, связанные с текущим потоком. Дополнительные сведения о потоках см. в разделе "Управление процессами и потоками".

В режиме **ядра команда r** отображает регистры, связанные с текущим *контекстом* регистра. Контекст регистрации можно задать для сопоставления определенного потока, записи контекста или кадра ловушки. Отображаются только наиболее важные регистры для указанного контекста регистра, и их значения нельзя изменить. Дополнительные сведения о контексте регистрации см. в разделе "Регистрация контекста".

При указании регистра с плавающей запятой по имени **параметр F** не требуется. При указании одного регистра с плавающей запятой в дополнение к десятичному значению отображается необработанное шестнадцатеричное значение.

Следующие *биты Маски* поддерживаются для процессора на основе x86 или процессора на основе x64.

Развернуть таблицу

| бит | значение | Описание |
|---|---|---|
| 0 1 | 0x1 0x2 | Отображает базовые целые регистры. (Установка одного или обоих этих битов имеет одинаковый эффект.) |
| 2 | 0x4 | Отображает регистры с плавающей запятой. |
| 3 | 0x8 | Отображает регистры сегментов. |
| 4 | 0x10 | Отображает регистры MMX. |
| 5 | 0x20 | Отображает регистры отладки. В режиме ядра этот бит также отображает регистр CR4. |
| 6 | 0x40 | Отображает регистры XMM SSE. |
| 7 | 0x80 | (только режим ядра) Отображает регистры элементов управления, например CR0, CR2, CR3 и CR8. |
| 8 | 0x100 | (только режим ядра) Отображает дескриптор и регистры состояния задачи. |
| 9 | 0x200 | Отображает регистры AVX YMM в плавающей запятой. |
| 10 | 0x400 | Отображает регистры AVX YMM в десятичных целочисленных числах. |
| 11 | 0x800 | Отображает регистры AVX XMM в десятичных целочисленных числах. |

В следующих примерах кода показаны **команды r** для процессора на основе x86.

В режиме ядра в следующей команде показаны регистры процессора 2.

```
dbgcmd
```

```
1: kd> 2r
```

В пользовательском режиме следующая команда отображает регистры для потока 2.

dbgcmd

```
0:000> ~2 r
```

В пользовательском режиме следующая команда отображает все **регистры eax** , связанные со всеми потоками (в порядке индекса потока).

dbgcmd

```
0:000> ~* r eax
```

Следующая команда задает **регистр eax** для текущего потока 0x000000FF.

dbgcmd

```
0:000> r eax=0x000000FF
```

Следующая команда задает **для st0** регистр 1.234e+10 ( **F** является необязательным).

dbgcmd

```
0:000> rF st0=1.234e+10
```

Следующая команда отображает нулевой флаг.

dbgcmd

```
0:000> r zf
```

Следующая команда отображает **регистр xmm0** в виде 16 незначенных байтов, а затем отображает полное содержимое **регистра xmm1** в формате с плавающей запятой двойной точности.

dbgcmd

```
0:000> r xmm0:16ub, xmm1:d
```

Если текущий синтаксис — C++, необходимо предшествуть регистрации по знаку (@). Поэтому можно использовать следующую команду, чтобы скопировать **регистр ebx** в **регистр eax** .

```
dbgcmd

0:000> r eax = @ebx
```

Следующая команда отображает псевдорегистры таким же образом, как **команда r** отображает регистры.

```
dbgcmd

0:000> r $teb
```

Вы также можете использовать **команду r** для создания *псевдонимов* фиксированного имени. Эти псевдонимы не регистрируются или псевдорегистрируются, даже если они связаны с командой **r** . Дополнительные сведения об этих псевдонимах см. в разделе "Использование псевдонимов".

Ниже приведен пример **команды r.** на процессоре на основе x86. Последняя запись стека вызовов предшествует самой команде.

```
dbgcmd

01004af3 8bec            mov     ebp,esp
0:000> r.
ebp=0006ffc0  esp=0006ff7c
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# rdmsr (чтение MSR)

Статья • 08.07.2024

Команда **rdmsr** считывает значение регистра для конкретной модели (MSR) из указанного адреса.

```
dbgcmd
```

```
rdmsr Address
```

## Параметры

*Адрес*
Указывает адрес MSR.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Все |

## Замечания

Команда **rdmsr может отображать MSR** на платформах на основе x86 и x64. Определения MSR зависят от платформы.

## См. также

**wrmsr (запись MSR)**

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# rm (регистрация маски)

Статья • 08.07.2024

Команда **rm** изменяет или отображает маску отображения регистра. Эта маска определяет, как регистры отображаются командой **r (Registers).**

dbgcmd

```
rm
rm ?
rm Mask
```

## Параметры

**?**
Отображает список возможных *битов маски* .

*Маска*
Указывает маску, используемую при отображении регистров отладчиком. *Маска* — это сумма битов, указывающих что-то о отображении регистра. Значение битов зависит от процессора и режима. Дополнительные сведения; см. таблицы в следующем разделе "Примечания".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Буква "m" в имени команды должна быть строчной буквой.

Если вы используете **rm** без параметров, отображается текущее значение, а также объяснение его битов.

Чтобы отобразить базовые целые регистры, необходимо задать бит 0 (0x1) или бит 1 (0x2). По умолчанию 0x1 заданы для 32-разрядных целевых объектов и 0x2 заданы для 64-разрядных целевых объектов. Эти два бита нельзя задать одновременно, если вы пытаетесь задать оба бита, 0x2 переопределяет 0x1.

Маску по умолчанию можно переопределить с помощью **команды r (Registers)** вместе с параметром **M**.

Следующие *биты Маски* поддерживаются для процессора на основе x86 или процессора на основе x64.

Развернуть таблицу

| бит | значение | Описание |
|---|---|---|
| 0 1 | 0x1 0x2 | Отображает базовые целые регистры. (Установка одного или обоих этих битов имеет одинаковый эффект.) |
| 2 | 0x4 | Отображает регистры с плавающей запятой. |
| 3 | 0x8 | Отображает регистры сегментов. |
| 4 | 0x10 | Отображает регистры MMX. |
| 5 | 0x20 | Отображает регистры отладки. В режиме ядра этот бит также отображает регистр CR4. |
| 6 | 0x40 | Отображает регистры XMM SSE. |
| 7 | 0x80 | (только режим ядра) Отображает регистры элементов управления, например CR0, CR2, CR3 и CR8. |
| 8 | 0x100 | (только режим ядра) Отображает дескриптор и регистры состояния задачи. |
| 9 | 0x200 | Отображает регистры AVX YMM в плавающей запятой. |

| бит | значение | Описание |
| --- | --- | --- |
| 10 | 0x400 | Отображает регистры AVX YMM в десятичных целочисленных числах. |
| 11 | 0x800 | Отображает регистры AVX XMM в десятичных целочисленных числах. |
| 12 | 0x1000 | Отображает регистры AVX-512 zmm0-zmm31 в формате с плавающей запятой. |
| 13 | 0x2000 | Отображает регистры AVX-512 zm00-zmm31 в целочисленном формате. |
| 14 | 0x4000 | Отображает регистры AVX-512 k0-k7. |

# Примеры

Включите целочисленное состояние и регистры сегментов.

dbgcmd

```
0: kd> rm 0x00a
0: kd> rm
Register output mask is a:
       2 - Integer state (64-bit)
       8 - Segment registers
```

Включите 0x1000 (отображает регистры AVX-512 zmm0-zmm31 в формате с плавающей запятой).

dbgcmd

```
0: kd> rm 0x100a
0: kd> rm
Register output mask is 100a:
       2 - Integer state (64-bit)
       8 - Segment registers
    1000 - AVX-512 ZMM registers
```

Включите маску 0x2000 (отображает регистры AVX-512 zmm00-zmm31 в целочисленном формате).

dbgcmd

```
0: kd> rm 0x200a
0: kd> rm
Register output mask is 200a:
       2 - Integer state (64-bit)
       8 - Segment registers
    2000 - AVX-512 ZMM Integer registers
```

Включите все маски регистрации AVX-512:

dbgcmd

```
0: kd> rm 0x700a
0: kd> rm
Register output mask is 700a:
       2 - Integer state (64-bit)
       8 - Segment registers
    1000 - AVX-512 ZMM registers
    2000 - AVX-512 ZMM Integer registers
    4000 - AVX-512 Opmask registers
```

Если вы попытаетесь установить маску регистрации на оборудовании, которое не поддерживает его, недопустимые биты маски регистра будут игнорироваться.

dbgcmd

```
kd> rm 0x100a
Ignored invalid bits 1000
kd> rm
Register output mask is a:
      2 - Integer state (64-bit)
       8 - Segment registers
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# s (память поиска)

Статья • 26.10.2023

Команда **s** выполняет поиск по памяти, чтобы найти определенный шаблон байтов.

Не путайте эту команду с **~s (изменение текущего процессора),~s (задать текущий поток),|s (задать текущий процесс)** или **||команды s (задать текущую систему).**

dbgcmd

```
s [-[[Flags]Type]] Range Pattern
s -[[Flags]]v Range Object
s -[[Flags]]sa Range
s -[[Flags]]su Range
```

# Параметры

[*Флаги*]
Задает один или несколько параметров поиска. Каждый флаг представляет собой одну букву. Флаги необходимо заключить в один набор скобок ([]). Нельзя добавлять пробелы между квадратными скобками, кроме как между **n** или **l** и его аргументом. Например, если вы хотите указать параметры **s** и **w** , используйте команду s -[sw]Type Range Pattern.

Можно указать один или несколько следующих флагов:

**s**
Сохраняет все результаты текущего поиска. Эти результаты можно использовать для повторения поиска позже.

**R**
Ограничивает текущий поиск результатами из последнего сохраненного поиска. Нельзя использовать флаги **s** и **r** в одной команде. При использовании **r** значение *Range* игнорируется, а отладчик выполняет поиск только тех попаданий, которые были сохранены **предыдущей командой** .

**n***Хиты*
Указывает количество обращений, сохраняемых при использовании **флага s** . Значение по умолчанию — 1024 попадания. Если вы используете **n** вместе с другими флагами, **n** должен быть последним флагом, за которым следует аргумент *Hits* . Интервал между **n** и *hits* является необязательным, но вы не можете добавить другие пробелы в квадратные скобки. Если более поздний поиск, использующий

флаг **s** , обнаруживает больше указанного числа попаданий, отображается сообщение **об ошибке Переполнение** , уведомляющее вас о том, что сохраняются не все попадания.

**l***Длина*
При поиске произвольных строк ASCII или Юникода возвращаются только строки длиной не менее *длины* . Длина по умолчанию — 3. Это значение влияет только на поисковые запросы, использующие флаги **-sa** или **-su** .

**Ж**
Выполняет поиск только в областях памяти, доступных для записи. Необходимо заключить "w" в квадратные скобки.

**1**
Отображает только адреса совпадений поиска в выходных данных поиска. Этот параметр полезен, если вы используете маркер **FOREACH** для передачи выходных данных команды в входные данные другой команды.

*Тип*
Указывает тип памяти для поиска. Добавьте дефис (-) перед *типом* . Можно использовать одно из следующих значений *Type* .

| **Тип** | **Описание** |
|---|---|
| **b** | Байт (8 бит) |
| **w** | WORD (16 бит) |
| **d** | DWORD (32 бита) |
| **Q** | QWORD (64 бит) |
| **a** | Строка ASCII (не обязательно строка с пустым завершением) |
| **u** | Строка Юникода (не обязательно строка, заканчивающаяся null) |

Если опустить *type*, используются байтовые значения. Однако при использовании *флагов* нельзя пропускать *тип*.

**Sa**
Выполняет поиск любой памяти, содержащей печатаемые строки ASCII. Используйте флаг **l***Length* , чтобы указать минимальную длину таких строк. Минимальная длина по умолчанию составляет 3 символа.

**Cy**

Выполняет поиск любой памяти, содержащей печатаемые строки Юникода. Используйте флаг **l***Length* , чтобы указать минимальную длину таких строк. Минимальная длина по умолчанию составляет 3 символа.

*Диапазон*

Указывает область памяти для поиска. Этот диапазон не может превышать 256 МБ, если не используется синтаксис **L?** . Дополнительные сведения об этом синтаксисе см. в разделе Синтаксис адреса и диапазона адресов.

*Узор*

Указывает одно или несколько значений для поиска. По умолчанию эти значения являются значениями байтов. В поле *Tun* можно указать различные типы памяти. Если вы указываете значение WORD, DWORD или QWORD в зависимости от других выбранных параметров, может потребоваться заключить шаблон поиска в одинарные кавычки (например, **"H").**

dbgcmd

```
0:000> s -d @rsp L1000000 'H'
0000003f`ff07ec00  00000048 00000000 500c0163 00000000  H.......c..P....
0000003f`ff07ec50  00000048 00000000 00000080 00000000  H...............
0000003f`ff07efc0  00000048 00000000 400c0060 00000000  H.......`..@....
```

Если вы укажете строку с помощью типа ascii, заключите ее в двойные кавычки (например, **"B7").**

dbgcmd

```
0:000> s -a @rsp L10000000 "B7"
0000003f`ff07ef0a  42 37 ff 7f 00 00 90 38-4e c2 6c 01 00 00 7d 26
B7.....8N.l...}&
0000003f`ff0ff322  42 37 ff 7f 00 00 f8 5d-42 37 ff 7f 00 00 20 41
B7.....]B7.... A
0000003f`ff0ff32a  42 37 ff 7f 00 00 20 41-42 37 ff 7f 00 00 98 59  B7.... 
AB7.....Y
```

**-V**

Выполняет поиск объектов того же типа, что и указанный *объект*.

*Объекта*

Указывает адрес объекта или адрес указателя на объект. Затем отладчик выполняет поиск объектов того же типа, что и объект, задающий *объект Object* .

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения об операциях с памятью и описание других команд, связанных с памятью, см. в разделе Чтение и запись памяти.

# Комментарии

Если отладчик находит указанный шаблон байтов, отладчик отображает первый адрес памяти в области *памяти Диапазон* , где был найден шаблон. Отладчик отображает фрагмент памяти, который начинается с этого расположения, в формате, соответствующем указанному типу памяти *Type* . Если *type***имеет значение** или **u**, отображается содержимое памяти и соответствующие символы ASCII или Юникода.

Параметр *Pattern* необходимо указать в виде ряда байтов, если не указано другое значение *Type* . Можно ввести значения байтов в виде числовых символов или символов ASCII:

- Числовые значения интерпретируются как числа в текущем радиксе (16, 10 или 8). Чтобы изменить радикс по умолчанию, используйте команду **n (задать числовую базу).** Вы можете переопределить радикс по умолчанию, указав префикс **0x** (шестнадцатеричный), префикс **0n** (decimal), префикс **0t** (восьмеричный) или префикс **0y** (двоичный). **Примечание** Радикс по умолчанию работает по-разному при использовании выражений C++. Дополнительные сведения об этих выражениях и радиксе см. в разделе Вычисление выражений.

- Символы ASCII необходимо заключать в одинарные прямые кавычки. Нельзя использовать escape-символы в стиле C (например, "\0" или "\n").

Если указано несколько байтов, их необходимо разделить пробелами.

Команды **s-a** и **s-u** выполняют поиск указанных строк ASCII и Юникода соответственно. Эти строки не обязательно должны заканчиваться null.

Команды **s-sa** и **s-su** выполняют поиск неопределенных строк ASCII и Юникода. Они полезны, если вы проверяете диапазон памяти, чтобы узнать, содержит ли он какие-либо печатные символы. Параметры флагов позволяют указать минимальную длину строки для поиска.

Пример. Следующая команда находит строки ASCII длиной >=3 в диапазоне, начиная с 0000000140000000 и заканчивая 400 байтами позже.

```
dbgcmd
```

```
s-sa 0000000140000000 L400
```

Следующая команда находит строки ASCII длиной >=4 в диапазоне, начиная с 0000000140000000 и заканчивая 400 байтами позже.

```
dbgcmd
```

```
s -[l4]sa 0000000140000000 L400
```

Следующая команда выполняет то же самое, но ограничивает поиск областями памяти, доступными для записи.

```
dbgcmd
```

```
s -[wl4]sa 0000000140000000 L400
```

Следующая команда выполняет то же самое, но отображает только адрес соответствия, а не адрес и значение.

```
dbgcmd
```

```
s -[1wl4]sa 0000000140000000 L400
```

Команда **s-v** выполняет поиск объектов с тем же типом данных, что и объект *Object* . Эту команду можно использовать только в том случае, если нужным объектом является класс C++ или другой объект, связанный с виртуальными таблицами функций (Vtables). Команда **s-v** выполняет поиск адресов виртуальныхtables этого класса в области памяти *Range* . Если в этом классе существует несколько виртуальных таблицы, алгоритм поиска ищет все эти значения указателя, разделенные соответствующим количеством байтов. При обнаружении

совпадений отладчик возвращает базовый адрес объекта и полные сведения об этом объекте, аналогичные выходным данным команды **dt (тип отображения).**

Пример. Предположим, что текущий радикс равен 16. Следующие три команды выполняют одно и то же: поиск расположений памяти от 0012FF40 до 0012FF5F для "Hello".

```
dbgcmd
```

```
0:000> s 0012ff40 L20 'H' 'e' 'l' 'l' 'o'
```

```
dbgcmd
```

```
0:000> s 0012ff40 L20 48 65 6c 6c 6f
```

```
dbgcmd
```

```
0:000> s -a 0012ff40 L20 "Hello"
```

Эти команды находят каждый вид "Hello" и возвращают адрес каждого такого шаблона, т. е. адрес буквы "H".

Отладчик возвращает только шаблоны, которые полностью содержатся в диапазоне поиска. Перекрывающиеся шаблоны находятся правильно. (Другими словами, шаблон "QQQ" встречается три раза в "QQQQQ".)

В следующем примере показан поиск, использующий параметр *Type* . Эта команда выполняет поиск в местах памяти с 0012FF40 по 0012FF5F для двойного слова "VUTS":

```
dbgcmd
```

```
0:000> s -d 0012ff40 L20 'VUTS'
```

На компьютерах с маленьким эндианом "VUTS" совпадает с шаблоном байтов "S" "T" "U" "V". Однако при поиске WORD, DWORD и QWORD возвращаются только те результаты, которые правильно выровнены по байтам.

# so (настройка параметров отладки ядра)

Статья • 05.04.2024

Таким **образом** , наборы команд или отображаются параметры отладки ядра.

```
dbgcmd
```

```
so [Options]
```

## Параметры

*Варианты* одного или нескольких из следующих параметров:

**NOEXTWARNING**
Не выдает предупреждение, если отладчик не может найти команду расширения.

**NOVERSIONCHECK**
Не проверка версию библиотек DLL расширения отладчика.

Если опустить *параметры*, отображаются текущие параметры.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Замечания

Можно также задать параметры отладки ядра с помощью переменной среды _NT_DEBUG_OPTIONS.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# sq (установить тихий режим)

Статья • 08.07.2024

Команда **sq** включает или отключает тихий режим.

```
dbgcmd

sq
sq{e|d}
```

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Команда **sqe** включает спокойный режим, и **команда sqd** отключает ее. Команда **sq** включает и отключает спокойный режим. Каждая из этих команд также отображает новое состояние тихого режима.

Вы можете задать спокойный режим в KD или в режиме ядра WinDbg с помощью переменной среды KDQUIET. (Обратите внимание, что спокойный режим существует как в пользовательском режиме, так и в режиме ядра, но переменная среды KDQUIET распознается только в режиме ядра.)

*Тихий режим* имеет три отдельных эффекта:

- Отладчик не отображает сообщения каждый раз при загрузке или выгрузке библиотеки DLL расширения.
- Команда **r (Registers)** больше не требует знака равенства (=) в его синтаксисе.
- При переходе на целевой компьютер во время отладки ядра сообщение с длинным предупреждением подавляется.

Не путайте тихий режим с эффектами **параметра командной строки -myob** (в CDB и KD) или **параметр командной строки Q** (в WinDbg).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ss (set Symbol Suffix)

Статья • 08.07.2024

Команда **ss** задает или отображает текущее значение суффикса, используемое для сопоставления символов в числовых выражениях.

dbgcmd

```
ss [a|w|n]
```

## Параметры

**a**
Указывает, что суффикс символа должен быть "A", соответствующий многим символам ASCII.

**w**
Указывает, что суффикс символа должен быть "W", соответствующий многим символам Юникода.

**n**
Указывает, что отладчик не должен использовать суффикс символов. (Этот параметр является поведением по умолчанию.)

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о сопоставлении символов см. в разделе "Синтаксис символов" и "Сопоставление символов".

# Замечания

Если указать **команду ss** вместе без параметров, отображается текущее состояние значения суффикса.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# sx, sxd, sxe, sxi, sxn, sxr, sx- (установка исключений)

Статья • 26.10.2023

Команды **sx** управляют действием, которое выполняет отладчик при возникновении исключения в отлаживаемом приложении или при возникновении определенных событий.

dbgcmd

```
sx

sx{e|d|i|n} [-c "Cmd1"] [-c2 "Cmd2"] [-h] {Exception|Event|*}

sx- [-c "Cmd1"] [-c2 "Cmd2"] {Exception|Event|*}

sxr
```

## Параметры

**-c** "*Cmd1*"
Указывает команду, которая выполняется при возникновении исключения или события. Эта команда выполняется при первой возможности обработки этого исключения, независимо от того, входит ли это исключение в отладчик. Строку *Cmd1* необходимо заключить в кавычки. Эта строка может содержать несколько команд, если разделять их точкой с запятой. Пространство между -c и строкой команды в кавычках является необязательным.

**-c2**"*Cmd2*"
Указывает команду, которая выполняется, если возникает исключение или событие и не обрабатывается при первой же возможности. Эта команда выполняется, когда возникает вторая возможность обработки этого исключения, независимо от того, входит ли это исключение в отладчик. Строку *Cmd2* необходимо заключить в кавычки. Эта строка может содержать несколько команд, если разделять их точкой с запятой. Пробел между -c2 и командной строкой в кавычках необязателен.

**-h**
Изменяет состояние обработки указанного события вместо состояния приостановки. Если *event* имеет значение **cc**, **hc**, **bpec** или **ssec**, вам не нужно использовать параметр **-h** .

*Exception*
Указывает номер исключения, с которым выполняется команда в текущем радиксе.

*Событие*
Указывает событие, с которым выполняется команда. Эти события идентифицируются короткими сокращениями. Список событий см. в разделе Управление исключениями и событиями.

*
Влияет на все исключения, которые не называются явным образом для **sx**. Список исключений с явным именем см. в разделе Управление исключениями и событиями.

## Среда

| | |
|---|---|
| **Режимы** | Пользовательский режим, режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Все |

## Дополнительные сведения

Дополнительные сведения о состоянии прерывания и состоянии обработки, описания всех кодов событий, список состояния по умолчанию для всех событий и другие методы управления этим состоянием см. в разделе Управление исключениями и событиями.

# Комментарии

Команда **sx** отображает список исключений для текущего процесса и список всех событий, не являющихся исключениями, а также поведение отладчика по умолчанию для каждого исключения и события.

Команды **sxe**, **sxd**, **sxn** и **sxi** управляют параметрами отладчика для каждого исключения и события.

Команда **sxr** сбрасывает все состояния исключений и фильтра событий до параметров по умолчанию. Например, команды очищаются и прерываются, а параметры продолжения сбрасываются до параметров по умолчанию.

Команда **sx-не** изменяет состояние обработки или состояние прерывания указанного исключения или события. Эту команду можно использовать, если вы хотите изменить команду первого или второго шанса, связанную с определенным событием, но не хотите изменять что-либо другое.

Если вы включаете параметр **-h** (или если указаны события **cc**, **hc**, **bpec** или **ssec** ), команды **sxe**, **sxd**, **sxn** и **sxi** управляют состоянием обработки исключения или события. Во всех остальных случаях эти команды управляют состоянием прерывания исключения или события.

При настройке состояния приостановки эти команды оказывают следующие эффекты:

| **Get-Help** | **Имя состояния** | **Описание** |
|---|---|---|
| **sxe** | **Разбиение (Включено)** | При возникновении этого исключения целевой объект немедленно врывается в отладчик перед активацией других обработчиков ошибок. Такой тип обработки называется *обработкой первого шанса* . |
| **Sxd** | **Второй шанс перерыв (Отключено)** | Отладчик не прерывается для исключения первого случая этого типа (хотя отображается сообщение). Если другие обработчики ошибок не устраняют это исключение, выполнение останавливается, а целевой объект прерывается в отладчике. Такой тип обработки называется *обработкой второго шанса* . |
| **sxn** | **Выходные данные (Уведомлять)** | При возникновении этого исключения целевое приложение вообще не переходит в отладчик. Однако отображается сообщение, уведомляющее пользователя об этом исключении. |
| **Sxi** | **Пропуск** | При возникновении этого исключения целевое приложение вообще не переходит в отладчик, и сообщение не отображается. |

При настройке состояния обработки эти команды оказывают следующие эффекты:

| **Get-Help** | **Имя состояния** | **Описание** |
|---|---|---|
| **sxe** | **Обрабатываются** | Событие считается обработанным при возобновлении выполнения. |
| **sxd,sxn,sxi** | **Не обработано** | Событие считается не обработанным при возобновлении выполнения. |

Параметр **-h** можно использовать вместе с исключениями, а не событиями. Используйте этот параметр с **ch**, **bpe** или **sse** , чтобы задать состояние обработки

для **hc**, **bpec** или **ssec** соответственно. Если вы используете параметр **-h** с любым другим событием, это не оказывает никакого влияния.

Используйте параметры **-c** или **-c2** с **hc**, **bpec** или **ssec** , чтобы связать указанные команды с **ch**, **bpe** или **sse** соответственно.

В следующем примере команда **sxe** используется для установки состояния прерывания событий нарушения доступа для прерывания при первом шансе, а для команды первого шанса, которая будет выполняться в этот момент, **значения r eax**. Затем команда **sx-** используется для изменения команды первого шанса на **r ebx** без изменения состояния обработки. Наконец, отображается часть выходных данных **sx** , указывающая текущие параметры для событий нарушения доступа:

dbgcmd

```
0:000> sxe -c "r eax" av

0:000> sx- -c "r ebx" av

0:000> sx
 av - Access violation - break - not handled
       Command: "r ebx"
  . . .
```

# См. также раздел

- Использование точек останова — методы отладки
- Условные точки останова в WinDbg
- Выполнение до достижения указанного состояния

# t (трассировка)

Статья • 05.04.2024

Команда **t** выполняет одну инструкцию или исходную строку и при необходимости отображает полученные значения всех регистров и флагов. При возникновении вызовов или прерываний подзадачей выполняется трассировка каждого шага.

Режим пользователя

```
dbgcmd
```

```
[~Thread] t [r] [= StartAddress] [Count] ["Command"]
```

Режим ядра

```
dbgcmd
```

```
t [r] [= StartAddress] [Count] ["Command"]
```

# Параметры

*Поток*
Указывает потоки для оттепели. Все остальные потоки заморожены. Дополнительные сведения об этом синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд pr**, **tr** или .prompt_allow -reg. Все три из этих команд управляют одинаковым параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других трех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором должно начинаться выполнение. Если вы не используете *StartAddress*, выполнение начинается с инструкции, на которую

указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Численность*
Указывает количество инструкций или исходных строк для трассировки перед остановкой. Каждый шаг отображается как отдельное действие в окне команды отладчика. Значение по умолчанию равно единице.

*Command*
Указывает команду отладчика, выполняемую после выполнения трассировки. Эта команда выполняется до отображения стандартных **результатов t** . Если вы также используете *Count*, эта команда выполняется после завершения трассировки (но до отображения результатов из конечной трассировки).

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о том, как выдавать **команду t** и обзор связанных команд, см. в разделе "Управление целевым объектом".

# Замечания

При указании *count* каждая инструкция отображается так, как она выполняется.

Каждая трассировка выполняет одну инструкцию сборки или одну исходную строку в зависимости от того, находится ли отладчик в режиме сборки или в исходном режиме. **Используйте команды l+t и l-t** или кнопки на панели инструментов WinDbg, чтобы переключаться между этими режимами.

Если вы хотите отслеживать большинство вызовов функций, но пропускать определенные вызовы, можно использовать **.step_filter (задать фильтр шагов),**

чтобы указать, какие вызовы следует выполнить шаг.

Команду t **можно использовать для трассировки инструкций** в POM.

При быстрой трассировке во многих случаях в WinDbg окна сведений отладки обновляются после каждой трассировки. Если это обновление приводит к замедлению времени отклика, используйте **.suspend_ui (приостановка интерфейса WinDbg)** для временной приостановки обновления этих окон.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ta (трассировка по адресу)

Статья • 08.07.2024

Команда **ta** выполняет программу до достижения указанного адреса, отображая каждый шаг (включая шаги в вызываемой функции).

Режим пользователя

```
dbgcmd
```

```
[~Thread] ta [r] [= StartAddress] StopAddress
```

Режим ядра

```
dbgcmd
```

```
ta [r] [= StartAddress] StopAddress
```

# Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд tar**, pr**, tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром и использованием любого из них переопределяет любое предыдущее использование этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других четырех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. Если вы не используете *StartAddress*, выполнение начинается с инструкции, на которую указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*StopAddress*
Указывает адрес, на котором останавливается выполнение. Этот адрес должен соответствовать точному адресу инструкции.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **ta** приводит к началу выполнения целевого объекта. Это выполнение продолжается до достижения указанной инструкции или точки останова.

**Обратите внимание** , что если вы используете **команду ta** в режиме ядра, выполнение останавливается при обнаружении инструкции по указанному виртуальному адресу в любом виртуальном адресном пространстве.

Во время этого выполнения все шаги отображаются явным образом. Если функция вызывается, отладчик также выполняет трассировку с помощью этой функции. Таким образом, отображение этой команды напоминает то, что вы видите, если вы неоднократно выполняли **t (trace)** до тех пор, пока счетчик программы не достиг указанного адреса.

Например, следующая команда явно выполняет трассировку по целевому коду до достижения возвращаемого адреса текущей функции.

dbgcmd

```
0:000> ta @$ra
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# tb (трассировка до следующей ветви)

Статья • 08.07.2024

Команда **tb** выполняет программу до достижения инструкции ветви.

```
dbgcmd
```

```
tb [r] [= StartAddress] [Count]
```

## Параметры

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд tbr**, pr**,** **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других четырех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором отладчик запускает выполнение. Если вы не используете *StartAddress*, выполнение начинается с инструкции, на которую указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает число разрешенных ветвей. При каждом обнаружении ветви адрес инструкции и инструкция отображаются. Если вы опустите *число*, значение по умолчанию равно 1.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | **x86:** режим ядра только **на основе x64:** режим пользователя, режим ядра |

| | |
|---|---|
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | на основе x86 (семейство процессоров 6 и более поздних версий) на основе x64 |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **tb** приводит к началу выполнения целевого объекта. Это выполнение продолжается до достижения команды ветви.

Выполнение останавливается в любой команде ветви, которую необходимо выполнить. Эта остановка выполнения всегда основана на коде *дизассемблии* , даже если отладчик находится в исходном режиме.

Инструкции ветви включают вызовы, возвраты, переходы, подсчитываемые циклы и циклы. Если отладчик встречает безусловную ветвь или условную ветвь, для которой условие имеет значение true, выполнение останавливается. Если отладчик обнаруживает условную ветвь, условие которой равно false, выполнение продолжается.

При остановке выполнения отображается адрес инструкции ветви и все связанные символы. За этой информацией следует стрелка, а затем адрес и инструкции нового расположения счетчика программ.

Команда **tb** работает только на текущем процессоре. Если вы используете **тб** в многопроцессорной системе, выполнение останавливается при достижении команды ветви или при возникновении события другого процессора, в зависимости от того, что происходит первым.

Обычно трассировка ветви включена после инициализации блока управления процессором (PRCB). (PRCB инициализирован в начале процесса загрузки.) Однако если перед этой точкой необходимо использовать команду tb**, можно использовать .force_tb (принудительно разрешить трассировку ветви)** для включения трассировки ветви ранее. **Используйте команду .force_tb** осторожно, так как она может повредить состояние процессора.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# tc (трассировка до следующего вызова)

Статья • 08.07.2024

Команда **tc** выполняет программу до достижения инструкции вызова.

Режим пользователя

```
dbgcmd
```

```
[~Thread] tc [r] [= StartAddress] [Count]
```

Режим ядра

```
dbgcmd
```

```
tc [r] [= StartAddress] [Count]
```

## Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд tcr**, pr, **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других четырех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. Если вы не используете *StartAddress*, выполнение начинается с инструкции, на которую

указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает количество инструкций по вызову, с которым отладчик должен столкнуться для **завершения команды tc**. Значение по умолчанию равно единице.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **tc** приводит к началу выполнения целевого объекта. Это выполнение продолжается до тех пор, пока отладчик не достигнет **инструкции вызова** или обнаруживает точку останова.

Если счетчик программы уже находится в **инструкции вызова** , отладчик трассирует вызов и продолжает выполняться до тех пор, пока он не столкнется с другим **вызовом**. Эта трассировка, а не выполнение вызова является единственной разницей между **tc** и **pc (шаг к следующему вызову).**

В исходном режиме можно связать одну исходную строку с несколькими инструкциями по сборке. Эта команда не останавливается в инструкции вызова, **связанной с текущей** исходной строкой.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# tct (трассировка до следующего вызова или возврата)

Статья • 08.07.2024

Команда **tct** выполняет программу, пока она не достигнет инструкции вызова или возвращаемой инструкции.

Режим пользователя

```
dbgcmd
```

```
[~Thread] tct [r] [= StartAddress] [Count]
```

Режим ядра

```
dbgcmd
```

```
tct [r] [= StartAddress] [Count]
```

# Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд tctr**, pr, **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других четырех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. Если вы не используете *StartAddress*, выполнение начинается с инструкции, на которую

указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает количество вызовов или возвращаемых **инструкций, с которыми должен столкнуться отладчик для завершения команды tct.** Значение по умолчанию равно единице.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **tct** приводит к началу выполнения целевого объекта. Это выполнение продолжается до тех пор, пока отладчик не достигнет **инструкции вызова** или возврата **или** не обнаруживает точку останова.

Если счетчик программы уже находится на вызове или возврате **инструкции, отладчик трассирует вызов или** возврат и продолжает выполняться до тех пор, пока он не столкнется с другим вызовом или **возвратом**. Эта трассировка, а не выполнение вызова является единственной разницей между **tct** и **pct (шаг к следующему вызову или возврату).**

В исходном режиме можно связать одну исходную строку с несколькими инструкциями по сборке. Эта команда не останавливается при вызове **или возврате инструкции, связанной с текущей** исходной строкой.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# th (трассировка до следующей ветви инструкции)

Статья • 08.07.2024

Команда **th** выполняет программу до тех пор, пока она не достигнет любого типа ветвления, включая условные или безусловные ветви, вызовы, возвраты и системные вызовы.

Режим пользователя

```
dbgcmd
```

```
[~Thread] th [r] [= StartAddress] [Count]
```

Режим ядра

```
dbgcmd
```

```
th [r] [= StartAddress] [Count]
```

## Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд thr**, pr, **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других четырех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. Если вы не

используете *StartAddress*, выполнение начинается с инструкции, на которую указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает количество ветвей инструкций, которые отладчик должен столкнуться для **завершения команды th** . Значение по умолчанию равно единице.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **th** приводит к началу выполнения целевого объекта. Выполнение продолжается до тех пор, пока отладчик не достигнет ветвления или не обнаруживает точку останова.

Если счетчик программы уже находится в ветвления, отладчик выполняет трассировку в инструкцию ветвления и продолжает выполняться до достижения другой инструкции ветвления. Эта трассировка, а не выполнение вызова является единственным различием между **th** и **ph (шаг к следующей ветви инструкции)**.

**th** доступен для всех динамических сеансов. Эта доступность является основным различием между **th** и **тб (трассировка до следующей ветви).**

В исходном режиме можно связать одну исходную строку с несколькими инструкциями по сборке. Эта команда не останавливается в инструкции ветвления, связанной с текущей исходной строкой.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# tt (трассировка до следующего возврата)

Статья • 08.07.2024

Команда **tt** выполняет программу до достижения инструкции возврата.

Режим пользователя

```
dbgcmd
```

```
[~Thread] tt [r] [= StartAddress] [Count]
```

Режим ядра

```
dbgcmd
```

```
tt [r] [= StartAddress] [Count]
```

## Параметры

*Нить*
Указывает потоки для продолжения выполнения. Все остальные потоки заморожены. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис потока". Потоки можно указать только в пользовательском режиме.

**r**
Включает и отключает отображение регистров и флагов. По умолчанию отображаются регистры и флаги. Вы можете отключить отображение регистрации с помощью **команд ttr**, pr**,** **tr** или .prompt_allow -reg. Все эти команды управляют тем же параметром, и вы можете использовать любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других четырех команд. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

*StartAddress*
Указывает адрес, в котором начинается выполнение отладчика. Если вы не используете *StartAddress*, выполнение начинается с инструкции, на которую

указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Считать*
Указывает количество инструкций **возврата** , которые отладчик должен столкнуться для **завершения команды th** . Значение по умолчанию равно единице.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | режим пользователя, режим ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о связанных командах см. в разделе "Управление целевым объектом".

# Замечания

Команда **tt** приводит к началу выполнения целевого объекта. Это выполнение продолжается до тех пор, пока отладчик не достигнет **инструкции возврата** или не обнаруживает точку останова

Если счетчик программы уже находится в **инструкции возврата** , отладчик трассирует возврат и продолжает выполняться до тех пор, пока не будет достигнут другой **возврат** . Эта трассировка, а не выполнение вызова является единственной разницей между **tt** и **pt (шаг к следующему возврату)**.

В исходном режиме можно связать одну исходную строку с несколькими инструкциями по сборке. Эта команда не останавливается в инструкции возврата, **связанной с текущей** исходной строкой.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# u, ub, uu (Разобрать)

Статья • 08.05.2025

Команды `u*` отображают в памяти ассемблерную трансляцию указанного программного кода.

Эту команду не следует путать с командой **~u (Unfreeze Thread).**

dbgcmd

```
u[u|b] Range
u[u|b] Address
u[u|b]
```

## Параметры

*Диапазон*
Указывает диапазон памяти, содержащий инструкции по разборке. Дополнительные сведения о синтаксисе см. в разделе Синтаксис адресов и диапазонов адресов. Если вы используете флаг **b** , вы должны указать *Range* с помощью синтаксиса "*Address***L***Length*", а не синтаксиса "*Address1 Address2*".

*адрес*
Указывает начало диапазона памяти для разбора. Восемь инструкций на процессоре на базе x86 разымаются. Дополнительные сведения о синтаксисе см. в разделе Синтаксис адресов и диапазонов адресов.

**b**
Определяет диапазон памяти для разборки путем обратного отсчета. Если используется **ub***Address* , то разобранный диапазон будет состоять из восьми инструкций, предшествующих *Address*. Если диапазон указан с помощью синтаксиса **ub***Address***L***Length*, то дизассемблированный диапазон будет диапазоном указанной длины, оканчивающимся на *Address*.

**u**
Указывает, что дизассемблирование будет продолжаться даже при возникновении ошибки чтения памяти.

## Окружающая среда

Развернуть таблицу

| Товар | Описание |
| --- | --- |
| Режимы | Режим пользователя, режим ядра |
| Цели | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительные сведения

Дополнительные сведения об отладке сборки и связанных командах см. в разделах Отладка в режиме сборки и Аннотированное дизассемблирование x86.

# Замечания

Если параметр для команды u не указан, дизассемблирование начинается по текущему адресу и распространяется на восемь инструкций на процессоре на базе x86 или x64. При использовании `ub` без параметра дизассемблирование включает в себя восемь инструкций перед текущим адресом. Дополнительные сведения о работе с командами отладчика см. в разделе Использование команд отладчика.

Не путайте эту команду с **командой up (Unassemble from Physical Memory).** Команда **u** дизассемблирует только виртуальную память, в то время как команда **up** дизассемблирует только физическую память.

# uf (Unassemble Function)

Статья • 08.07.2024

Команда **uf** отображает перевод сборки указанной функции в памяти.

```
dbgcmd
```

```
uf [Options] Address
```

## Параметры

*Варианты* одного или нескольких из следующих параметров:

**/c**
Отображает только инструкции по вызову в подпрограмме вместо полной расзаполнения. Инструкции по вызову могут быть полезны для определения связей вызывающего и вызываемого объекта из разобранного кода.

**/D**
Создает имена связанных вызывающих абонентов для навигации графа вызовов.

**/m**
Смягчает требования блокировки для разрешения нескольких выходов.

**/o**
Сортирует отображение по адресу вместо смещения функции. Этот параметр представляет представление макета памяти полной функции.

**/O**
Создает связанные строки вызовов для доступа к сведениям о вызове и создания точек останова.

**/i**
Отображает количество инструкций в подпрограмме.

*Адрес*
Указывает адрес функции для дизассембла. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения об отладке сборки и связанных командах см. в разделе "Отладка в режиме сборки".

# Замечания

Отображение отображает всю функцию в соответствии с порядком функции.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# up (Unassemble from Physical Memory)

Статья • 08.07.2024

Команда **up** отображает перевод сборки указанного кода программы в физической памяти.

dbgcmd

```
up Range
up Address
up
```

## Параметры

*Диапазон*
Указывает диапазон памяти в физической памяти, содержащий инструкции по дизассемблю. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Адрес*
Указывает начало диапазона памяти в физической памяти, чтобы диассемблировать. Восемь инструкций на процессоре на основе x86 неубраны. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения об отладке сборки и связанных командах см. в разделе "Отладка в режиме сборки".

# Замечания

Если параметр для **команды up** не указан, дезассембли начинается с текущего адреса и расширяет восемь инструкций на процессоре на основе x86.

Не путайте эту команду с u (Unassemble). Команда **up** дизассембляет только физическую память, а **команда u** дизассембляет только виртуальную память.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# your (Unassemble Real Mode BIOS)

Статья • 20.05.2024

**Команда** отображает перевод сборки указанного 16-разрядного кода реального режима.

dbgcmd

```
ur Range
ur Address
ur
```

## Параметры

*Диапазон*
Указывает диапазон памяти, содержащий инструкции по дизассемблю. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

*Адрес*
Указывает начало диапазона памяти для разочаровки. Восемь инструкций на процессоре на основе x86 неубраны. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения об отладке кода BIOS см. в разделе "Отладка BIOS Code".

# Замечания

Если вы не указываете *диапазон* или *адрес*, дизассембли начинается с текущего адреса и расширяет восемь инструкций на процессоре на основе x86.

Если вы изучаете 16-разрядный код реального режима на процессоре x86, команда и **команда u (Unassemble)** дают правильные результаты.

Однако если код реального режима существует в расположении, где отладчик не ожидает его (например, компьютер, отличный от x86, работающий или эмулирующий код BIOS на основе x86 из подключаемого модуля карта), необходимо использовать **для** правильной десассемблирования этого кода.

Если вы используете 32-разрядный или 64-разрядный код, команда пытается дизассемблировать код, как будто это был 16-разрядный код. Эта ситуация приводит к бессмысленным результатам.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# ux (Unassemble x86 BIOS)

Статья • 08.07.2024

Команда **ux** отображает набор инструкций кода BIOS на основе x86.

```
ux [Address]
```

## Параметры

*Адрес*
Указывает смещение памяти в коде BIOS на основе x86. Если этот параметр опущен или указан ноль, смещение по умолчанию является началом BIOS.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Только на основе x86 |

## Дополнительная информация

Дополнительные сведения об отладке кода BIOS см. в разделе "Отладка BIOS Code".

## Замечания

Отладчик отображает инструкции, созданные из первых восьми строк кода, начиная *со смещения адреса* .

Чтобы команда **ux** работала правильно, символы HAL должны быть доступны отладчику. Если отладчик не может найти эти символы, отладчик отображает ошибку "не удалось устранить".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# vercommand (показать командную строку отладчика)

Статья • 26.10.2023

Команда **vercommand** отображает команду, открывающую отладчик.

```
vercommand
```

## Среда

| Элемент | Описание |
| --- | --- |
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# version (Показать версию отладчика)

Статья • 26.10.2023

Команда **version** отображает сведения о версии отладчика и все загруженные библиотеки DLL расширения. Эта команда также отображает текущую версию операционной системы целевого компьютера.

Не путайте эту команду с командой расширения **!version (Показать версию DLL).**

```
version
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## См. также

**CTRL+W (показать версию отладчика)**

**vertarget (показать версию целевого компьютера)**

# vertarget (показать версию целевого компьютера)

Статья • 26.10.2023

Команда **vertarget** отображает текущую версию операционной системы Microsoft Windows на целевом компьютере.

```
vertarget
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## См. также

**CTRL+W (показать версию отладчика)**

**version (Показать версию отладчика)**

# wrmsr (запись MSR)

Статья • 05.04.2024

Команда **wrmsr** записывает значение в регистр для конкретной модели (MSR) по указанному адресу.

```
wrmsr Address Value
```

## Параметры

*Адрес*
Указывает адрес MSR.

*Value*
Указывает 64-разрядное шестнадцатеричное значение для записи в MSR.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Все |

## Замечания

Команда **wrmsr может задать msR** на платформах на основе x86 и x64.
Определения MSR зависят от платформы.

## См. также

rdmsr (чтение MSR)

---

## Обратная связь

**Были ли сведения на этой странице полезными?**

Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# wt (данные трассировки и контрольных значений)

Статья • 26.10.2023

Команда **wt** выполняется через всю функцию, а затем отображает статистику при выполнении этой команды в начале вызова функции.

dbgcmd

```
wt [WatchOptions] [= StartAddress] [EndAddress]
```

## Параметры

*WatchOptions*
Указывает способ изменения отображения. Можно использовать любой из следующих вариантов.

| Параметр | Действие |
|---|---|
| **-I***Глубина* | (Только в пользовательском режиме) Указывает максимальную глубину отображаемых вызовов. Все вызовы, уровни *глубины* которых по крайней мере глубже начальной точки, выполняются автоматически. |
| **Модуль -m** | (Только в пользовательском режиме) Ограничивает отображение кодом внутри указанного модуля, а также первым уровнем вызовов, выполненных из этого модуля. Вы можете включить несколько параметров -m для отображения кода из нескольких модулей и ни одного другого модуля. |
| **Модуль -i** | (Только в пользовательском режиме) Игнорирует любой код в указанном модуле. Можно включить несколько параметров -i, чтобы игнорировать код из нескольких модулей. При использовании параметра -m отладчик игнорирует все параметры -i. |
| **-Ni** | (Только в пользовательском режиме) Не отображает ни одну запись в коде, которая игнорируется из-за параметра -m или -i. |

| Параметр | Действие |
| --- | --- |
| **-Nc** | Не отображает сведения об отдельных звонках. |
| **-Ns** | Не отображает сводную информацию. |
| **-Nw** | Не отображает предупреждения во время трассировки. |
| **-Oa** | (Только в пользовательском режиме) Отображает фактический адрес сайтов звонков. |
| **-or** | (Только в пользовательском режиме) Отображает возвращаемые значения регистра вызываемой функции, используя в качестве основы радикс по умолчанию. |
| **-Или** | (Только в пользовательском режиме) Отображает возвращаемые значения регистра вызываемой функции в соответствующем типе для каждого возвращаемого значения. |

*StartAddress*
Указывает адрес, с которого начинается выполнение отладчика. Если вы не используете *StartAddress*, выполнение начинается с инструкции, на которую указывает указатель инструкции. Дополнительные сведения о синтаксисе см. в разделе Синтаксис адреса и диапазона адресов.

*EndAddress*
Указывает адрес, на котором заканчивается трассировка. Если вы не *используете EndAddress*, выполняется одна инструкция или вызов функции.

## Среда

| | |
| --- | --- |
| **Режимы** | Пользовательский режим, режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | **Пользовательский режим:** все **режимы ядра:** только на основе x86 |

## Дополнительные сведения

Дополнительные сведения о выполнении команды **wt** и обзор связанных команд см. в разделе Управление целевым объектом.

# Комментарии

Команда **wt** полезна, если требуется информация о поведении конкретной функции, но вы не хотите выполнять ее пошагово. Вместо этого перейдите к началу этой функции и выполните команду **wt** .

Если счетчик программы находится в точке, соответствующей символу (например, началу функции или точке входа в модуль), команда **wt** выполняет трассировку, пока не достигнет текущего возвращаемого адреса. Если счетчик программы находится в инструкции **вызова** , команда **wt** выполняет трассировку, пока не вернется в текущее расположение. Эта трассировка профилируется в окне Команда отладчика вместе с выходными данными, описывающими различные вызовы, встречающиеся командой.

Если команда **wt** выполняется где-то, отличном от начала функции, она ведет себя как команда **p (шаг).** Однако если указать *EndAddress*, выполнение продолжается до тех пор, пока не будет достигнут этот адрес, даже если это выполнение включает в себя множество шагов программы и вызовов функций.

При отладке в исходном режиме следует выполнять трассировку в функцию только до точки, где отображается открывающая скобка текста функции. Затем можно использовать команду **wt** . (Обычно проще вставить точку останова в первую строку функции или использовать **отладку | Запустите до курсора**, а затем используйте команду **wt** .)

Так как выходные данные **из wt** могут быть длинными, для записи выходных данных может потребоваться использовать файл журнала.

В следующем примере показан типичный файл журнала.

dbgcmd

```
0:000> l+                   Source options set to show source lines
Source options are f:
     1/t - Step/trace by source line
     2/l - List source line for LN and prompt
     4/s - List source code at prompt
     8/o - Only show source code at prompt
0:000> p                    Not yet at the function call: use "p"
>  44:       minorVariableOne = 12;
0:000> p
>  45:       variableOne = myFunction(2, minorVariable);
```

```
0:000> t                    At the function call: now use "t"
MyModule!ILT+10(_myFunction):
0040100f e9cce60000      jmp     MyModule!myFunction (0040f6e0)
0:000> t
>  231:    {
0:000> wt                   At the function beginning:  now use "wt"
Tracing MyModule!myFunction to return address 00401137

  105     0 [  0] MyModule!myFunction
    1     0 [  1]   MyModule!ILT+1555(_printf)
    9     0 [  1]   MyModule!printf
    1     0 [  2]     MyModule!ILT+370(__stbuf)
   11     0 [  2]     MyModule!_stbuf
    1     0 [  3]       MyModule!ILT+1440(__isatty)
   14     0 [  3]       MyModule!_isatty
   50    15 [  2]     MyModule!_stbuf
   17    66 [  1]   MyModule!printf
    1     0 [  2]     MyModule!ILT+980(__output)
   59     0 [  2]     MyModule!_output
   39     0 [  3]       MyModule!write_char
  111    39 [  2]     MyModule!_output
   39     0 [  3]       MyModule!write_char

....

   11     0 [  5]           kernel32!__SEH_epilog4
   54 11675 [  4]         kernel32!ReadFile
  165 11729 [  3]       MyModule!_read
  100 11895 [  2]     MyModule!_filbuf
   91 11996 [  1]   MyModule!fgets
54545 83789 [  0] MyModule!myFunction
    1     0 [  1]   MyModule!ILT+1265(__RTC_CheckEsp)
    2     0 [  1]   MyModule!_RTC_CheckEsp
54547 83782 [  0] MyModule!myFunction

112379 instructions were executed in 112378 events (0 from other threads)

Function Name                               Invocations MinInst MaxInst
AvgInst
MyModule!ILT+1265(__RTC_CheckEsp)                     1       1       1
1
MyModule!ILT+1440(__isatty)                          21       1       1
1
MyModule!ILT+1540(__ftbuf)                           21       1       1
1
....
ntdll!memcpy                                         24       1      40
19
ntdll!memset                                          2      29      29
29

23 system calls were executed

Calls  System Call
   23  ntdll!KiFastSystemCall
```

В списке трассировки первое число указывает количество выполненных инструкций, второе — количество инструкций, выполненных дочерними процессами функции, а третье число (в квадратных скобках) — глубину функции в стеке (принимая начальную функцию за ноль). Отступ имени функции показывает глубину вызова.

В предыдущем примере **MyModule!myFunction** выполняет 105 инструкций перед вызовом нескольких подпрограмм, включая **printf** и **fgets**, а затем выполняет дополнительные инструкции 54545 после вызова этих функций, но перед выполнением еще нескольких вызовов. Однако в окончательном подсчете отображается, что **myFunction** выполняет 112 379 инструкций, так как это число включает все инструкции, выполняемые **myFunction** и его дочерними элементами. (Дочерние *элементы***myFunction** — это функции, вызываемые из **myFunction** прямо или косвенно.)

Обратите внимание, что в предыдущем примере **ILT+1440 (__isatty)** вызывается 21 раз. В последнем подсчете сводка по поведению этой функции показывает количество вызовов, наименьшее количество инструкций в одном выполнении, наибольшее число инструкций в одном выполнении и среднее число инструкций за выполнение.

Если выполняются какие-либо системные вызовы, они отображаются в счетчике и снова отображаются в конце выходных данных команды.

# x (изучение символов)

Статья • 26.10.2023

Команда **x** отображает символы во всех контекстах, соответствующих указанному шаблону.

```
dbgcmd
```

```
x [Options] Module!Symbol
x [Options] *
```

## Параметры

*Параметры* Задает параметры поиска символов. Можно использовать один или несколько из следующих параметров:

**/0**
Отображает только адрес каждого символа.

**/1**
Отображает только имя каждого символа.

**/2**
Отображает только адрес и имя каждого символа (не тип данных).

**/D**
Отображает выходные данные с помощью языка разметки отладчика.

**/T**
Отображает тип данных каждого символа, если тип данных известен.

**/V**
Отображает тип символа (локальный, глобальный, параметр, функция или неизвестный) каждого символа. Этот параметр также отображает размер каждого символа. Размер символа функции — это размер функции в памяти. Размер других символов — это размер типа данных, который представляет символ. Размер всегда измеряется в байтах и отображается в шестнадцатеричном формате.

**Размер /s**
Отображение только тех символов, размер которых в байтах равен значению *Size*. *Размер* символа функции — это размер функции в памяти. *Размер* других символов — это размер типа данных, который представляет символ. Всегда отображаются

символы, размер которых не удается определить. *Размер* должен быть ненулевым целым числом.

**/Q**
Отображает имена символов в кавычках.

**/P**
Пропускает пробел перед открывающей скобкой, когда отладчик отображает имя функции и ее аргументы. Этот вид отображения может упростить работу, если вы копируете имена функций и аргументы из отображения **x** в другое расположение.

**/F**
Отображает размер данных функции.

**/D**
Отображает размер данных.

**/a**
Сортирует отображение по адресу в порядке возрастания.

**/A**
Сортирует отображение по адресу в порядке убывания.

**/N**
Сортирует отображение по имени в порядке возрастания.

**/N**
Сортирует отображение по имени в порядке убывания.

**/Z**
Сортирует отображение по размеру в порядке возрастания.

**/Z**
Сортирует отображение по размеру в порядке убывания.

*Модуль*
Указывает модуль для поиска. Этот модуль может быть файлом .exe, .dll или .sys. *Модуль* может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе см. в разделе Синтаксис подстановочных знаков строки.

*Символ*
Указывает шаблон, который должен содержать символ. *Символ* может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе см. в разделе Синтаксис подстановочных знаков строки.

Так как этот шаблон сопоставляется с символом, совпадение не учитывает регистр, а один символ подчеркивания в начале (_) представляет любое количество символов подчеркивания в начале. Вы можете добавить пробелы в *symbol*, чтобы можно было указать имена символов, которые содержат пробелы (например, "оператор new" или "Шаблон<A, B>"), не используя подстановочные знаки.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

Следующая команда находит все символы в MyModule, содержащие строку spin.

dbgcmd

```
0:000> x mymodule!*spin*
```

Следующая команда быстро находит символы DownloadMinor и DownloadMajor в MyModule.

dbgcmd

```
0:000> x mymodule!downloadm??or
```

Вы также можете отобразить все символы в MyModule с помощью следующей команды.

dbgcmd

```
0:000> x mymodule!*
```

Предыдущие команды также заставляют отладчик перезагрузить сведения о символах из MyModule. Если вы хотите перезагрузить символы в модуле с минимальным отображением, используйте следующую команду.

dbgcmd

```
0:000> x mymodule!*start*
```

Некоторые символы всегда содержат строку "start". Таким образом, предыдущая команда всегда отображает некоторые выходные данные, чтобы убедиться, что команда работает. Но предыдущая команда позволяет избежать чрезмерной длины отображения **x mymodule!***.

На экране отображается начальный адрес каждого символа и полное имя символа. Если символом является имя функции, отображение также включает список ее типов аргументов. Если символ является глобальной переменной, отображается его текущее значение.

Существует еще один особый случай команды **x** . Чтобы отобразить адреса и имена всех локальных переменных для текущего контекста, используйте следующую команду.

dbgcmd

```
0:000> x *
```

**Примечание** В большинстве случаев доступ к локальным переменным невозможен, если не были загружены закрытые символы. Дополнительные сведения об этой ситуации см. в разделе dbgerr005: требуются закрытые символы. Чтобы отобразить значения локальных переменных, используйте команду **dv (Отобразить локальные переменные).**

В следующем примере показаны параметры **/0**, **/1** и **/2** .

dbgcmd

```
0:000:x86> x /0 MyApp!Add*
00b51410
00b513d0

0:000:x86> x /1 MyApp!Add*
MyApp!AddThreeIntegers
MyApp!AddTwoIntegers

0:000:x86> x /2 MyApp!Add*
00b51410          MyApp!AddThreeIntegers
00b513d0          MyApp!AddTwoIntegers
```

Параметры **/0**, **/1** и **/2** полезны, если вы хотите использовать выходные данные команды **x** в качестве входных данных для команды **FOREACH** .

dbgcmd

```
.foreach ( place { x /0 MyApp!*MySym*} ) { .echo ${place}+0x18 }
```

В следующем примере показан параметр **/f** , используемый для фильтрации функций в модуле notepad.exe.

dbgcmd

```
0:000> x /f /v notepad!*main*
prv func   00000001`00003340  249 notepad!WinMain (struct HINSTANCE__ *,
struct HINSTANCE__ *, char *, int)
prv func   00000001`0000a7b0   1c notepad!WinMainCRTStartup$filt$0 (void)
prv func   00000001`0000a540  268 notepad!WinMainCRTStartup (void)
```

При использовании параметра **/v** в первом столбце отображается тип символа (локальный, глобальный, параметр, функция или неизвестно). Второй столбец — это адрес символа. Третий столбец — это размер символа в байтах. В четвертом столбце показано имя модуля и имя символа. В некоторых случаях за этим отображением следует знак равенства (=), а затем тип данных символа. Также отображается источник символа (общедоступная или полная информация о символе).

dbgcmd

```
kd> x /v nt!CmType*
global 806c9e68    0 nt!CmTypeName = struct _UNICODE_STRING []
global 806c9e68  150 nt!CmTypeName = struct _UNICODE_STRING [42]
global 806c9e68    0 nt!CmTypeName = struct _UNICODE_STRING []
global 805bd7b0    0 nt!CmTypeString = unsigned short *[]
global 805bd7b0   a8 nt!CmTypeString = unsigned short *[42]
```

В предыдущем примере размер задается в шестнадцатеричном формате, а тип данных — в десятичном формате. Таким образом, в последней строке предыдущего примера тип данных представляет собой массив из 42 указателей на неподписанные короткие целые числа. Размер этого массива составляет 42*4 = 168, а 168 отображается в шестнадцатеричном формате как 0xA8.

Параметр **/s***Size* можно использовать для отображения только тех символов, размер которых в байтах является определенным значением. Например, команду в предыдущем примере можно ограничить символами, представляющими объекты, размер которых 0xA8.

dbgcmd

```
kd> x /v /s a8 nt!CmType*
global 805bd7b0   a8 nt!CmTypeString = unsigned short *[42]
```

### Работа с типами данных

Параметр **/t** приводит к тому, что отладчик отображает сведения о типе данных каждого символа. Обратите внимание, что для многих символов эти сведения отображаются даже без параметра **/t** . При использовании **параметра /t** сведения о типе данных для таких символов отображаются дважды.

dbgcmd

```
0:001> x prymes!__n*
00427d84 myModule!__nullstring = 0x00425de8 "(null)"
0042a3c0 myModule!_nstream = 512
Type information missing error for _nh_malloc
004021c1 myModule!MyStructInstance = struct MyStruct
00427d14 myModule!_NLG_Destination = <no type information>

0:001> x /t prymes!__n*
00427d84 char * myModule!__nullstring = 0x00425de8 "(null)"
0042a3c0 int myModule!_nstream = 512
Type information missing error for _nh_malloc
004021c1 struct MyStruct myModule!MyStructInstance = struct MyStruct
00427d14 <NoType> myModule!_NLG_Destination = <no type information>
```

Команда x отобразит экземпляр типа .

dbgcmd

```
0:001> x foo!MyClassInstance
00f4f354          foo!MyClassInstance = 0x00f78768
```

Команда x не отображает ничего на основе только имени типа.

dbgcmd

```
0:001> x foo!MyClass
0:001>
```

Для отображения сведений о типе с помощью имени типа рекомендуется использовать **dt (тип отображения),** который предоставляет сведения как для типов, так и для экземпляров типов:

dbgcmd

```
0:001> dt foo!MyClass
   +0x000 IntMemberVariable : Int4B
   +0x004 FloatMemberVariable : Float
   +0x008 BoolMemberVariable : Bool
   +0x00c PtrMemberVariable : Ptr32 MyClass
```

### Работа с шаблонами

Вы можете использовать подстановочные знаки с командой x для отображения классов шаблонов, как показано в этом примере.

dbgcmd

```
0:001>  x Fabric!Common::ConfigEntry*TimeSpan?
000007f6`466a2f9c
Fabric!Common::ConfigEntry<Common::TimeSpan>::ConfigEntry<Common::TimeSpan>
(void)
000007f6`466a3020
Fabric!Common::ConfigEntry<Common::TimeSpan>::~ConfigEntry<Common::TimeSpan>
(void)
```

При работе с шаблонами рекомендуется использовать команду **dt (тип отображения** ), так как команда x не отображает отдельные элементы класса шаблона.

dbgcmd

```
0:001> dt foo!Common::ConfigEntry<Common::TimeSpan>
   +0x000 __VFN_table : Ptr64
   +0x008 componentConfig_ : Ptr64 Common::ComponentConfig
   +0x010 section_         :
std::basic_string<wchar_t,std::char_traits<wchar_t>,std::allocator<wchar_t>
>
   +0x038 key_             :
std::basic_string<wchar_t,std::char_traits<wchar_t>,std::allocator<wchar_t>
>
```

# См. также

Проверка символов

**dv (отображение локальных переменных)**

# z (выполнение в то время как)

Статья • 08.07.2024

Команда **z** выполняет команду, пока данное условие имеет значение true.

Режим пользователя

```
dbgcmd
```

```
Command ; z( Expression )
```

Режим ядра

```
dbgcmd
```

```
Command ; [Processor] z( Expression )
```

# Параметры

*Команда*
Указывает команду, выполняемую во время *вычисления условия выражения* , ненулевого значения. Эта команда всегда выполняется по крайней мере один раз.

*Процессор*
Указывает процессор, который применяется к тесту. Дополнительные сведения о синтаксисе см. в разделе "Многопроцессорный синтаксис". Вы можете указать процессоры только в режиме ядра.

*Выражение*
Указывает условие для тестирования. Если это условие оценивается как ненулевое значение, команда выполняется снова, а затем выражение проверяется снова. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис числовых выражений".

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |

| Позиция | Description |
| --- | --- |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Во многих командах отладчика точка с запятой используется для разделения несвязанных команд. Однако в команде **z** точка с запятой отделяет "z" от *параметра Command* .

Команда всегда выполняется по крайней мере один раз, а затем *проверяется выражение*. Если условие является ненулевое значение, команда снова выполняется, а затем выражение проверяется снова. (Это поведение похоже на выполнение C-языка **— в то время** как цикл, а не простой цикл.)

Если имеется несколько запятых слева от "z", все команды слева от "z" повторяются до тех пор, пока *условие выражения* имеет значение true. Такие команды могут быть любыми командами отладчика, разрешающими точку с запятой терминала.

Если добавить еще одну точку с запятой и дополнительные команды после **команды z** , эти дополнительные команды выполняются после завершения цикла. Обычно не рекомендуется использовать строку, которая начинается с "z", так как она создает неинтересные выходные данные навсегда, если условие не становится ложным из-за какого-либо другого действия. Обратите внимание, что можно вложить **команды z** .

Чтобы разорвать цикл, продолжающийся слишком долго, используйте CTRL+C в CDB или KD **или используйте отладку | Перерыв** из меню или CTRL+BREAK в WinDbg.

В следующем примере кода показан ненужный сложный способ ноль регистра **eax** .

dbgcmd

```
0:000> reax = eax - 1 ; z(eax)
```

В следующем примере увеличивается **число регистров eax** и **ebx** до тех пор, пока одно из них не равно 8, а затем увеличивает **регистр ecx** один раз.

dbgcmd

```
0:000> reax=eax+1; rebx=ebx+1; z((eax<8)|(ebx<8)); recx=ecx+1
```

В следующем примере используется синтаксис выражения C++ и в качестве переменной цикла используется псевдорегистрирующий **$t 0** .

dbgcmd

```
0:000> .expr /s c++
Current expression evaluator: C++ - C++ source expressions

0:000> db pindexcreate[@$t0].szKey; r$t0=@t0+1; z( @$t0 < cIndexCreate )
```

## См. также

**j (Execute If-Else)**

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Meta-Commands

18.07.2025

This section of the reference discusses the various debugger *meta-commands* that can be used in CDB, KD, and WinDbg. Эти команды предшествуют периоду (.).

# .abandon (Abandon Process)

Статья • 26.10.2023

Команда **.abandon** завершает сеанс отладки, но оставляет целевое приложение в состоянии отладки. При этом отладчик возвращается в неактивный режим.

dbgcmd

```
.abandon [/h|/n]
```

## Параметры

**/H**

Все невыполненные события отладки будут продолжены и помечены как обрабатываемые. Это значение по умолчанию.

**/N**

Все невыполненные события отладки будут продолжаться без обработки.

## Среда

Эта команда поддерживается только в Windows XP и более поздних версиях Windows.

| Элемент | Описание |
|---|---|
| Режимы | Только в пользовательском режиме |
| Цели | Только динамическая отладка |
| Платформы | все |

## Дополнительные сведения

Если целевой объект остается в состоянии отладки, к нему можно подключить новый отладчик. Дополнительные сведения см. в разделе Повторное присоединение к целевому приложению . Однако после того, как процесс был прерван один раз, он никогда не может быть восстановлен до запущенного состояния без подключенного отладчика.

# .allow_exec_cmds (разрешить команды выполнения)

Статья • 08.07.2024

Команда **.allow_exec_cmds** определяет, можно ли использовать команды выполнения.

dbgcmd

```
.allow_exec_cmds 0
.allow_exec_cmds 1
.allow_exec_cmds
```

## Параметры

**0**
Запрещает использование команд выполнения.

**1**
Позволяет использовать команды выполнения.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | режим пользователя и режим ядра |
| **Целевые объекты** | только динамическая отладка |
| **Платформы** | all |

## Дополнительная информация

Полный список команд выполнения см. в разделе "Управление целевым объектом".

## Замечания

Без параметров **.allow_exec_cmds** будет отображаться, разрешены ли команды выполнения в настоящее время.

Команды выполнения включают g (Go),t (Trace),**p (Step), а также любые другие команды или действие графического интерфейса WinDbg, которое приведет к выполнению целевого объекта.**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .allow_image_mapping (разрешить сопоставление изображений)

Статья • 08.07.2024

Команда **.allow_image_mapping** определяет, будут ли сопоставлены файлы изображений.

dbgcmd

```
.allow_image_mapping [/r] 0
.allow_image_mapping [/r] 1
.allow_image_mapping
```

## Параметры

**/r**
Перезагрузит все модули в списке модулей отладчика. Это эквивалентно перезагрузке **/d**.

**0**
Запрещает сопоставление файлов изображений.

**1**
Позволяет сопоставлять файлы изображений.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | режим пользователя и режим ядра |
| **Целевые объекты** | live, аварийное дампа |
| **Платформы** | all |

## Замечания

Без параметров .allow_image_mapping **будет отображаться,** разрешено ли сопоставление файлов изображений. По умолчанию это сопоставление разрешено.

Сопоставление изображений наиболее распространено при отладке мини-dump. Сопоставление изображений также может произойти, если DbgHelp не может получить доступ к записям отладки (например, во время отладки ядра при выгружаемой памяти).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .apply_dbp (применение точки останова данных к контексту)

Статья • 08.07.2024

Команда **.apply_dbp** применяет существующие точки останова данных текущего процесса к указанному контексту регистра.

```
dbgcmd

    .apply_dbp [/m Context]
```

## Параметры

**/m** *Контекст*
Указывает адрес контекста регистра (структура CONTEXT) в памяти, к которой применяются точки останова данных текущего процесса.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | режим пользователя и режим ядра |
| **Целевые объекты** | только для целевого объекта live |
| **Платформы** | all |

## Дополнительная информация

Дополнительные сведения о точках останова, контролируемых процессором, см. в разделе "Точки останова процессора" (точки останова ba). Дополнительные сведения о контексте регистрации (контексте потока) см. в разделе "Регистрация контекста".

## Замечания

Точки останова, контролируемые обработчиком, называются *точками* останова данных или *точками останова* процессора. Эти точки останова создаются

командой **ba (break on Access).**

Эти точки останова связаны с расположением памяти в адресном пространстве определенного процесса. Команда **.apply_dbp** изменяет указанный контекст регистра, чтобы эти точки останова данных были активными при использовании этого контекста.

**Если параметр /m** *Address* не используется, точки останова данных будут применены к текущему контексту регистра.

Эта команда может использоваться только в том случае, если целевой объект находится в собственном режиме компьютера. Например, если целевой объект работает на 64-разрядном компьютере, эмулируя процессор x86 с помощью *WOW64*, эту команду нельзя использовать.

Один из примеров времени, когда эта команда полезна, заключается в том, что вы находитесь в фильтре исключений. Команда **.apply_dbp** может обновить сохраненный контекст фильтра исключений. Затем точки останова данных будут применены при выходе фильтра исключений и возобновлении хранимого контекста. Без такого изменения возможно, что точки останова данных будут потеряны.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .asm (изменение параметров дизассемблы)

Статья • 08.07.2024

Команда **ASM** управляет отображением кода дизассемблы.

dbgcmd

```
.asm
.asm[-] Options
```

## Параметры

-

Приводит к отключению указанных параметров. Если знак минуса не используется, указанные параметры будут включены.

*Параметры* могут быть любым количеством следующих параметров:

**ignore_output_width**
Запрещает отладчику проверять ширину строк в отображении дизассемблы.

**no_code_bytes**
(только целевые объекты x86 и x64) Подавляет отображение необработанных байтов.

**source_line**
Префиксирует каждую строку дизассемблы с номером строки исходного кода.

**многословный**
(Только целевой объект Itanium) Приводит к отображению сведений о типе пакета вместе со стандартными сведениями об дизассемблы.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |

| Позиция | Description |
| --- | --- |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Описание отладки сборки и связанных команд см. в разделе "Отладка в режиме сборки".

## Замечания

При использовании **asm** отображается текущее состояние параметров.

Эта команда влияет на отображение всех инструкций дизассемблии в окне команды отладчика. В WinDbg он также изменяет содержимое окна Дизассемблии.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .attach (присоединение к процессу)

Статья • 08.07.2024

Команда **.attach** присоединяется к новому целевому приложению.

```
dbgcmd
```

```
.attach [-premote RemoteOptions] AttachOptions PID
```

## Параметры

*RemoteOptions*
Указывает сервер обработки для подключения. Параметры совпадают с параметрами командной строки **-premote** . Дополнительные сведения о синтаксисе см. в статье "Активация смарт-клиента".

*AttachOptions*
Указывает, как будет выполнено подключение. Это может быть любой из следующих параметров:

**-b**
Запрещает отладчику запрашивать начальный разрыв при присоединении к целевому процессу. Это может быть полезно, если приложение уже приостановлено или если вы хотите избежать создания потока останова в целевом объекте.

**-e**
Позволяет отладчику подключаться к процессу, который уже отлаживается. Дополнительные сведения см. в статье "Повторное присоединение к целевому приложению ".

**-k**
Начинает сеанс отладки локального ядра. Дополнительные сведения см. в разделе "Выполнение локальной отладки ядра".

**-f**
Зависает все потоки во всех целевых приложениях, за исключением нового целевого объекта. Эти потоки будут оставаться замороженными до тех пор, пока исключение не возникнет в недавно присоединенном процессе. Обратите внимание, что начальная точка останова квалифицируется как исключение. Отдельные потоки можно отменить с помощью **команды ~u (Unfreeze Thread).**

**-r**

Вызывает отладчик запустить целевой процесс, запущенный при подключении к нему. Это может быть полезно, если приложение уже приостановлено, и вы хотите возобновить выполнение.

**-v**
Приводит к тому, что указанный процесс будет отлаживаться невиновно.

*PiD*
Указывает идентификатор процесса нового целевого приложения.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Замечания

Эту команду можно использовать, если CDB неактивна или если она уже отлаживать один или несколько процессов. Его нельзя использовать, если WinDbg неактивен.

Если эта команда выполнена успешно, отладчик прикрепится к указанному процессу при следующем выполнении команды отладчика. Если эта команда используется несколько раз в строке, выполнение должно запрашиваться столько раз, сколько было использовано этой командой.

Так как выполнение не допускается во время неисправной отладки, отладчик не может выполнять неисправную отладку нескольких процессов одновременно. Это также означает, что использование **команды .attach -v** может привести к снижению полезности уже существующего сеанса инвазивной отладки.

Несколько целевых процессов всегда будут выполняться вместе, если некоторые из их потоков не заморожены или приостановлены.

Если вы хотите присоединиться к новому процессу и заморозить все существующие целевые объекты, используйте **параметр -f** . Например, вы можете отладить сбой в клиентском приложении и присоединиться к процессу сервера, не позволяя клиентскому приложению продолжать работу.

**Если используется параметр -premote**, новый процесс будет частью новой системы. Дополнительные сведения см. в разделе "Отладка нескольких целевых объектов".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .beep (звуковой сигнал говорящего)

Статья • 11.11.2023

Команда **.beep** создает шум на динамике компьютера.

```
dbgcmd

    .beep
```

## Среда

Эту команду нельзя использовать в файлах скриптов.

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# .bpcmds (отображение команд точки останова)

Статья • 26.10.2023

Команда **.bpcmds** отображает команды, которые использовались для задания каждой из текущих точек останова.

```
dbgcmd
```

```
.bpcmds
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения и примеры использования точек останова, других команд и методов управления точками останова см. в статье Использование точек останова.

# Комментарии

Если неясно, задана ли определенная точка останова в адресе, в символьной ссылке или символе, используйте команду **.bpcmds** , чтобы определить, какая команда точки останова использовалась для ее создания. Команда, которая использовалась для создания точки останова, определяет ее характер:

- Команда **bp (Set Breakpoint)** задает точку останова по адресу.
- Команда **bu (Задать неразрешенную точку останова)** задает точку останова для символьной ссылки.

- Команда **bm (Set Symbol Breakpoint)** задает точку останова для символов, соответствующих указанному шаблону. Если параметр **/d** включен, он создает ноль или более точек останова для адресов (например, bp), в противном случае он создает ноль или более точек останова для символических ссылок (например, bu).

- Команда **ba (Break on Access)** задает точку останова данных по адресу.

Выходные данные **.bpcmds** отражают текущую природу каждой точки останова. Каждая строка отображения **.bpcmds** начинается с команды, используемой для ее создания (**bp**, **bu** или **ba**), за которой следует идентификатор точки останова, а затем расположение точки останова.

Если точка останова была создана **ba**, также отображаются тип доступа и размер.

Если точка останова была создана **bm** без переключателя **/d** , отображается тип точки останова **bu,** за которой следует вычисляемый символ, заключенный в маркер **@!""** (который указывает, что это литеральный символ, а не числовое выражение или регистр). Если точка останова была создана **bm** с параметром **/d** , то на экране отображается тип точки останова как **bp**.

Вот пример:

dbgcmd

```
0:000> bp notepad!winmain

0:000> .bpcmds
bp0 0x00000001`00003340 ;

0:000> bu myprog!winmain
breakpoint 0 redefined

0:000> .bpcmds
bu0 notepad!winmain;

0:000> bu myprog!LoadFile

0:000> bp myprog!LoadFile+10

0:000> bm myprog!openf*
  3: 00421200 @!"myprog!openFile"
  4: 00427800 @!"myprog!openFilter"

0:000> bm /d myprog!closef*
  5: 00421600 @!"myprog!closeFile"

0:000> ba r2 myprog!LoadFile+2E
```

```
0:000> .bpcmds
bu0 notepad!winmain;
bu1 notepad!LoadFile;
bp2 0x0042cc10 ;
bu3 @!"myprog!openFile";
bu4 @!"myprog!openFilter";
bp5 0x00421600 ;
ba6 r2 0x0042cc2e ;
```

В этом примере обратите внимание, что выходные данные **bpcmds** начинаются с соответствующей команды ("bu", "bp" или "ba"), за которой следует номер точки останова (без промежуточного пространства).

Обратите внимание, что, поскольку число точки останова 0 изначально было задано с использованием **bp**, а затем было переопределено с помощью **bu**, на экране отображается тип bu.

Обратите внимание, что точки останова 3, 4 и 5, созданные командами **bm** , показанными в этом примере, отображаются как тип "bp" или "bu", в зависимости от того, был ли включен параметр **/d** при использовании **bm** .

# .bpsync (синхронизация потоков в точке останова)

Статья • 08.07.2024

Когда поток достигает точки останова, **команда BPSYNC** зависает все остальные потоки, пока поток, к которому применяется точка останова, перейдет через точку останова.

dbgcmd

```
.bpsync 1
.bpsync 0
.bpsync
```

## Параметры

**1**

Заставляет все потоки заморозить, когда один поток достиг точки останова. После того, как поток, к которому применяется точка останова, прошел через точку останова, другие потоки не раскроются.

**0**

Позволяет другим потокам продолжать выполняться, когда один поток достиг точки останова. Это поведение принимается по умолчанию.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только в пользовательском режиме |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Без параметров. Команда BPSYNC **отображает текущее** правило, которое управляет поведением синхронизации точек останова.

Команда **BPSYNC** применяется как к точкам останова программного обеспечения (задано с **bp**, **bu** или **bm**), так и к точкам останова процессора (набор с **ba**).

Если существует возможность выполнения нескольких потоков в одном коде, **команда BPsync 1** может быть полезна для записи всех вхождений точек останова. Без этой команды может быть пропущено вхождение точки останова, так как первый поток для достижения точки останова всегда приводит к временному удалению точки останова отладчику. В короткий период, когда точка останова удаляется, другие потоки могут достичь того же места в коде, а не активировать точку останова, как это предполагается.

Временное удаление точек останова — это обычный аспект операции отладчика. Когда целевой объект достигает точки останова и возобновляется, отладчик должен временно удалить точку останова, чтобы целевой объект смог выполнить реальный код. После выполнения реальной инструкции отладчик повторно встраивает разрыв. Для этого отладчик восстанавливает код (или отключает разрывы данных), выполняет одношаговые действия, а затем возвращает разрыв.

Обратите внимание, что при использовании **BPSYNC 1** существует риск взаимоблокировок между потоками, которые были заморожены.

---


## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .breakin (break to the Kernel Debugger)

Статья • 26.10.2023

Команда **.breakin** переключается с отладки в пользовательском режиме на отладку в режиме ядра. Эта команда особенно полезна при управлении отладчиком пользовательского режима из отладчика ядра.

dbgcmd

```
.breakin
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Только в пользовательском режиме |
| Цели | Только динамическая отладка |
| Платформы | все |

# Комментарии

Если отладка в режиме ядра была включена во время загрузки и вы используете отладчик пользовательского режима, можно использовать команду **.breakin** , чтобы остановить операционную систему и передать управление отладчику ядра.

Команда **.breakin** вызывает разрыв в режиме ядра в контексте процесса отладчика. Если подключен отладчик ядра, он станет активным. Контексту процесса отладчика ядра автоматически присваивается процесс отладчика пользовательского режима, а не целевой процесс отладчика пользовательского режима.

Эта команда в первую очередь полезна, если для отладки проблемы в пользовательском режиме требуется получить сведения о состоянии ядра системы. Возобновление выполнения в отладчике ядра необходимо для продолжения сеанса отладки в пользовательском режиме.

Если вы управляете отладчиком пользовательского режима из отладчика ядра , а запрос отладчика пользовательского режима отображается в отладчике ядра, эта команда приостановит отладчик пользовательского режима и отобразит запрос отладки в режиме ядра.

Если системе не удается войти в отладчик ядра, отображается сообщение об ошибке.

Эта команда также полезна, если вы используете отладчик ядра, чтобы задать точку останова в пространстве пользователя, и эта точка останова перехватывалась отладчиком пользовательского режима, а не отладчиком ядра. Выполнение этой команды в отладчике пользовательского режима передаст управление отладчику ядра.

Если команда **.breakin** используется в системе, которая не была загружена с включенной отладкой, она не оказывает никакого влияния.

# .browse (Команда отображения в браузере)

Статья • 26.10.2023

Команда **.browse** отображает выходные данные указанной команды в новом окне командного браузера.

```
dbgcmd
```

```
.browse Command
```

## Параметры

*Команды*
Команда, выполняемая и отображаемая в новом окне командного браузера.

## Комментарии

В следующем примере команда **.browse** используется для отображения выходных данных команды **.chain /D** в окне обозревателя команд.

```
dbgcmd
```

```
.browse .chain /D
```

# .bug проверка (отображение данных проверки ошибок)

Статья • 05.04.2024

Команда **.bug проверка** отображает данные из проверка ошибки на целевом компьютере.

```
dbgsyntax
.bugcheck
```

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения об ошибках проверка см. в разделе "Проверки ошибок" (синие экраны). Описание отдельных проверка ошибок см. в разделе справочника по коду проверки ошибок.

## Замечания

Эта команда отображает текущую ошибку проверка данных. (Эта ошибка проверка данные будут доступны до перезагрузки компьютера.

Кроме того, можно отобразить данные об ошибке проверка в 32-разрядных системах с помощью **dd NT! KiBugCheckData L5** или в 64-разрядных системах с помощью **dq NT! KiBugCheckData L5**. Однако команда .bug проверка является более надежной, так как она работает в некоторых сценариях, что команда d*

(отображаемая память) **не будет (например,** мини-ddumps в режиме пользователя).

Команда **!analyze** extension также полезна после возникновения ошибки проверка.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .cache (задать размер кэша)

Статья • 08.07.2024

Команда **.cache** задает размер кэша, используемого для хранения данных, полученных из целевого объекта. Также задает ряд параметров кэша и памяти.

dbgsyntax

```
.cache Size
.cache Option
.cache
```

# Параметры

*Размер*
Размер кэша отладки ядра в килобайтах. Если *размер* равен нулю, кэш отключен. Выходные данные команды отображают размер кэша в байтах. (Размер по умолчанию — 1000 КБ.)

*Выбор*
Может быть одним из следующих вариантов:

**держать**
Автоматическая очистка кэша отключена.

**отмена удержания**
Отключает **параметр удержания** . (Это параметр по умолчанию.)

**декодепты**
Все записи таблицы страниц перехода (PTEs) будут неявно декодированы. (Это параметр по умолчанию.)

**nodecodeptes**
Отключает **параметр декодептов** .

**принудительные кодпты**
Все виртуальные адреса будут преобразованы в физические адреса перед доступом. Этот параметр также приводит к отключению кэша. Если вы не беспокоитесь с памятью в режиме ядра, вместо этого эффективнее использовать **принудительное использование codeuser** .

**forcedecodeuser**
Все виртуальные адреса в режиме пользователя будут преобразованы в физические адреса перед доступом. Этот параметр также приводит к отключению кэша.

**Обратите внимание**, что перед использованием **thread (Set Register Context), .context (Set Register Context), .context (Set User-Mode Address Context), .process (Set Process Context) (Set Process Context) (Set Process Context) (Set ProcessCodeptes) (Set ProcessCodeptes) (Set Process Context) (Set ProcessCodeptes) (Set ProcessCodeptes) (Set ProcessCodeptes)** (Set Если вы используете параметр /p **с потоком и процессом**, **параметр принудительного кода** будет автоматически задан. В любом другом случае необходимо явно использовать **команду .cache forcedecodeuser** .

**noforcedecodeptes**
Отключает параметры принудительного **кода и принудительного кода** . (Это параметр по умолчанию.)

**flushall**
Удаляет весь кэш виртуальной памяти.

**flushu**
Удаляет все записи диапазонов с ошибками из кэша, а также все записи в режиме пользователя.

**Адрес очистки**
Удаляет блок 4096-байтов кэша, начиная с *адреса*.

# Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | только динамическая отладка |
| **Платформы** | all |

# Замечания

Если **кэш** используется без аргументов, отображается текущий размер, состояние и параметры кэша.

Параметр **.cache forcedecodeuser** или **.cache forcedecodeptes** будет длиться только до тех пор, пока отладчик остается разбитым на целевой компьютер. Если выполняется любой шаг или выполнение целевого объекта, **состояние noforcedecodeptes** снова вступает в силу. Это запрещает отладчику вмешиваться в выполнение или перезагрузку непродуктивным способом.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .call (функция вызова)

Статья • 08.07.2024

Команда **вызова** вызывает целевой процесс для выполнения функции.

```dbgsyntax
.call [/v] Function( Arguments )
.call /s Prototype Function( Arguments )
.call /c
.call /C
```

## Параметры

**/v**
Подробные сведения о вызове и его аргументах отображаются.

**/s** *Прототип*
Позволяет вызывать функцию, указанную *функцией,* даже если у вас нет правильных символов. В этом случае необходимо иметь символы для другой функции, которая имеет тот же прототип вызова, что и функция, которую вы пытаетесь вызвать. Параметр *Прототипа* — это имя этой функции прототипа.

*Функция*
Указывает вызываемую функцию. Это может быть имя функции (желательно квалифицировать с именем модуля) или любое другое выражение, которое оценивается по адресу функции. Если необходимо вызвать конструктор или деструктор, необходимо указать адрес или использовать выражение C++ для оценки именованного синтаксиса операторов (см . сведения о синтаксисе числовых выражений).

*Аргументы*
Задает аргументы, переданные функции. Если вы вызываете метод, первый аргумент должен быть **таким**, и все остальные аргументы следуют ему. Аргументы должны быть разделены запятыми и должны соответствовать обычному синтаксису аргументов. Поддерживаются списки аргументов переменной длины. Выражения в аргументе анализируются вычислителем выражений C++; Дополнительные сведения см. в разделе "Номера И операторы C++". Вы не можете ввести литеральную строку в качестве аргумента, но можно использовать указатель на строку или любую другую память, доступную для целевого процесса.

**/c**
Очищает существующий вызов текущего потока.

**/C**
Очищает любой существующий вызов текущего потока и сбрасывает контекст текущего потока в контекст, хранящийся невыполненным вызовом.

# Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | только в пользовательском режиме |
| **Целевые объекты** | только динамическая отладка |
| **Платформы** | Только x86 и x64 |

# Замечания

Указанная функция вызывается текущим потоком текущего процесса.

**Поддерживаются только соглашения о вызовах cdecl**, **stdcall, fastcall и thiscall**. Управляемый код не может вызываться этой командой.

После **использования вызова** отладчик обновит стек, измените указатель инструкции, чтобы указать начало вызываемой функции, а затем остановиться. Используйте g (Go) для возобновления выполнения или **~. g** для выполнения только потока, выполняющего вызов.

Когда функция возвращается, происходит разрыв, а отладчик отображает возвращаемое значение функции. Возвращаемое значение также хранится в псевдорегистрируемом **$callret** , который получает тип возвращаемого значения.

Если вы врезались в целевой объект с помощью CTRL+C или CTRL+BREAK, текущий поток является дополнительным потоком, созданным для обработки разрыва. При возникновении **команды вызова .call** на этом этапе для вызываемой функции будет использоваться дополнительный поток.

Если вы достигли предопределенной точки останова, нет дополнительного потока останова. При использовании **вызова** в точке останова в пользовательском режиме можно использовать g для выполнения всего процесса или **~. g** для выполнения только текущего потока. Использование **g** может исказить поведение программы, так как вы взяли один поток и перенаправили его на эту новую

функцию. С другой стороны, этот поток по-прежнему будет иметь свои блокировки и другие атрибуты, и, следовательно **, ~. g** может рисковать взаимоблокировками.

Самый безопасный способ использования **вызова** .— задать точку останова в коде в расположении, где можно безопасно вызвать определенную функцию. При попадании в эту точку останова можно использовать **вызов .call** , если требуется, чтобы эта функция выполнялась. Если вы используете **вызов .в точке, где эта функция не может быть вызвана обычно, взаимоблокировка** или повреждение целевого объекта может привести к повреждению.

Это может быть полезно для добавления дополнительных функций в исходный код, которые не вызываются существующим кодом, но предназначены для вызова отладчиком. Например, можно добавить функции, которые используются для изучения текущего состояния кода и его среды и хранения сведений о состоянии в известном расположении памяти. Не оптимизируйте код или эти функции могут быть удалены компилятором. Используйте этот метод только в качестве последнего средства, так как при отладке файла дампа приложение завершит сбой **вызова** .

**Команды .call /c** и **.call /C** следует использовать только в том случае, если попытка использования **вызова** завершилась ошибкой или если вы изменили свое мнение перед вводом **команды g**. Они не должны использоваться случайно, так как отказ от незавершенного вызова может привести к поврежденном целевому состоянию.

В следующем примере кода показано, как **используется команда .call /s** .

```
dbgcmd
```

```
.call /s KnownFunction UnknownFunction( 1 )
```

В этом примере есть частные символы для **KnownFunction**, которые принимают целое число в качестве единственного аргумента и возвращают, например указатель на массив. У вас нет символов или, возможно, у вас есть только открытые символы для **UnknownFunction**, но вы знаете, что он принимает целое число в качестве единственного аргумента и возвращает указатель на массив. С помощью **параметра /s** можно указать, что **UnknownFunction будет работать таким же образом, как и Функция KnownFunction**. Таким образом, можно успешно создать вызов **UnknownFunction**.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .chain (перечисление расширений отладчика)

Статья • 26.10.2023

Команда **.chain** выводит список всех загруженных расширений отладчика в порядке поиска по умолчанию.

dbgsyntax

```
.chain
.chain /D
```

## Параметры

**/D**

Отображает выходные данные с помощью языка разметки отладчика. В выходных данных каждый указанный модуль представляет собой ссылку, которую можно щелкнуть, чтобы получить сведения о расширениях, реализованных модулем.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения о загрузке, выгрузке и управлении расширениями см. в разделе Загрузка библиотек DLL расширения отладчика. Дополнительные сведения о выполнении команд расширения и описание порядка поиска по умолчанию см. в разделе Использование команд расширения отладчика.

# .childdbg (отладка дочерних процессов)

Статья • 08.07.2024

Команда **.childdbg** управляет отладкой дочерних процессов.

```
dbgsyntax
```

```
.childdbg 0
.childdbg 1
.childdbg
```

## Параметры

**0**

Запрещает отладчику отлаживать дочерние процессы.

**1**

Приводит к отладке дочерних процессов отладчиком.

## Среда

Эта команда поддерживается только в Windows XP и более поздних версиях Windows.

Развернуть таблицу

| | |
|---|---|
| **Режимы** | только в пользовательском режиме |
| **Целевые объекты** | только динамическая отладка |
| **Платформы** | x86 и x64 |

## Замечания

**Дочерние процессы — это дополнительные процессы** , запускаемые исходным целевым приложением.

Без параметров **.childdbg** будет отображать текущее состояние отладки дочернего процесса.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .clients (перечисление клиентов отладки)

Статья • 26.10.2023

Команда **.clients** выводит список всех клиентов отладки, подключенных к сеансу отладки.

```
dbgsyntax
    .clients
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения и другие команды, которые можно использовать при удаленной отладке с помощью отладчика, см. в разделе Управление сеансом удаленной отладки.

# .closehandle (Close Handle)

Статья • 08.07.2024

Команда **closehandle** закрывает дескриптор, принадлежащий целевому приложению.

dbgsyntax

```
.closehandle Handle
.closehandle -a
```

## Параметры

*Ручка*
Указывает дескриптор, который должен быть закрыт.

**-a**
Приводит к закрытию всех дескрипторов, принадлежащих целевому приложению.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Замечания

Расширение !handle можно использовать для отображения существующих дескрипторов.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .cls (clear screen)

Статья • 26.10.2023

Команда **.cls** очищает окно команд отладчика.

```dbgsyntax
    .cls
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# .context (настройка контекста адреса в режиме пользователя)

Статья • 05.04.2024

Команда **контекста .context** указывает, какой каталог страницы процесса будет использоваться для контекста адреса в пользовательском режиме или отображает текущий контекст адреса в режиме пользователя.

```
dbgsyntax
```

```
.context [PageDirectoryBase]
```

## Параметры

*PageDirectoryBase*
Задает базовый адрес для каталога страницы требуемого процесса. Контекст адреса в пользовательском режиме будет задан в этом каталоге страницы. Если *PageDirectoryBase* равно нулю, контекст адреса в пользовательском режиме будет установлен в каталог страниц для текущего состояния системы. Если *PageDirectoryBase* опущен, отображается текущий контекст адреса в пользовательском режиме.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения о контексте адреса в пользовательском режиме и других параметрах контекста см. в разделе "Изменение контекстов".

# Замечания

Как правило, при отладке ядра только видимое адресное пространство в пользовательском режиме связано с текущим процессом.

Команда **.context** указывает отладчику ядра использовать указанный каталог страницы в качестве *контекста* адреса в режиме пользователя. После выполнения этой команды отладчик получит доступ к этому виртуальному адресное пространство. Таблицы страниц для этого адресного пространства будут использоваться для интерпретации всех адресов памяти в режиме пользователя. Это позволяет читать и записывать в эту память.

Команда .process (Set Process Context) имеет аналогичный эффект. **Однако команда .context** задает контекст адреса в пользовательском режиме для определенного каталога страницы, а **команда .process** задает *контекст* процесса определенному процессу. На процессоре x86 эти две команды по сути имеют одинаковый эффект. Дополнительные сведения см. в контексте процесса.

Если выполняется динамическая отладка, в дополнение к команде .context следует выполнить команду .cache forcedecodeuser. Это заставляет отладчика искать физические адреса необходимого пространства памяти. (Это может быть медленно, так как часто это означает, что огромный объем данных должен передаваться по отладочному кабелю.)

Если выполняется отладка аварийного дампа, команда кэша не требуется. Однако у вас не будет доступа к каким-либо частям виртуального адресного пространства процесса пользовательского режима, которые были страницы на диск при сбое.

Ниже приведен пример. Используйте расширение !process, чтобы найти базу каталогов для требуемого процесса:

dbgcmd

```
kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS fe5039e0  SessionId: 0  Cid: 0008    Peb: 00000000  ParentCid: 0000
    DirBase: 00030000  ObjectTable: fe529b68  TableSize:  50.
    Image: System

...

PROCESS fe3c0d60  SessionId: 0  Cid: 0208    Peb: 7ffdf000  ParentCid: 00d4
 DirBase: 0011f000  ObjectTable: fe3d0f48  TableSize:  30.
    Image: regsvc.exe
```

Теперь используйте команду .context **с базой** каталога страницы.

dbgcmd

```
kd> .context 0011f000
```

Это позволяет изучить адресное пространство различными способами. Например, ниже приведены выходные данные **расширения !peb** :

dbgcmd

```
kd> !peb
PEB at 7FFDF000
    InheritedAddressSpace:    No
    ReadImageFileExecOptions: No
    BeingDebugged:            No
    ImageBaseAddress:         01000000
    Ldr.Initialized: Yes
    Ldr.InInitializationOrderModuleList: 71f40 . 77f68
    Ldr.InLoadOrderModuleList: 71ec0 . 77f58
    Ldr.InMemoryOrderModuleList: 71ec8 . 77f60
        01000000 C:\WINNT\system32\regsvc.exe
        77F80000 C:\WINNT\System32\ntdll.dll
        77DB0000 C:\WINNT\system32\ADVAPI32.dll
        77E80000 C:\WINNT\system32\KERNEL32.DLL
        77D40000 C:\WINNT\system32\RPCRT4.DLL
        77BE0000 C:\WINNT\system32\secur32.dll
    SubSystemData:     0
    ProcessHeap:       70000
    ProcessParameters: 20000
        WindowTitle: "'C:\WINNT\system32\regsvc.exe'"
        ImageFile:    'C:\WINNT\system32\regsvc.exe'
        CommandLine:  'C:\WINNT\system32\regsvc.exe'
        DllPath:
'C:\WINNT\system32;.;C:\WINNT\System32;C:\WINNT\system;C:\WINNT;C:\WINNT\sys
tem32;C:\WINNT;C:\WINNT\System32\Wbem;C:\PROGRA~1\COMMON~1\AUTODE~1'
        Environment:  0x10000
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .copysym (копирование файлов символов)

Статья • 08.07.2024

Команда **COPYSYM** копирует текущие файлы символов в указанный каталог.

```
dbgsyntax
    .copysym [/l] Path
```

## Параметры

**/l**
Вызывает загрузку каждого файла символов при копировании.

*Путь*
Указывает каталог, в который должны быть скопированы файлы символов. Копии не перезаписывают существующие файлы.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Во многих случаях символы хранятся в сети. Доступ к символам часто может быть медленным или может потребоваться перенести сеанс отладки на другой компьютер, где у вас больше нет сетевого доступа. В таких сценариях команда COPYSYM **может использоваться для копирования символов,** необходимых для сеанса, в локальный каталог.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .cordll (управление отладкой CLR)

Статья • 26.10.2023

Команда **cordll** управляет отладкой управляемого кода и средой CLR Microsoft .NET.

```
dbgsyntax
.cordll [Options]
```

## Параметры

*Параметры* Один или несколько из следующих вариантов:

**-l** (строчная буква L)
Загружает модули отладки среды CLR.

**-я** **** *Модуль* (прописная i)
Указывает имя или базовый адрес модуля CLR для отладки. Дополнительные сведения см. в подразделе "Примечания".

**-U**
Выгружает модули отладки СРЕДЫ CLR.

**-E**
Включает отладку в среде CLR.

**-D**
Отключает отладку среды CLR.

**-D**
Отключает отладку среды CLR и выгружает модули отладки CLR.

**-N**
Перезагружает модули отладки CLR.

**-Lp** **** *Путь*
Указывает путь к каталогу модулей отладки CLR.

**-Se**
Позволяет использовать короткое имя модуля отладки CLR, mscordacwks.dll.

**-Sd**
Отключает, используя короткое имя модуля отладки CLR, mscordacwks.dll. Вместо

этого отладчик использует длинное имя модуля отладки CLR mscordacwks_<spec>.dll. Отключение использования коротких имен позволяет избежать использования локальной среды CLR, если вас беспокоит несоответствие.

**-Ve**
Включает подробный режим для загрузки модуля CLR.

**-Vd**
Отключает подробный режим для загрузки модуля CLR.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

Для отладки управляемого приложения отладчик должен загрузить компонент доступа к данным (DAC), соответствующий загруженной среде CLR. Однако в некоторых случаях приложение загружает более одной среды CLR. В этом случае можно использовать параметр **I** , чтобы указать, какой DAC должен загрузить отладчик. Версия 2 среды CLR называется Mscorwks.dll, а версия 4 — Clr.dll. В следующем примере показано, как указать, что отладчик должен загружать приложение уровня данных для версии 2 (mscorwks).

```
dbgcmd
```

```
.cordll -I mscorwks -lp c:\dacFolder
```

Если опустить параметр **I** , отладчик по умолчанию использует версию 4. Например, следующие две команды эквивалентны.

```
dbgcmd
```

```
.cordll -lp c:\dacFolder
.cordll -I clr -lp c:\dacFolder
```

Sos.dll — это компонент, используемый для отладки управляемого кода. Текущая версия средств отладки для Windows не включает версию sos.dll. Сведения о том, как получить sos.dll, см. в статье *Получение расширения отладки SOS (sos.dll)*статьи Отладка управляемого кода с помощью отладчика Windows.

Команда **CORDLL** поддерживается в отладке в режиме ядра. Однако эта команда может не работать, если не будет выглядеть необходимая память.

## См. также раздел

Отладка управляемого кода с помощью отладчика Windows

Расширение отладки SOS

# .crash (принудительный сбой системы)

Статья • 26.10.2023

Команда **.crash** приводит к тому, что целевой компьютер выдает ошибку проверка.

```
dbgsyntax
.crash
```

## Среда

| | |
|---|---|
| **Режимы** | Только в режиме ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | все |

## Дополнительные сведения

Обзор связанных команд и описание параметров, доступных после сбоя системы, см. в разделе Сбой и перезагрузка целевого компьютера.

# Комментарии

Эта команда немедленно приведет к аварийному завершению работы целевого компьютера.

Если вы уже находитесь в обработчике проверка ошибок, не используйте **.crash**. Вместо этого используйте **g (Go),** чтобы продолжить выполнение обработчика, который создаст аварийный дамп.

Файл дампа в режиме ядра будет записан, если включены аварийные дампы. Дополнительные сведения см. в статье Создание файла дампа Kernel-Mode .

# .create (Create Process)

Статья • 08.07.2024

Команда **create** создает новое целевое приложение.

```
dbgsyntax
```

```
.create [-premote RemoteOptions] [-f] CommandLine
```

## Параметры

*RemoteOptions*
Указывает сервер обработки, к которому необходимо подключиться. Параметры совпадают с параметрами командной строки **-premote** . Дополнительные сведения о синтаксисе см. в статье "Активация смарт-клиента".

**-f**
Зависает все потоки во всех целевых приложениях, за исключением создаваемого нового целевого объекта. Эти потоки будут оставаться замороженными до тех пор, пока исключение не возникнет в только что созданном процессе. Обратите внимание, что начальная точка останова квалифицируется как исключение. Отдельные потоки можно отменить с помощью **команды ~u (Unfreeze Thread).**

*CommandLine*
Указывает полную командную строку для нового процесса. *CommandLine* может содержать пробелы и не должен быть окружен кавычками. Весь текст после **команды CREATE** принимается в составе *CommandLine*. Эта команда не может быть выполнена с запятой и дополнительными командами отладчика.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Замечания

Эту команду можно использовать, если CDB неактивна или если она уже отлаживать один или несколько процессов. Его нельзя использовать, если WinDbg неактивен.

Если эта команда выполнена успешно, отладчик создаст указанный процесс при следующем выполнении команды отладчика. Если эта команда используется несколько раз в строке, выполнение должно запрашиваться столько раз, сколько было использовано этой командой.

Несколько целевых процессов всегда будут выполняться вместе, если некоторые из их потоков не заморожены или приостановлены.

Если вы хотите создать новый процесс и заморозить все существующие целевые объекты, используйте параметр -f.

**Если используется параметр -premote**, новый процесс будет частью новой системы. Дополнительные сведения см. в разделе "Отладка нескольких целевых объектов".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .createdir (Set Created Process Directory)

Статья • 08.07.2024

Команда **.createdir** управляет начальным каталогом и обрабатывает наследование для всех процессов, созданных отладчиком.

dbgsyntax

```
.createdir [-i | -I] [Path]
```

## Параметры

**-я**
Вызывает процессы, созданные отладчиком, наследовать дескриптор от отладчика. Это значение по умолчанию.

**-Я**
Запрещает процессам, созданным отладчиком, наследовать дескриптор от отладчика.

*Путь*
Указывает начальный каталог для всех дочерних процессов, созданных любым целевым процессом. Если *Путь* содержит пробелы, он должен быть заключен в кавычки.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Замечания

Если **файл .createdir** используется без параметров, отображается текущий начальный каталог и состояние наследования.

Если **файл .createdir** никогда не использовался, любой созданный процесс будет использовать свой обычный каталог по умолчанию в качестве начального каталога. Если вы уже установили путь с **.createdir и хотите вернуться к состоянию по умолчанию, используйте .createdir ""** без кавычек.

Параметр **.createir** влияет на все процессы, созданные с помощью .create (Create Process). Это также влияет на процессы, созданные файлом WinDbg | **Откройте команду меню исполняемого** файла, если только текстовое **поле "Пуск каталога** " не используется для переопределения этого параметра.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .cxr (отображаемая запись контекста)

Статья • 08.07.2024

Команда **CXR** отображает запись контекста, сохраненную по указанному адресу. Он также задает контекст регистрации.

```
dbgsyntax
```

```
.cxr [Options] [Address]
```

## Параметры

*Параметры* . Может быть любым сочетанием следующих параметров.

**/f** *Size*
Принудительно задает размер контекста *равным значению Size* в байтах. Это может быть полезно, если контекст не соответствует фактическому целевому объекту, например при использовании контекста x86 в 64-разрядном целевом объекте во время *отладки WOW64* . Если указан недопустимый или несогласованный размер, появится сообщение об ошибке "Не удается преобразовать контекст в каноническую форму".

**/w**
Записывает текущий контекст в память и отображает адрес места записи.

*Адрес* — адрес записи системного контекста (структура CONTEXT), который должен быть прочитан или записан. Адрес -1 приводит к чтению структуры CONTEXT из файла дампа. Это то же поведение, что и ecxr. Адрес 0 не поддерживается. При отладке файла дампа опустить адрес сбрасывает контекст регистра в контекст по умолчанию для этого потока (режим пользователя) или обработчика (режим ядра).

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |

| Позиция | Description |
| --- | --- |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения о контексте регистрации и других параметрах контекста см. в разделе "Изменение контекстов".

# Замечания

Сведения из записи контекста можно использовать для отладки системы остановки системы, в которой произошло необработанное исключение, и точную трассировку стека недоступна. Команда **CXR** отображает важные регистры для указанной записи контекста.

Эта команда также указывает отладчику использовать указанную запись контекста в качестве контекста регистрации. После выполнения этой команды отладчик получит доступ к наиболее важным регистрам и трассировке стека для этого потока. Этот контекст регистра сохраняется до тех пор, пока целевой объект не будет выполнять или использовать другую команду контекста регистра (.thread, **ECXR, .trap или CXR еще раз).** В пользовательском режиме он также будет сбрасываться при изменении текущего процесса или потока. Дополнительные сведения см. в разделе "Контекст регистрации".

Команда **CXR** часто используется для отладки проверки ошибок 0x1E. Дополнительные сведения и пример см. в разделе **"Проверка ошибок" 0x1E** (KMODE_EXCEPTION_NOT_HANDLED).

Команда **CXR /w** записывает контекст в память и отображает адрес, в котором он был сохранен. Этот адрес можно передать **в .apply_dbp (применить точку останова данных к контексту),** если необходимо применить точки останова данных к этому контексту.

# См. также

Изменение контекстов

Регистрация контекста

.exr (отображение записи исключений)

.ecxr

.ловушка

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .dbgdbg (отладка текущего отладчика)

Статья • 26.10.2023

Команда **.dbgdbg** запускает новый экземпляр CDB; этот новый отладчик принимает текущий отладчик в качестве целевого объекта.

```
dbgcmd
```

```
.dbgdbg
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Комментарии

Команда **.dbgdbg** аналогична клавише **управления CTRL+P (Отладчик текущего отладчика).** Однако **dbgdbg** является более универсальным, так как его можно использовать как из WinDbg, так и из KD и CDB, а также для отладки сервера отладки на удаленном компьютере.

# .detach (отсоединение от процесса)

Статья • 08.07.2024

Команда **.detach** завершает сеанс отладки, но оставляет любое целевое приложение в пользовательском режиме запущенным.

```
dbgcmd
```

```
.detach [ /h | /n ]
```

## Параметры

**/h**
Любое выдающееся событие отладки будет продолжено и отмечено как обработанное. Это значение по умолчанию.

**/n**
Любое выдающееся событие отладки будет продолжено без пометки как обработанное.

## Среда

Для отладки в реальном режиме пользователя эта команда поддерживается только в Windows XP и более поздних версиях Windows.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Эта команда завершит сеанс отладки в любом из следующих сценариев:

- При отладке файла дампа в режиме пользователя или в режиме ядра.

- При отладке целевого объекта в реальном режиме пользователя.
- При неисключательной отладке целевого объекта пользовательского режима.

Если выполняется отладка только одного целевого объекта, отладчик вернется в режим спячки после отсоединения.

При отладке нескольких целевых объектов сеанс отладки продолжится с оставшимися целевыми объектами.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .dml_flow (unasemmble with Links)

Статья • 14.11.2023

Команда **.dml_flow** отображает дизассемблированное блок кода и предоставляет ссылки, которые можно использовать для создания графа потока кода.

```
dbgcmd
```

```
.dml_flow Start Target
```

## Параметры

*Начать*
Адрес инструкции, с которой можно связаться с целевым адресом.

*Целевой*
Адрес в блоке кода для дизассемблировки.

## Комментарии

Рассмотрим стек вызовов, показанный в следующем примере.

```
dbgcmd
```

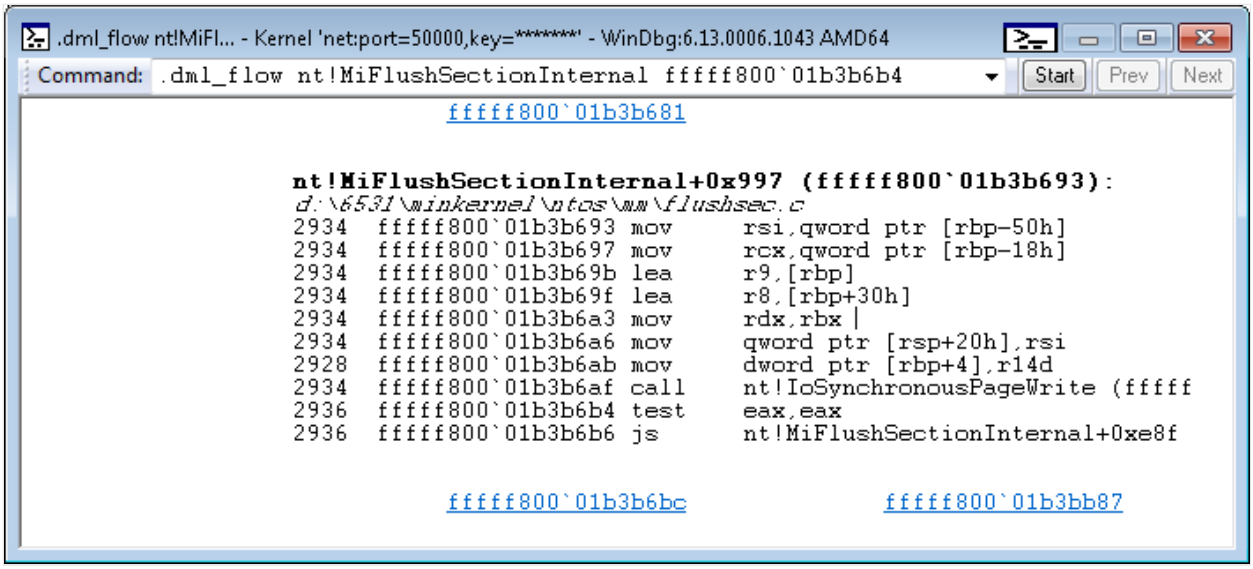
```
0: kd> kL
Child-SP          RetAddr           Call Site
fffff880`0335c688 fffff800`01b41f1c nt!IofCallDriver
fffff880`0335c690 fffff800`01b3b6b4 nt!IoSynchronousPageWrite+0x1cc
fffff880`0335c700 fffff800`01b4195e nt!MiFlushSectionInternal+0x9b8
...
```

Предположим, вы хотите изучить все пути к коду с начала **nt! MiFlushSectionInternal** для блока кода, содержащего обратный адрес , `fffff800`01b3b6b4`. Следующая команда позволяет приступить к работе.

```
dbgcmd
```

```
.browse .dml_flow nt!MiFlushSectionInternal fffff800`01b3b6b4
```

Выходные данные в окне Обозревателя команд показаны на следующем рисунке.




На предыдущем рисунке показан блок кода, содержащий целевой адрес , `fffff800`01b3b6b4`. В верхней части изображения есть только одна ссылка (`fffff800`01b3b681`). Это означает, что существует только один блок кода, из которого можно получить доступ к текущему блоку кода. Если щелкнуть ссылку, вы увидите, что блок кода дизассемблированно, а также ссылки, позволяющие продолжить изучение графа потока кода.

Две ссылки в нижней части предыдущего изображения указывают на то, что из текущего блока кода можно получить доступ к двум блокам кода.

# См. также

Команды языка разметки отладчика

**uf (функция Unasemmble)**

# .dml_start (начальная точка отображения DML)

Статья • 08.07.2024

Команда **.dml_start** отображает выходные данные, которые служат отправной точкой для изучения с помощью команд, поддерживающих язык разметки отладчика (DML).

dbgcmd

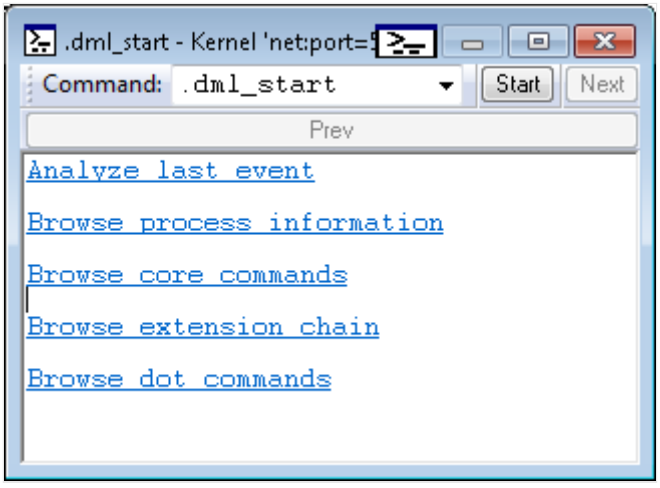
```
.dml_start
.dml_start filename
```

## Параметры

*filename*
Имя DML-файла, отображаемого в качестве начального выходных данных.

## Использование выходных данных по умолчанию

Если *имя* файла не указано, отладчик отображает выходные данные DML по умолчанию, как показано на следующем рисунке.




Каждая строка выходных данных в предыдущем примере — это ссылка, которую можно щелкнуть, чтобы вызвать другие команды.

# Предоставление DML-файла

Если указать путь к DML-файлу, файл используется в качестве начального выходных данных. Например, предположим, что файл c:\MyFavoriteCommands.txt содержит следующие текстовые и DML-теги.

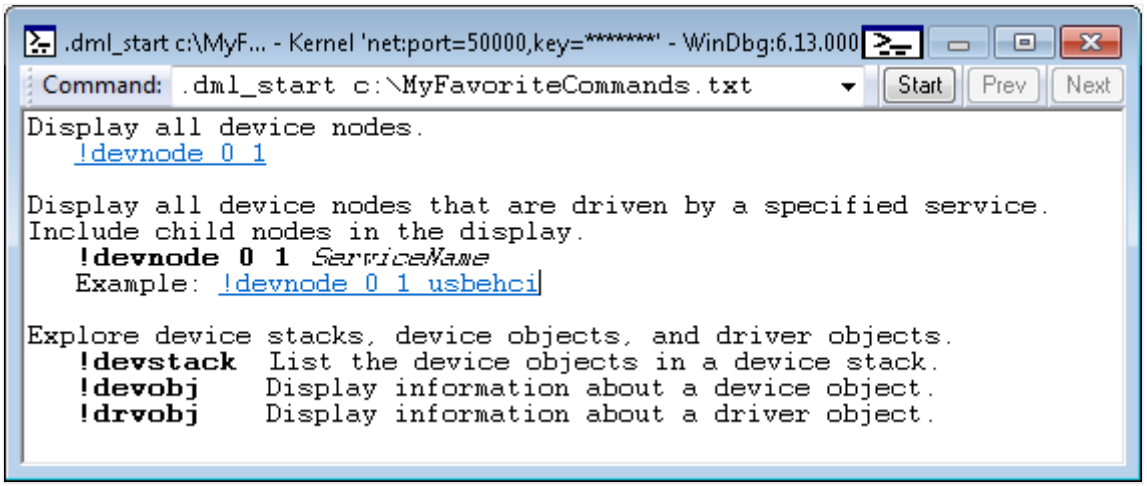
dbgcmd

```
Display all device nodes.
   <link cmd="!devnode 0 1">!devnode 0 1</link>

Display all device nodes that are driven by a specified service.
Include child nodes in the display.
   <b>!devnode 0 1</b> <i>ServiceName</i>
   Example: <link cmd="!devnode 0 1 usbehci">!devnode 0 1 usbehci</link>

Explore device stacks, device objects, and driver objects.
   <b>!devstack</b>  List the device objects in a device stack.
   <b>!devobj</b>    Display information about a device object.
   <b>!drvobj</b>    Display information about a driver object.
```

**Команда .dml_start c:\MyFavoriteCommands.txt** отобразит файл, как показано на следующем рисунке.




# Замечания

Сведения о тегах DML, которые можно использовать в файлах DML, см. в dml.doc в папке установки средств отладки для Windows.

Выходные данные DML часто хорошо работают в окне командного браузера. Чтобы отобразить DML-файл в окне командного браузера, используйте **команду .browse .dml_start** *имя* файла.

# См. также

Команды языка разметки отладчика

**.просматривать**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .dump (создание файла дампа)

Статья • 08.07.2024

Команда **.dump создает файл аварийного дампа** в режиме пользователя или в режиме ядра.

dbgcmd

```
.dump [options] FileName
.dump /?
```

# Параметры

*Параметры*
Представляет один или несколько следующих параметров.

**/a**
Создайте дампы для всех процессов (требуется -u).

**/b[a]**
Дампа пакетов в CAB-файле и удалении дампа. Дополнительные сведения включаются, если указан параметр.

**/c <комментарий>**
Добавьте комментарий (не поддерживается во всех форматах).

**/j <addr>**
Укажите адрес JIT_DEBUG_INFO.

**/o**
Перезаписывает существующий файл дампа с тем же именем. Если этот параметр не используется и имеется файл с тем же именем файла, файл дампа не записывается.

**/u**
Добавьте уникальный идентификатор в имя дампа.

**/f[***FullOptions***]**
(режим ядра:) Создает полный дамп памяти.

(Режим пользователя:) Не поддерживается. Ранее этот параметр создаст полный *дамп* в пользовательском режиме, который является устаревшим форматом с

меньшей информацией, чем новый формат minidump. Дополнительные сведения см. в разделе "Разновидности файлов дампа в режиме пользователя".

Чтобы изменить содержимое файла дампа, можно добавить следующие *функции FullOptions* . Параметр учитывает регистр.

Развернуть таблицу

| FullOption | Действие |
|---|---|
| r | Добавляет сведения о регистрации AVX в файл дампа. |

**/m**[*MiniOptions*]
*Создает небольшой дамп* памяти (в режиме ядра) или *мини-dump* (в пользовательском режиме) Дополнительные сведения см. в разделе "Файлы дампа в режиме пользователя". Если параметр **/f** и **/m не указан, значение /m** по умолчанию.

В пользовательском режиме **за /m** можно следовать с дополнительными *МиниOptions* , указывающими дополнительные данные, которые должны быть включены в дамп. Если мини-options *не* включены, дамп будет включать сведения о модуле, потоке и стеке, но нет дополнительных данных. Вы можете добавить любой из следующих *MiniOptions* , чтобы изменить содержимое файла дампа; они чувствительны к регистру.

Развернуть таблицу

| MiniOption | Действие |
|---|---|
| a | Создает мини-dump со всеми необязательными дополнениями. Параметр /ma эквивалентен параметру /mfFhut. Он добавляет полные данные памяти, обрабатывает данные, выгруженные сведения о модуле, основные сведения о памяти и сведения о времени потока в мини-dump. Любая ошибка чтения недоступной памяти приводит к прекращению создания мини-dump. |
| a | Параметр /mA эквивалентен /ma, за исключением того, что он игнорирует любые ошибки чтения недоступной памяти и продолжает создавать мини-dump. |
| f | Добавляет полные данные памяти в мини-dump. Будут включены все доступные зафиксированные страницы, принадлежащие целевому приложению. |
| F | Добавляет все основные сведения о памяти в мини-dump. Это добавляет поток в мини-dump, содержащий все основные сведения о памяти, а не |

| MiniOption | Действие |
| --- | --- |
| | только сведения о допустимой памяти. Это позволяет отладчику восстановить полный макет виртуальной памяти процесса при отладке мини-dump. |
| ч | Добавляет данные об дескрипторах, связанных с целевым приложением, в мини-dump. |
| u | Добавляет выгруженные сведения о модуле в мини-dump. |
| t | Добавляет дополнительные сведения о потоке в мини-dump. Это включает время потока, которое можно отобразить с помощью расширения !runaway или команды Ttime (Display Thread Times) при отладке мини-ddump. |
| i | Добавляет вторичную память в мини-dump. Вторичная память — это любая память, на которую ссылается указатель на стек или резервное хранилище, а также небольшой регион, окружающий этот адрес. |
| п | Добавляет блок среды обработки (PEB) и блок среды потоков (TEB) в мини-dump. Это может быть полезно, если вам нужен доступ к системным данным Windows относительно процессов и потоков приложения. |
| w | Добавляет все зафиксированные частные страницы для чтения и записи в мини-dump. |
| d | Добавляет все сегменты данных чтения и записи в исполняемом образе в мини-dump. |
| c | Добавляет разделы кода в изображениях. |
| r | Удаляет из мини-dump эти части стека и хранения памяти, которые не полезны для повторного восстановления трассировки стека. Локальные переменные и другие значения типов данных также удаляются. Этот параметр не делает мини-dump меньше (так как эти разделы памяти просто ноль), но полезно, если вы хотите защитить конфиденциальность других приложений. |
| R | Удаляет полные пути модуля из мини-модуля. Будут включены только имена модулей. Это полезный вариант, если вы хотите защитить конфиденциальность структуры каталогов пользователя. |
| г | Добавляет сведения о регистрации AVX в файл дампа. |

## Параметры режима ядра

Следующие параметры доступны в режиме ядра.

**/k**

Создайте дамп только с памятью ядра.

**/Ka**
Создайте дамп с активной памятью ядра и пользовательского режима.

# Дополнительная информация

Описание файлов дампа в режиме ядра и описание их использования см. в разделе "Файлы дампа в режиме ядра". Описание файлов дампа в пользовательском режиме и описание их использования см. в разделе "Файлы дампа в режиме пользователя".

# Замечания

Эта команда может использоваться в различных ситуациях:

- Во время динамической отладки в режиме пользователя эта команда направляет целевое приложение для создания файла дампа, но целевое приложение не завершает работу.
- Во время отладки в режиме динамического ядра эта команда направляет целевой компьютер на создание файла дампа, но целевой компьютер не завершает работу.
- Во время отладки аварийного дампа эта команда создает новый файл аварийного дампа из старого. Это полезно, если у вас есть большой файл дампа аварийного сбоя и требуется создать меньший.

Вы можете управлять типом файла дампа:

- В режиме ядра для создания полного дампа памяти используйте **параметр /f** . Чтобы создать небольшой дамп памяти, используйте **параметр /m** (или нет параметров). Команда .dump не может создать дамп памяти ядра.
- В пользовательском режиме **.dump /m**[*MiniOptions*] — лучший вариант. Хотя "m" обозначает "minidump", файлы дампа, созданные с помощью этого *MiniOption* , могут отличаться от очень малого до очень большого. Указав правильные *miniOptions* , вы можете точно контролировать, какие сведения включены. Например, **.dump /ma** создает дамп с большой информацией. Старая команда , **dump /f** создает умеренно большой файл "стандартный дампа" и не может быть настроен.

Нельзя указать, какой процесс сбрасывается. Все выполняемые процессы будут дампами.

Параметры **/xc**, /xr, **/xp** и /xt **используются для хранения сведений об исключении и** контексте в файле дампа. Это позволяет **выполнять команду ECXR (отображаемая запись контекста исключения)** в этом файле дампа.

В следующем примере будет создан мини-dump в пользовательском режиме, содержащий полную память и обрабатывать сведения:

```
dbgcmd

0:000> .dump /mfh myfile.dmp
```

Сведения об обработке **можно считывать с помощью команды расширения !handle** .

## См. также

Разновидности файлов дампа в режиме ядра

.ecxr

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .dumpcab (создание CAB-файла дампа)

Статья • 08.07.2024

Команда **.dumpcab** создает CAB-файл, содержащий текущий файл дампа.

```
dbgcmd
```

```
.dumpcab [-a] CabName
```

## Параметры

**-a**
Вызывает включение всех загруженных в данный момент символов в CAB-файл. Для мини-модулей все загруженные изображения также будут включены. Используйте **любовь к моей жизни** , чтобы определить, какие символы и изображения загружаются.

*CabName*
Имя CAB-файла, включая расширение. *CabName* может включать абсолютный или относительный путь; относительные пути относятся к каталогу, в котором был запущен отладчик. Рекомендуется выбрать расширение .cab.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | режим пользователя, режим ядра |
| **Целевые объекты** | аварийного дампа |
| **Платформы** | all |

## Дополнительная информация

Дополнительные сведения о аварийных дампах см. в разделе "Файлы аварийного дампа".

# Замечания

Эту команду можно использовать только в том случае, если вы уже отладили файл дампа.

Если вы выполняете отладку динамического целевого объекта и хотите создать файл дампа и поместить его в CAB-файл, используйте **команду .dump (Create Dump File).** Затем запустите новый сеанс отладки с файлом дампа в качестве целевого объекта и используйте **.dumpcab**.

**Команда .dumpcab не может использоваться для размещения нескольких файлов дампа** в один CAB-файл.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Dvalloc (выделение памяти)

Статья • 08.07.2024

Команда **Dvalloc** приводит к тому, что Windows выделяет дополнительную память целевому процессу.

```
dbgcmd
```

```
.dvalloc [Options] Size
```

## Параметры

*Параметры* могут быть любым количеством следующих параметров:

**/b** **** *BaseAddress*
Задает виртуальный адрес начала выделения.

**/r**
Резервирует память в виртуальном адресном пространстве, но на самом деле не выделяет физическую память. Если этот параметр используется, отладчик вызывает **VirtualAllocEx** с параметром *flAllocationType* , равным MEM_RESERVE. Если этот параметр не используется, значение MEM_COMMIT | используется MEM_RESERVE. Дополнительные сведения см. в пакете SDK для Microsoft Windows.

*Размер*
Указывает объем выделенной памяти в байтах. Объем памяти, доступный программе, равен *размеру*. Объем памяти на самом деле может быть немного больше, так как это всегда целый ряд страниц.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

# Замечания

Команда **Dvalloc вызывает VirtualAllocEx**, чтобы выделить новую память для целевого процесса. Выделенная память разрешает чтение, запись и выполнение.

Чтобы освободить эту память, используйте **.dvfree (бесплатная память)**.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .dvfree (бесплатная память)

Статья • 08.07.2024

Команда **DVFREE** освобождает выделение памяти, принадлежащей целевому процессу.

```
dbgcmd
```

```
.dvfree [/d] BaseAddress Size
```

## Параметры

**/d**
Отменяет выделение, но на самом деле не освобождает страницы, содержащие выделение. Если этот параметр используется, отладчик вызывает **VirtualFreeEx** с параметром *dwFreeType* равным MEM_DECOMMIT. Если этот параметр не используется, используется значение MEM_RELEASE. Дополнительные сведения см. в пакете SDK для Microsoft Windows.

*BaseAddress*
Задает виртуальный адрес начала выделения.

*Размер*
Указывает объем памяти для освобождения в байтах. Фактическое освобождение памяти всегда будет целым количеством страниц памяти.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Замечания

Команда **DVFREE** вызывает **VirtualFreeEx** для освобождения существующего выделения памяти. Если **параметр /d** не указан, страницы, содержащие эту память, освобождаются.

Эту команду можно использовать для освобождения выделения, сделанного **dvalloc (выделение памяти).** Его также можно использовать для освобождения любого блока памяти, принадлежащих целевому процессу, но освобождение памяти, которая не была приобретена с помощью **Dvalloc** , естественно, представляет угрозу стабильности целевого процесса.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .echo (эхо-комментарий)

Статья • 08.07.2024

Команда **.echo** отображает строку комментария.

```
dbgcmd
```

```
.echo String
.echo "String"
```

## Параметры

*Струна*
Задает отображаемый текст. Можно также заключить *строку* в кавычки ("). Независимо от того, используется ли кавычки, *Строка* может содержать любое количество пробелов, запятых и отдельных кавычки ('). Если строка заключена в кавычки, она может включать запятую, но не дополнительные кавычки. Если строка не заключена в кавычки, она может включать кавычки в любое расположение, кроме первого символа, но не может включать запятую.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Команда **.echo** приводит к отображению *строки* отладчиком сразу после ввода команды.

Команда **.echo** завершается, если отладчик встречает точку с запятой (если точка с запятой не возникает в кавычки). Это ограничение позволяет использовать **.echo** в

более сложных конструкциях, таких как с помощью команды j (Execute If - Else), как показано в следующем примере.

```
dbgcmd

0:000> j (poi(MyVar)>5) '.echo MyVar Too Big'; '.echo MyVar Acceptable;
```

Команда **.echo** также предоставляет простой способ взаимодействия с пользователями серверов отладки и отладки клиентов для взаимодействия друг с другом. Дополнительные сведения об этой ситуации см. в разделе "Управление сеансом удаленной отладки".

Команда **.echo** отличается от **токена $$ (Описатель комментариев)** и **маркера * (Описатель строки комментариев),** так как эти маркеры вызывают отладчик игнорировать входной текст без отображения.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .echocpunum (показать номер ЦП)

Статья • 08.07.2024

Команда **.echocpunum** включает или отключает отображение номера процессора при отладке целевого компьютера с несколькими обработчиками.

dbgcmd

```
.echocpunum 0
.echocpunum 1
.echocpunum
```

## Параметры

**0**
Отключает отображение номера процессора.

**1**
Включает отображение номера процессора.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Все |

## Дополнительная информация

Дополнительные сведения об отладке многопроцессорных компьютеров см. в разделе "Синтаксис многопроцессора".

## Замечания

Если вы используете **команду .echocpunum** без каких-либо аргументов, отображение номеров процессора включается или отключается, а новое состояние отображается.

По умолчанию экран отключен.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .echotime (показать текущее время)

Статья • 26.10.2023

Команда **.echotime** отображает текущее время.

```
dbgcmd
```

```
.echotime
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

Команда **.echotime** отображает время для компьютера, на котором запущен отладчик.

# См. также раздел

.echotimestamps (Show Time Stamps)

.time (отображение системного времени)

!runaway

# .echotimestamps (показать метки времени)

Статья • 08.07.2024

Команда **.echotimestamps** включает или отключает отображение сведений метки времени.

dbgcmd

```
.echotimestamps 0
.echotimestamps 1
.echotimestamps
```

## Параметры

**0**
Отключает отображение сведений о метки времени. Это поведение отладчика по умолчанию.

**1**
Включает отображение сведений метки времени.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о **DbgPrint, KdPrint, DbgPrintEx и KdPrintEx** см. в разделе "Буфер DbgPrint" в разделе "Чтение и фильтрация сообщений отладки".

# Замечания

При использовании **команды .echotimestamps** без параметров отображение меток времени включается или отключается, а новое состояние отображается.

Если включить этот дисплей, отладчик отображает метки времени для загрузки модулей, создания потоков, исключений и других событий.

**Подпрограммы dbgPrint, KdPrint, DbgPrintEx и KdPrintEx** в режиме ядра отправляют форматированную строку в буфер на хост-компьютере. Строка отображается в окне команды отладчика (если вы не отключили такую печать). Кроме того, можно отобразить форматированную строку с помощью команды расширения !dbgprint.

При использовании **эхо-меток времени для включения отображения меток** времени отображается время и дата каждого комментария в буфере **DbgPrint** .

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .ecxr (отображение записи контекста исключения)

Статья • 26.10.2023

Команда **.ecxr** отображает запись контекста, связанную с текущим исключением.

```
dbgcmd
```

```
.ecxr
```

## Среда

| | |
|---|---|
| **Режимы** | Только в пользовательском режиме |
| **Целевые объекты** | Только аварийный дамп (только минидампы) |
| **Платформы** | Все |

## Дополнительные сведения

Дополнительные сведения о контексте регистрации и других параметрах контекста см. в разделе Изменение контекстов.

# Комментарии

Команда **ECXR** находит сведения о контексте текущего исключения и отображает важные регистры для указанной записи контекста.

Эта команда также указывает отладчику использовать запись контекста, связанную с текущим исключением, в качестве контекста регистра. После запуска **.ecxr** отладчик может получить доступ к наиболее важным регистрам и трассировке стека для этого потока. Этот контекст регистра сохраняется до тех пор, пока вы не включите выполнение целевого объекта, не измените текущий процесс или поток или не используете другую команду контекста **регистра (.cxr** или **.ecxr**). Дополнительные сведения о контекстах регистрации см. в разделе Контекст регистрации.

Команда **.excr** является синонимом и имеет идентичные функциональные возможности.

# См. также

.excr

# .effmach (эффективный компьютер)

Статья • 08.07.2024

Команда **.effmach** отображает или изменяет режим процессора, который использует отладчик.

dbgcmd

```
.effmach [MachineType]
```

## Параметры

*MachineType*
Указывает тип процессора, используемый отладчиком для этого сеанса. Если этот параметр не указан, отладчик отображает текущий тип компьютера.

Можно ввести один из следующих типов компьютеров.

Развернуть таблицу

| Тип компьютера | Description |
|---|---|
| . | Используйте режим процессора собственного процессора целевого компьютера. |
| # | Используйте режим процессора кода, выполняемого для последнего события. |
| **x86** | Используйте режим процессора на основе x86. |
| **amd64** | Используйте режим процессора на основе x64. |
| **ebc** | Используйте режим обработчика кода EFI байтов. |
| **arm** | Используйте режим процессора Arm64. |
| **chpe** | Используйте режим процессора CHPE. |

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Режим процессора влияет на многие функции отладчика:

- Какой процессор используется для трассировки стека.
- Используется ли процесс 32-разрядных или 64-разрядных указателей.
- Какой набор регистров процессора активен.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .enable_long_status (включение длинного целочисленного отображения)

Статья • 08.07.2024

Команда **.enable_long_status** указывает, отображает ли отладчик длинные целые числа в десятичном формате или в радиксе по умолчанию.

dbgcmd

```
.enable_long_status 0
.enable_long_status 1
```

## Параметры

**0**
Отображает все длинные целые числа в десятичном формате. Это поведение отладчика по умолчанию.

**1**
Отображает все длинные целые числа в радиксе по умолчанию.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Команда **.enable_long_status влияет на выходные данные** команды dt (тип отображения).

В WinDbg **.enable_long_status** также влияет на отображение в окне "Локальные" и "Локальные"

Окно просмотра. Эти окна автоматически обновляются после .enable_long_status проблемы.

Эта команда влияет только на отображение длинных целых чисел. Чтобы управлять отображением стандартных целых чисел в десятичном формате или радиксом по умолчанию, используйте **команду .force_radix_output (использовать Radix для целых чисел).**

**Обратите внимание, что команда dv (Display Local Variables)** всегда отображает длинные целые числа в текущей базе чисел.

Чтобы изменить радикс по умолчанию, используйте **команду n (Set Number Base).**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .enable_unicode (включить отображение Юникода)

Статья • 08.07.2024

Команда **.enable_unicode** указывает, отображает ли отладчик указатели и массивы USHORT в виде строк Юникода.

dbgcmd

```
.enable_unicode 0
.enable_unicode 1
```

## Параметры

**0**
Отображает все 16-разрядные массивы (USHORT) и указатели в виде коротких целых чисел. Это поведение отладчика по умолчанию.

**1**
Отображает все 16-разрядные массивы и указатели USHORT в виде строк Юникода.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Команда **.enable_unicode влияет на выходные данные** команды dt (тип отображения).

В WinDbg **команда .enable_unicode** также влияет на отображение в окне "Локальные" и окне "Контроль". Эти окна автоматически обновляются после

.enable_unicode проблемы.

Вы также можете выбрать или очистить **16-разрядные значения** отображения в виде Юникода в контекстном меню окна "Локальные" или "Контрольные", чтобы указать отображение массивов и указателей USHORT.

## См. также

**ds, dS (отображаемая строка)**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .endpsrv (end process server)

Статья • 26.10.2023

Команда **.endpsrv** приводит к закрытию текущего сервера обработки или сервера подключения KD.

```
dbgcmd
```

```
.endpsrv
```

## Среда

Эту команду можно использовать только при выполнении удаленной отладки через сервер обработки или сервер подключения KD.

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения об этих серверах см. в разделах Серверы обработки (режим пользователя) или Серверы подключения KD (режим ядра)

# Комментарии

Команда **.endpsrv** завершает сервер обработки или сервер подключения KD, подключенный к интеллектуальному клиенту.

Если вы хотите завершить работу сервера обработки или сервера подключения KD с компьютера, на котором он запущен, используйте диспетчер задач для завершения процесса (dbgsrv.exe или kdsrv.exe).

Команда **.endpsrv** может завершить сервер обработки или сервер подключения KD, но не может завершить сервер отладки. Сведения о том, как это сделать, см. в разделе Управление сеансом удаленной отладки.

# .endrv (конечный сервер отладки)

Статья • 08.07.2024

Команда **.endsrv** приводит к отмене активного сервера отладки отладчиком.

```
dbgcmd

.endsrv ServerID
```

## Параметры

*ServerID*
Указывает идентификатор сервера отладки.

## Среда

Эту команду можно использовать только при выполнении удаленной отладки через отладчик.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только в пользовательском режиме |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения об удаленной отладке см. в разделе "Удаленная отладка через отладчик".

## Замечания

Необходимо выполнить **команду .endsrv** с сервера отладки или одного из клиентов отладки, подключенных к серверу отладки.

Чтобы определить идентификатор сервера отладки, используйте **команду .servers (List Debugging Servers).**

**Команда .endsrv** может завершить отладчик сервера, но не может завершить сервер обработки или сервер подключения KD. Сведения о том, как завершить эти серверы, см. в разделе "Управление сеансом сервера обработки и управление сеансом сервера подключения KD". (Однако существует один исключительный случай, когда **.endrv** может завершить сервер обработки, запущенный программным способом; дополнительные сведения см. в разделе **IDebugClient::StartProcessServer**.)

Если отменить отладку сервера, вы не допустите подключение будущих клиентов отладки к серверу. Однако при отмене отладчика сервера вы не отсоединяете клиенты, которые в настоящее время подключены через сервер.

Рассмотрим следующую ситуацию. Предположим, что вы запускаете некоторые серверы отладки, как показано в следующем примере.

dbgcmd

```
0:000> .server npipe:pipe=rabbit
Server started with 'npipe:pipe=rabbit'
0:000> .server tcp:port=7
Server started with 'tcp:port=7'
```

Затем вы решили использовать пароль, как показано в следующем примере.

dbgcmd

```
0:000> .server npipe:pipe=tiger,password=hardtoguess
Server started with 'npipe:pipe=tiger,password=hardtoguess'
```

Но предыдущие серверы по-прежнему работают, поэтому их следует отменить, как показано в следующем примере.

dbgcmd

```
0:000> .servers
0 - Debugger Server - npipe:Pipe=rabbit
1 - Debugger Server - tcp:Port=7
2 - Debugger Server - npipe:Pipe=tiger,Password=*
0:000> .endsrv 0
Server told to exit.  Actual exit may be delayed until
the next connection attempt.
0:000> .endsrv 1
Server told to exit.  Actual exit may be delayed until
the next connection attempt.
```

```
0:000> .servers
0 - <Disabled, exit pending>
1 - <Disabled, exit pending>
2 - Debugger Server - npipe:Pipe=tiger,Password=*
```

Наконец, чтобы убедиться, что ничего не подключено к компьютеру, пока предыдущие серверы были активными, используйте **команду .client (List Debugging Clients).**

dbgcmd

```
0:000> .clients
HotMachine\HostUser, last active Mon Mar 04 16:05:21 2002
```

**Внимание, что** использование пароля с протоколом TCP, NPIPE или COM обеспечивает только небольшое количество защиты, так как пароль не шифруется. При использовании пароля вместе с протоколом SSL или SPIPE пароль шифруется. Если вы хотите установить безопасный удаленный сеанс, необходимо использовать протокол SSL или SPIPE.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Enumtag (перечисление вторичных данных обратного вызова)

Статья • 05.04.2024

Команда **enumtag** отображает вторичные ошибки проверка данных обратного вызова и всех тегов данных.

```
dbgcmd
```

```
.enumtag
```

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения и другие способы отображения этих данных см. в статье "Чтение данных обратного вызова проверки ошибок".

## Замечания

Команду enumtag **можно использовать** только после возникновения ошибки проверка или при отладке файла аварийного дампа в режиме ядра.

Для каждого блока вторичных ошибок проверка данных **обратного вызова команда enumtag** отображает тег (в формате GUID) и данные (как байты и символы ASCII).

Рассмотрим следующий пример.

```
dbgcmd
```

```
kd> .enumtag
{87654321-0000-0000-0000000000000000} - 0xf9c bytes
  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..............
  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
  ....
  00 00 00 00 00 00 00 00 00 00 00 00              ............

{12345678-0000-0000-0000000000000000} - 0x298 bytes
  F4 BF 7B 80 F4 BF 7B 80 00 00 00 00 00 00 00 00  ..{...{.........
 4B 44 42 47 98 02 00 00 00 20 4D 80 00 00 00 00  KDBG..... M.....
  54 A5 57 80 00 00 00 00 A0 50 5A 80 00 00 00 00  T.W......PZ.....
 40 01 08 00 18 00 00 00 BC 7D 50 80 00 00 00 00  @........}P.....
  ....
  00 00 00 00 00 00 00 00                          ........
```

В этом примере первый пакет вторичных данных имеет GUID{87654321-0000-0000-0000000000000000} )) в качестве тега, а второй пакет данных имеет GUID ({12345678-0000-0000-0000000000000000}) в качестве тега. Однако данные не в удобном формате.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .event_code (отображение кода события)

Статья • 26.10.2023

Команда **.event_code** отображает текущие инструкции по событиям.

```
dbgcmd
```

```
.event_code
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | пользовательский режим, режим ядра |
| Цели | Только динамическая отладка |
| Платформы | все |

# Комментарии

Команда **.event_code** отображает шестнадцатеричные инструкции в указателе инструкции текущего события. Отображение содержит до 64 байт инструкций, если они доступны.

# .eventlog (отображение последних событий)

Статья • 26.10.2023

Команда **.eventlog** отображает последние события отладки Microsoft Win32, такие как загрузка модуля, создание и завершение процесса, создание и завершение потока.

dbgcmd

```
.eventlog
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

Команда **.eventlog** отображает только 1024 символа.

В следующем примере показана команда **.eventlog** .

dbgcmd

```
0:000> .eventlog
0904.1014: Load module C:\Windows\system32\ADVAPI32.dll at 000007fe`fed80000
0904.1014: Load module C:\Windows\system32\RPCRT4.dll at 000007fe`fe8c0000
0904.1014: Load module C:\Windows\system32\GDI32.dll at 000007fe`fea00000
0904.1014: Load module C:\Windows\system32\USER32.dll at 00000000`76b10000
0904.1014: Load module C:\Windows\system32\msvcrt.dll at 000007fe`fe450000
0904.1014: Load module C:\Windows\system32\COMDLG32.dll at 000007fe`fecf0000
0904.1014: Load module C:\Windows\system32\SHLWAPI.dll at 000007fe`fe1f0000
0904.1014: Load module C:\Windows\WinSxS\amd64_microsoft.windows.common-
controls_6595b6414
4ccf1df_6.0.6000.16386_none_1559f1c6f365a7fa\COMCTL32.dll at
000007fe`fbda0000
0904.1014: Load module C:\Windows\system32\SHELL32.dll at 000007fe`fd4a0000
0904.1014: Load module C:\Windows\system32\WINSPOOL.DRV at 000007fe`f77d0000
```

```
0904.1014: Load module C:\Windows\system32\ole32.dll at 000007fe`feb10000
0904.1014: Load module C:\Windows\system32\OLEAUT32.dll at 000007fe`feeb0000
Last event: Break instruction exception - code 80000003 (first chance)
```

# .exdicmd (команда EXDI)

Статья • 01.11.2024

**Exdicmd** отправляет команду расширенного интерфейса отладки (EXDI) в целевую систему с помощью активного подключения отладки EXDI. Дополнительные сведения об EXDI см. в разделе "Настройка транспорта отладчика EXDI".

```
dbgcmd

exdicmd component|target parameters
```

Эта команда передает параметры непосредственно компоненту EXDI. Дополнительные сведения о допустимых командах, доступных для целевой системы, см. в документации по компоненту EXDI.

Не все компоненты EXDI реализуют эту функцию.

## Параметры

Это допустимые параметры для **exdicmd**.

Развернуть таблицу

| Параметр | Описание |
|---|---|
| `target:*:<string>` | Передайте функцию целевой `<string>` конечной сущности для всех ядер процессора. |
| `target:<n>:<string>` | Передайте функцию `<string>` целевой конечной сущности для ядра процессора n (n-десятичное число). |
| `component:*:<string>` | Выполните функцию компонента `<string>` EXDI во всех ядрах процессора. |
| `component:<n>:<string>` | Выполните функцию компонента `<string>` EXDI в ядре процессора n (n-десятичное число). |
| `help` | Отображение базовой справки. |

## Использование exdicmd целевого объекта

`.exdicmd target:*:<string>`

Использование целевого параметра позволяет взаимодействовать между отладчиком Windows и COM-сервером EXDI. Отладчик отобразит результат команды, если команда возвращает ответ обратно подсистеме отладчика.

Параметр целевого объекта .exdicmd принимает любую команду, которую может обрабатывать СЕРВЕР EXDI COM. Это позволяет использовать команды вне того, что поддерживается непосредственно в интерфейсе EXDI.

# Использование exdicmd компонента

`.exdicmd <component>:`

Эта команда предназначена в первую очередь для выполнения и тестирования функций COM-сервера EXDI для проверки основных функциональных возможностей. Обычно он используется меньше, чем использование целевого параметра, описанное выше.

Обратите внимание, что существуют команды, которые могут выполняться внутри сервера EXDI COM без необходимости отправлять его в целевую сущность JTAG. Например, можно собирать данные телеметрии с COM-сервера EXDI для проверки правильной операции.

# Среда

Развернуть таблицу

| Дескриптор | Значение |
|---|---|
| Режимы | Только режим ядра |
| Назначение | Только динамическая отладка |
| Платформы | Все |

# Дополнительная информация

Пример использования **exdicmd** с целевым объектом OpenOCD, который использует синтаксис, `.exdicmd target:0:<OpenOCD command>` показан ниже.

dbgcmd

```
0: kd> .exdicmd target:0:info network
Target command response: e1000.0:
```

```
index=0,type=nic,model=e1000,macaddr=52:54:00:12:34:56
 \ net0: index=0,type=user,net=10.0.2.0,restrict=off
OK
exdiCmd: The function: 'info network' was completed.
```

dbgcmd

```
0: kd> .exdicmd target:0:info registers system -v
Target command response:
NumberOfRegisters: 9

     Name | Value            | Access code
  fs_base | 0000000000000000 | n/a
  gs_base | fffff8047b907000 | n/a
k_gs_base | 000000e7cbdbe000 | n/a
      cr0 | 0000000080050033 | n/a
      cr2 | fffff8048454de64 | n/a
      cr3 | 00000000001ae000 | n/a
      cr4 | 00000000000006f8 | n/a
      cr8 | 0000000000000000 | n/a
     efer | 0000000000000d01 | 0xc0000080

exdiCmd: The function: 'info registers system -v' was completed.
```

Если целевая система не может понять команду, и *будет возвращено неизвестное сообщение команды* .

dbgcmd

```
0: kd> .exdicmd target:0:Foo
Target command response: unknown command: 'Foo'
```

# .outmask

Если выходные данные не отображаются, убедитесь, что для маски вывода задан подробный вывод с помощью outmask (Control Output Mask).

Если отладчик не был запущен в подробном режиме, можно задать .outmask с помощью `.outmask 7FF`.

# Замечания

Дополнительные сведения о настройке подключения отладчика EXDI см. в разделе "Настройка транспорта отладчика EXDI".

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .exepath (задать путь к исполняемому файлу)

Статья • 08.07.2024

Команда **.exepath** задает или отображает путь поиска исполняемого файла.

```
dbgcmd
.exepath[+] [Directory [; ...]]
```

## Параметры

+
Указывает, что отладчик должен добавить новые каталоги в предыдущий путь поиска исполняемого файла (вместо замены пути).

*Каталог*
Указывает один или несколько каталогов, которые нужно поместить в путь поиска. Если каталог не указан, отображается текущий путь. Можно разделить несколько каталогов с запятой.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

В большинстве случаев отладчик знает расположение исполняемых файлов, поэтому вам не нужно задавать путь к этому файлу.

Однако существуют ситуации, когда этот путь необходим. Например, файлы дампа памяти в режиме ядра не содержат все исполняемые файлы, существующие в

памяти во время ошибки остановки (то есть сбой). Аналогичным образом мини-файлы в режиме пользователя не содержат двоичные файлы приложения. Если задать путь к исполняемым файлам, отладчик может найти эти двоичные файлы. Дополнительные сведения см. в разделе "Настройка пути к символу и исполняемому изображению" в WinDbg.

Путь поиска исполняемого файла также можно задать с помощью переменной `_NT_EXECUTABLE_IMAGE_PATH` среды. Дополнительные сведения см. в разделе "Общие переменные среды".

## См. также

Настройка путей к символам и исполняемым изображениям в WinDbg

Общие переменные среды

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .expr (выбор вычислителя выражений)

Статья • 08.07.2024

Команда **EXPR** указывает средство оценки выражений по умолчанию.

dbgcmd

```
.expr /s masm
.expr /s c++
.expr /q
.expr
```

## Параметры

**/s masm**
Изменяет тип выражения по умолчанию на средство оценки выражений Microsoft Assembler (MASM). Этот тип является значением по умолчанию при запуске отладчика.

**/s c++**
Изменяет тип выражения по умолчанию на средство оценки выражений C++.

**/q**
Отображает список возможных типов выражений.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

При использовании **команды EXPR** без аргумента отладчик отображает текущий тип выражения по умолчанию.

? **? (Оценка выражения C++ )** команда, окно "Контроль" и окно "Локальные" всегда используют синтаксис выражения C++. Все остальные команды и окна сведений отладки используют средство оценки выражений по умолчанию.

Дополнительные сведения о том, как управлять используемым синтаксисом, см. в разделе "Оценка выражений". Дополнительные сведения о синтаксисе см. в разделе "Синтаксис числовых выражений".

## См. также

? (Вычислять выражение)

Номера и операторы MASM

Номера и операторы C++

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .exptr (отображение указателей исключений)

Статья • 08.07.2024

Команда **.exptr** отображает структуру EXCEPTION_POINTERS.

```
dbgcmd
```

```
.exptr Address
```

## Параметры

*Адрес*
Задает адрес структуры EXCEPTION_POINTERS.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .exr (отображение записи исключений)

Статья • 08.07.2024

Команда **EXR** отображает содержимое записи исключений.

```
dbgcmd
```

```
.exr Address
.exr -1
```

## Параметры

*Адрес* — указывает адрес записи исключения. При указании **-1** в качестве адреса отладчик отображает последнее исключение.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Команда **EXR** отображает сведения, связанные с исключением, которое отладчик столкнулся на целевом компьютере. Отображаемые сведения включают адрес исключения, код исключения, флаги исключений и список параметров исключения.

Обычно адрес можно получить *с помощью **расширения !pcr***.

Команда **EXR** часто используется для отладки проверки ошибок 0x1E. Дополнительные сведения и пример см. в разделе **"Проверка ошибок" 0x1E** (KMODE_EXCEPTION_NOT_HANDLED).

## См. также

Изменение контекстов

Регистрация контекста

.ecxr (отображаемая запись контекста исключения)

.ловушка

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .excr (Display Exception Context Record)

Статья • 26.10.2023

Команда **.excr** отображает запись контекста, связанную с текущим исключением.

```
dbgcmd
```

```
.excr
```

## Среда

| | |
|---|---|
| **Режимы** | Только в пользовательском режиме |
| **Целевые объекты** | Только аварийный дамп (только минидампы) |
| **Платформы** | Все |

## Дополнительные сведения

Дополнительные сведения о контексте регистрации и других параметрах контекста см. в разделе Изменение контекстов.

# Комментарии

Команда **.excr** находит сведения о контексте текущего исключения и отображает важные регистры для указанной записи контекста.

Эта команда также указывает отладчику использовать запись контекста, связанную с текущим исключением, в качестве контекста регистра. После запуска **EXCR** отладчик может получить доступ к наиболее важным регистрам и трассировке стека для этого потока. Этот контекст регистра сохраняется до тех пор, пока вы не разрешите выполнение целевого объекта, не измените текущий процесс или поток или не используете другую команду контекста **регистра (.cxr** или **.excr**). Дополнительные сведения о контекстах регистрации см. в разделе Контекст регистрации.

Команда **ECXR** является командой синонима и имеет идентичную функциональность.

# См. также

**ECXR-файл**

# .extmatch (отображение всех расширений сопоставления)

Статья • 08.07.2024

В команде **.extmatch** отображаются команды расширения, экспортированные загруженными библиотеками DLL расширения, которые соответствуют указанному шаблону.

dbgcmd

```
.extmatch [Options] Pattern
```

## Параметры

*Параметры* задает параметры поиска. Можно использовать один или несколько следующих параметров:

**/e** **** *ExtDLLPattern*
Ограничивает перечисление командами расширения, экспортируемыми библиотеками DLL расширения, которые соответствуют указанной строке шаблона. *ExtDLLPattern* соответствует имени файла DLL расширения.

**/n**
Исключает имя расширения при отображении расширений. Таким образом, если этот параметр указан, будет отображаться только имя расширения.

**/D**
Отображает выходные данные с помощью языка разметки отладчика (DML). В выходных данных каждое указанное расширение — это ссылка, которую можно щелкнуть, чтобы получить дополнительные сведения о расширении. Не все модули расширения поддерживают DML.

*Рисунок*
Указывает шаблон, который должен содержать расширение. *Шаблон*может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис подстановочных знаков строки".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Чтобы отобразить список загруженных библиотек DLL расширения, используйте **команду цепочки** .

Ниже приведен пример этой команды, в которой показаны все загруженные библиотеки DLL расширения с именем !help:

dbgcmd

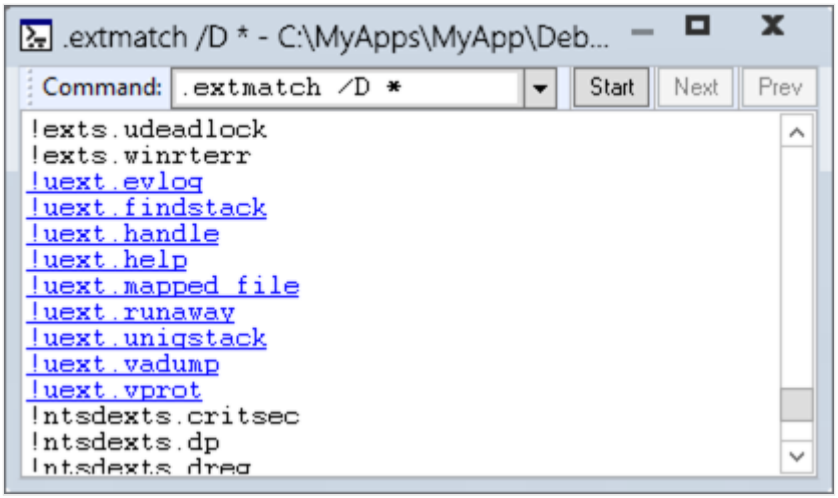
```
0:000> .extmatch help
!ext.help
!exts.help
!uext.help
!ntsdexts.help
```

В следующем примере перечислены все команды расширения, начинающиеся со строки "he", экспортируемой библиотеками DLL расширения, имена которых начинаются со строки ex:

dbgcmd

```
0:000> .extmatch /e ext* he*
!ext.heap
!ext.help
!exts.heap
!exts.help
```

В следующем примере перечислены все команды расширения, поэтому мы видим, какие из них поддерживают DML.




## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .extpath (задать путь расширения)

Статья • 08.07.2024

Набор **команд .extpath** или отображение пути поиска библиотеки DLL расширения.

```
dbgcmd
```

```
.extpath[+] [Directory[;...]]
```

## Параметры

+
Указывает, что отладчик должен добавлять новые каталоги в предыдущий путь поиска библиотеки DLL расширения (вместо замены пути).

*Каталог*
Указывает один или несколько каталогов, которые нужно поместить в путь поиска. Если каталог не указан, отображается текущий путь. Можно разделить несколько каталогов с запятой.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о пути поиска расширений и загрузке БИБЛИОТЕК DLL расширений см. в разделе "Загрузка БИБЛИОТЕК DLL расширений отладчика".

## Замечания

Путь поиска библиотеки DLL расширения сбрасывается до значения по умолчанию в начале каждого сеанса отладки.

Во время динамической отладки в режиме ядра перезагрузка целевого компьютера приводит к началу нового сеанса отладки. Поэтому любые изменения, внесенные в путь поиска библиотеки DLL расширения во время отладки в режиме ядра, не будут сохраняться во время перезагрузки целевого компьютера.

Значение по умолчанию пути поиска библиотеки DLL расширения содержит все пути расширения, известные отладчику, и все пути в переменной среды %PATH% . Например, предположим, что переменная среды %PATH% имеет значение `C:\Windows\system32;C:\Windows`. Затем значение по умолчанию пути поиска расширения DLL может выглядеть следующим образом.

dbgcmd

```
0:000> .extpath
Extension search path is: C:\Program Files\Debugging Tools for Windows
(x64)\WINXP;C:\Program Files\
Debugging Tools for Windows (x64)\winext;C:\Program Files\Debugging Tools
for Windows (x64)\winext\
arcade;C:\Program Files\Debugging Tools for Windows (x64);C:\Program
Files\Debugging Tools for
Windows (x64)\winext\arcade;C:\Windows\system32;C:\Windows
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .f+, .f- (локальный контекст SHIFT)

Статья • 26.10.2023

Команда **.f+** перемещает индекс кадра на следующий кадр в текущем стеке.
Команда **.f-** перемещает индекс кадра на предыдущий кадр в текущем стеке.

dbgcmd

```
.f+
.f-
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения о локальном контексте и других параметрах контекста см. в разделе Изменение контекстов. Дополнительные сведения о том, как отображать локальные переменные и другие команды, связанные с памятью, см. в разделе Чтение и запись памяти.

# Комментарии

*Фрейм* указывает локальный контекст (область), используемый отладчиком для интерпретации локальных переменных.

Команды **.f+** и .f- — это сочетания клавиш для перехода к следующему и предыдущему кадрам в текущем стеке. Эти команды эквивалентны следующим командам **.frame** , но для удобства команды **.f** короче:

- **.f+** совпадает с **.frame @$frame + 1**.
- **.f-** совпадает с **.frame @$frame - 1**.

Знак доллара ($) идентифицирует значение кадра как псевдорегистр. Знак at (@) приводит к тому, что отладчик быстрее получает доступ к значению, так как уведомляет отладчик о том, что строка является регистром или псевдорегистралем.

При выполнении приложения значение локальных переменных зависит от расположения счетчика программы, так как область таких переменных распространяется только на функцию, в которую они определены. Если вы не используете команды **.f+** или **.f-** (или **frame**), отладчик использует область текущей функции (текущий кадр в стеке) в качестве локального контекста.

*Номер кадра* — это положение кадра стека в трассировке стека. Эту трассировку стека можно просмотреть с помощью команды **k, kb, kc, kd, kp, kP, kv (Display Stack Backtrace)** или окна Вызовы. Первая строка (текущий кадр) представляет номер кадра 0. Последующие строки представляют собой номера кадров 1, 2, 3 и т. д.

Для просмотра новых сведений о локальных переменных в локальном контексте можно задать другой фрейм стека. Однако фактические доступные переменные зависят от выполняемого кода.

При выполнении программы отладчик сбрасывает локальный контекст до область счетчика программы. При изменении контекста регистра локальный контекст сбрасывается в верхний кадр стека.

# .fibre (задать контекст волокна)

Статья • 08.07.2024

Команда **.fibre** указывает, какое волокна используется для контекста волокна.

```
dbgcmd
```

```
.fiber [Address]
```

## Параметры

*Адрес*
Указывает адрес волокна. Если этот параметр опущен или указать ноль, контекст волокна сбрасывается в текущее волокна.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о контексте процесса, контексте регистрации и локальном контексте см. в разделе "Изменение контекстов".

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .fiximports (исправление импорта целевого модуля)

Статья • 08.07.2024

Команда **.fiximports** проверяет и исправляет все статические ссылки импорта для целевого модуля.

```
dbgcmd
```

```
.fiximports Module
```

## Параметры

*Модуль*
Указывает целевой модуль, импорт которого выполняет исправление отладчика. *Модуль* может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис подстановочных знаков строки". Если вы хотите включить пробелы в *модуль*, необходимо заключить параметр в кавычки.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Режим пользователя, режим ядра |
| **Целевые объекты** | Только аварийный дамп (только minidump) |
| **Платформы** | Все |

## Замечания

Команду .fiximports **можно использовать** только в том случае, если целевой объект является мини-dump, который не содержит собственные исполняемые образы.

Когда отладчик сопоставляет изображения для использования в качестве памяти, отладчик не автоматически подключает импорт изображений к экспортерам. Поэтому инструкции, ссылающиеся на импорт, удаляются так же, как и в

динамическом сеансе отладки. С помощью .fiximports **можно запросить**, чтобы отладчик выполнял соответствующую компоновку импорта.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .flash_on_break (вспышка при разрыве)

Статья • 08.07.2024

Команда **.flash_on_break** указывает, мигает ли запись панели задач WinDbg при сворачивании WinDbg и целевых разрывов.

dbgcmd

```
.flash_on_break on
.flash_on_break off
.flash_on_break
```

## Параметры

**on**
Вызывает вспышку записи панели задач WinDbg, если WinDbg свернута, а целевой объект разбивается на отладчик. Это поведение по умолчанию для WinDbg.

**от**
Запрещает запись панели задач WinDbg мигать.

## Среда

Команда **.flash_on_break** доступна только в WinDbg. Эту команду нельзя использовать в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

При использовании **команды .flash_on_break** без параметров отладчик отображает текущий параметр флэш-памяти.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .fnent (данные функции отображения)

Статья • 08.07.2024

Команда **Fnent** отображает сведения о записи таблицы функций для указанной функции.

```
dbgcmd
```

```
.fnent Address
```

## Параметры

*Адрес*
Указывает адрес функции.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Алгоритм поиска символов для команды Fnent **совпадает с** алгоритмом ln (список ближайших символов). Сначала отображается ближайшие символы. Затем отладчик отображает запись функции для первого из этих символов.

Если ближайший символ не находится в таблице функций, информация не отображается.

В следующем примере показано возможное отображение.

```
dbgcmd
```

```
0:001> .fnent 77f9f9e7
Debugger function entry 00b61f50 for:
(77f9f9e7)   ntdll!RtlpBreakWithStatusInstruction   |  (77f9fa98)
ntdll!DbgPrintReturnControlC

Params:    1
Locals:    0
Registers: 0

0:001> .fnent 77f9fa98
Debugger function entry 00b61f70 for:
(77f9fa98)   ntdll!DbgPrintReturnControlC   |  (77f9fb21)   ntdll!DbgPrompt

Non-FPO

0:001> .fnent 01005a60
No function entry for 01005a60
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Fnret (возвращаемое значение функции отображения)

Статья • 08.07.2024

Команда **Fnret** отображает сведения о возвращаемом значении функции.

```
dbgcmd
```

```
.fnret [/s] Address [Value]
```

## Параметры

***/s***
**Задает $callret** псевдорегистрировать значение, отображаемое возвращаемое значение, включая сведения о типе.

*Адрес*
Указывает адрес функции.

*Value*
Указывает возвращаемое значение для отображения. Если включить *значение*, **Fnret** приведет *значение* к типу возвращаемой функции и отображает его в формате возвращаемого типа. Если не указано *значение*, отладчик получает возвращаемое значение функции из регистров возвращаемых значений.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Если включить *параметр Value* , **команда Fnret** приведет это значение только к соответствующему типу и отображает результаты.

Если не указано *значение*, отладчик использует регистры возвращаемого значения для определения этого значения. Если функция возвращалась недавно, чем функция, *указывающая параметр Address* , отображаемое значение, скорее всего, не будет значением, возвращаемым этой функцией.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .force_radix_output (использование Radix для целых чисел)

Статья • 08.07.2024

Команда **.force_radix_output** указывает, отображаются ли целые числа в десятичном формате или в радиксе по умолчанию.

```
dbgcmd
```

```
.force_radix_output 0
.force_radix_output 1
```

## Параметры

**0**
Отображает все целые числа (за исключением длинных целых чисел) в десятичном формате. Это поведение по умолчанию для отладчика.

**1**
Отображает все целые числа (за исключением длинных целых чисел) в радиксе по умолчанию.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Команда **.force_radix_output влияет на выходные данные** команды dt (тип отображения).

В WinDbg **.force_radix_output** также влияет на отображение в окне "Локальные" и в окне "Контроль". Вы можете выбрать или очистить **номера всегда отображаться в радиксе** по умолчанию в контекстном меню окна "Локальные" или "Просмотр", чтобы иметь тот же эффект, что **и .force_radix_output**. Эти окна автоматически обновляются после выполнения этой команды.

Команда **.force_radix_output** влияет только на отображение стандартных целых чисел. Чтобы указать, отображаются ли длинные целые числа в десятичном формате или радиксе по умолчанию, используйте **команду .enable_long_status (включить длинное целочисленное значение).**

Чтобы изменить радикс по умолчанию, используйте **команду n (Set Number Base).**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .force_tb (принудительное разрешение трассировки ветвей)

Статья • 26.10.2023

Команда **.force_tb** заставляет обработчик выполнять трассировку ветвей на ранних этапах процесса загрузки.

dbgcmd

```
.force_tb
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

Как правило, трассировка ветвей включается после того, как отладчик инициализирует блок управления процессором (PRCB). Эта инициализация выполняется на ранних этапах процесса загрузки.

Однако если перед этой инициализацией необходимо использовать команду **tb (Trace to Next Branch),** можно использовать команду **.force_tb** , чтобы включить трассировку ветвей раньше. Используйте эту команду осторожно, так как она может повредить состояние процессора.

# Форматы .форматов (отображение числовых форматов)

Статья • 08.07.2024

**Команда .форматирует** выражение или символ в контексте текущего потока и процесса и отображает его в нескольких числовых форматах.

```
dbgcmd

.formats expression
```

## Параметры

*выражение*
Указывает выражение для вычисления. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис числовых выражений".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Вычисляемое выражение отображается в шестнадцатеричных, десятичных, восьмимерных и двоичных форматах, а также в формате с плавающей запятой с двойной точностью и двойной точностью. Форматы символов ASCII также отображаются, если байты соответствуют стандартным символам ASCII. Выражение также интерпретируется как метка времени, если она находится в допустимом диапазоне.

В следующем примере показана **команда .форматов** .

dbgcmd

```
0:000> .formats 1c407e62
Evaluate expression:
  Hex:     1c407e62
  Decimal: 473988706
  Octal:   03420077142
  Binary:  00011100 01000000 01111110 01100010
  Chars:   .@~b
  Time:    Mon Jan 07 15:31:46 1985
  Float:   low 6.36908e-022 high 0
  Double:  2.34182e-315
```

Поле "**Время** " отображает 32-разрядное значение в формате метки времени CRT и отображает 64-разрядное значение в формате FILETIME. Эти форматы можно различать, так как формат FILETIME включает миллисекунд, а формат CRT не поддерживается.

## См. также

**? (Вычислять выражение)**

**?? (Оценка выражения C++ )**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .fpo (переопределения FPO элемента управления)

Статья • 08.07.2024

Команда **FPO управляет переопределением указателя кадров (FPO** ).

dbgcmd

```
.fpo -s [-fFlag] Address
.fpo -d Address
.fpo -x Address
.fpo -o Address
.fpo Address
```

## Параметры

**-s**
Задает переопределение FPO по указанному адресу.

**-f***Флаг*
Задает флаги FPO для переопределения. Не следует добавлять пробел между **-f** и *Flag*. Если флаг принимает аргумент, необходимо добавить пробел между флагом и этим аргументом. Если требуется несколько флагов, необходимо повторить **параметр -f (например, -fb -fp 4 -fe**). Параметр -f можно использовать **только с -s.** Для Flag *можно использовать одно из следующих значений*.

Развернуть таблицу

| **Флаг** | **Действие** |
|---|---|
| **b** | Задает **fUseBP** равным **TRUE**. |
| **e** | Задает **fUseSEH** равным **TRUE**. |
| **n** | Задает **cbFrame** равным FRAME_NONFPO. (По умолчанию cbFrame имеет значение FRAME_FPO.) |
| **l** *Термин* | Задает **cdwLocals**, равный термину. *Термин* должен указать нужное число локальных DWORD. |
| **Термин p** | Задает **cdwParams, равный** термину. *Термин* должен указать нужное число DWORD |

| Флаг | Действие |
| --- | --- |
| | параметров. |
| **Термин r** | Задает **cbRegs, равный** термину. *Термин* должен указать нужное число регистров. |
| *Термин* | Задает **cbProcSize** равный термину. *Термин* должен указать нужный размер процедуры. |
| **T** *Термин* | Задает **cbFrame**, равный термину. *Термин* должен указать один из следующих типов кадров:<br>• FRAME_FPO 0<br>• FRAME_TRAP 1<br>• FRAME_TSS 2<br>• FRAME_NONFPO 3 |

*Адрес*
Указывает адрес, в котором отладчик задает или удаляет переопределение или адрес, переопределяет отладчик. Этот адрес должен находиться в модуле в текущем списке модулей.

**-дн.**
Удаляет переопределение FPO по указанному адресу.

**-x**
Удаляет все переопределения FPO в модуле, который содержит *адрес адреса* .

**-o**
Отображает все переопределения FPO в модуле, который содержит *адрес адреса* .

# Среда

Развернуть таблицу

| Позиция | Description |
| --- | --- |
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Без параметров **команда FPO отображает переопределения FPO** для указанного адреса.

Некоторые компиляторы (включая Microsoft Visual Studio 6.0 и более ранние версии) создают сведения O FPO, чтобы указать, как настроен кадр стека. Во время перехода по стеку отладчик использует эти записи FPO для понимания стека. Если компилятор создал неверные сведения O FPO, можно использовать **команду FPO** для устранения этой проблемы.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .frame (установка локального контекста)

Статья • 26.10.2023

Команда **.frame** указывает, какой локальный контекст (область) используется для интерпретации локальных переменных или отображает текущий локальный контекст.

dbgcmd

```
.frame [/c] [/r] [FrameNumber]
.frame [/c] [/r] = BasePtr [FrameIncrement]
.frame [/c] [/r] = BasePtr StackPtr InstructionPtr
```

## Параметры

**/C**
Задает указанный кадр в качестве текущего локального контекста переопределения. Это действие позволяет пользователю получить доступ к энергонезависимому регистру для любой функции в стеке вызовов.

**/R**
Отображает регистры и другие сведения об указанном локальном контексте.

*FrameNumber*
Указывает номер кадра, локальный контекст которого требуется. Если этот параметр равен нулю, команда задает текущий кадр. Если этот параметр не указан, эта команда отображает текущий локальный контекст.

*BasePtr*
Указывает базовый указатель для трассировки стека, которая используется для определения кадра, если после имени команды (**.frame**) добавляется знак равенства (=). На процессоре на базе x86 вы добавляете еще один аргумент после *BasePtr* (который интерпретируется как *FrameIncrement*) или еще два аргумента после *BasePtr* (которые интерпретируются как *InstructionPtr* и *StackPtr*).

*FrameIncrement*
(только для процессоров x86)

Указывает дополнительное количество кадров после базового указателя. Например, если базовый указатель 0x0012FF00 является адресом кадра 3, команда

**.frame 12ff00** эквивалентна **.frame 3**, а **.frame 12ff00 2** — **.frame 5**.

*StackPtr*
(только для процессоров x86) Указывает указатель стека для трассировки стека, которая используется для определения кадра. Если опустить *StackPtr* и *InstructionPtr*, отладчик использует указатель стека, который указывает регистр **esp** , и указатель инструкции, который указывает регистр **eip** .

*ИнструкцияPtr*
(только для процессоров x86) Указывает указатель инструкции для трассировки стека, которая используется для определения кадра. Если опустить *StackPtr* и *InstructionPtr*, отладчик использует указатель стека, который указывает регистр **esp** , и указатель инструкции, который указывает регистр **eip** .

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения о локальном контексте и других параметрах контекста см. в разделе Изменение контекстов. Дополнительные сведения о том, как отображать локальные переменные и другие команды, связанные с памятью, см. в разделе Чтение и запись памяти.

# Комментарии

При выполнении приложения значение локальных переменных зависит от расположения счетчика программы, так как область таких переменных распространяется только на функцию, в которую они определены. Если команда **.frame** не используется, отладчик использует область текущей функции (текущий кадр в стеке) в качестве локального контекста.

Чтобы изменить локальный контекст, используйте команду **.frame** и укажите нужный номер кадра.

*Номер кадра* — это положение кадра стека в трассировке стека. Эту трассировку стека можно просмотреть с помощью команды **k (Display Stack Backtrace)** или окна Вызовы. Первая строка (текущий кадр) — это номер кадра 0. Последующие строки представляют собой номера кадров 1, 2, 3 и т. д.

При использовании параметра **n** с командой **k** команда **k** отображает номера кадров вместе с трассировкой стека. Эти номера кадров всегда отображаются в шестнадцатеричной форме. С другой стороны, команда **.frame** интерпретирует свой аргумент в радиксе по умолчанию, если этот параметр не переопределен префиксом, например 0x. Чтобы изменить радикс по умолчанию, используйте команду **n (задать числовую базу).**

Вы можете задать для локального контекста другой кадр стека, чтобы вы могли просматривать новые сведения о локальных переменных. Однако фактические доступные переменные зависят от выполняемого кода.

Локальный контекст сбрасывается до область счетчика программы, если выполняется какое-либо приложение. При изменении контекста регистра локальный контекст сбрасывается в верхний кадр стека.

# .help (справка мета-команд)

Статья • 08.07.2024

Команда **.help** отображает список всех метакоманд.

```
dbgcmd
```

```
.help
.help /D
```

## Параметры

**/D**
Отображает выходные данные с помощью языка разметки отладчика. Выходные данные содержат список ссылок, которые можно щелкнуть, чтобы просмотреть метакоманды, начинающиеся с определенной буквы.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Дополнительные сведения о мета-командах см. в команде **.help** . Дополнительные сведения о стандартных командах используйте **команду ?** . Дополнительные сведения о командах расширения см. в **расширении !help** .

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# .hh (open HTML Help File)

Статья • 05.04.2024

Команда **HH** открывает документацию по средствам отладки для Windows.

```
dbgcmd
```

```
.hh [Text]
```

## Параметры

*Текст*
Указывает текст, который нужно найти в индексе документации справки.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

Эту команду нельзя использовать при выполнении удаленной отладки с помощью Remote.exe.

## Дополнительная информация

Дополнительные сведения о документации справки см. в разделе "Использование файла справки".

## Замечания

Команда **HH** открывает средства отладки для Windows (Debugger.chm). Если указать *текст*, отладчик открывает **область индекса** в документации и ищет *текст* в качестве ключевое слово в индексе. Если текст не указан, отладчик открывает **область "Содержимое"** документации.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .hideinjectedcode (скрытие внедренного кода)

Статья • 08.07.2024

Команда **.hideinjectedcode** включает или отключает скрытие внедренного кода.

dbgcmd

```
.hideinjectedcode { on | off }
.hideinjectedcode help
.hideinjectedcode
```

## Параметры

**on**

Включает скрытие внедренного кода.

**от** ****

Отключение скрытия внедренного кода

**Справка**

Отображает справку по этой команде.

## Замечания

Если вы вводите эту команду без параметра, она отображает справку по команде.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .holdmem (удержание и сравнение памяти)

Статья • 08.07.2024

Команда **.holdmem** сохраняет диапазоны памяти и сравнивает их с другими диапазонами памяти.

dbgcmd

```
.holdmem -a Range
.holdmem -d { Range | Address }
.holdmem -D
.holdmem -o
.holdmem -c Range
```

## Параметры

**-a** **** *Диапазон*
Задает диапазон памяти для сохранения. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

**-d** *** { *адрес диапазона* | }
Указывает диапазоны памяти для удаления. Если указать *адрес*, отладчик удаляет любой сохраненный диапазон, содержащий этот адрес. Если указать *диапазон*, отладчик удаляет любой сохраненный диапазон, перекрывающийся с *диапазоном*. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

**-D**
Удаляет все сохраненные диапазоны памяти.

**-o**
Отображает все сохраненные диапазоны памяти.

**-c** *Диапазон*
Сравнивает указанный диапазон со всеми сохраненными диапазонами памяти. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

## Среда

Развернуть таблицу

| Позиция | Description |
| --- | --- |
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения о том, как управлять памятью и описанием других команд, связанных с памятью, см. в разделе "Чтение и запись памяти".

# Замечания

Команда **.holdmem** сравнивает диапазоны памяти байтами для байтов.

Если ни одно из указанных расположений памяти не существует в виртуальном адресном пространстве, команда возвращает ошибку.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .idle_cmd (установка команды простоя)

Статья • 05.04.2024

Команда **.idle_cmd** задает команду *простоя*. Это команда, которая выполняется всякий раз, когда элемент управления возвращается из целевого объекта в отладчик. Например, когда целевой объект достигает точки останова, эта команда выполняется.

dbgcmd

```
.idle_cmd
.idle_cmd String
.idle_cmd /d
```

## Параметры

*String*
Указывает строку, для которой должна быть задана команда простоя.

**/d**
Очищает команду простоя.

## Среда

Эта команда не может использоваться в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

При **использовании .idle_cmd** без параметров отображается текущая команда простоя.

В WinDbg команды простоя хранятся в рабочих областях.

Ниже приведен пример. Для команды простоя задано **значение rax**. Затем, так как отладчик уже неактивен, эта команда немедленно выполняется, отображая **регистр eax** :

dbgcmd

```
windbg> .idle_cmd r eax
Execute when idle: r eax
eax=003b0de8
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .ignore_missing_pages (подавление отсутствующих ошибок страницы)

Статья • 20.05.2024

Команда **.ignore_missing_pages** подавляет сообщения об ошибках, когда дамп памяти ядра содержит отсутствующие страницы.

dbgcmd

```
.ignore_missing_pages 0
.ignore_missing_pages 1
.ignore_missing_pages
```

## Параметры

**0**
Отображает все отсутствующие ошибки страниц при отладке дампа памяти ядра. Это поведение отладчика по умолчанию.

**1**
Подавляет отображение отсутствующих ошибок страниц при отладке дампа памяти ядра.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | Только отладка файла дампа |
| **Платформы** | Все |

## Дополнительная информация

Дополнительные сведения об отладке этих файлов дампа см. в разделе "Дамп памяти ядра".

## Замечания

Без параметров .ignore_missing_pages отображает текущее состояние этого параметра.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .inline (переключение встроенной функции отладки)

Статья • 08.07.2024

**Встроенная** команда включает или отключает встроенную отладку функции.

dbgcmd

```
.inline 0
.inline 1
```

## Параметры

**0**

Отключает встроенную отладку функции.

**1**

Включает встроенную отладку функции.

## См. также

Отладка оптимизированного кода и встроенных функций

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .imgscan (поиск заголовков изображений)

Статья • 05.04.2024

Команда **.imgscan** сканирует виртуальную память для заголовков изображений.

```
dbgcmd

.imgscan [Options]
```

## Параметры

*Параметры* любого из следующих параметров:

**/r** **** *Range*
Задает диапазон для поиска. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов". Если указать только один адрес, отладчик выполняет поиск диапазона, который начинается с этого адреса и расширяет 0x10000 байтов.

**/l**
Загружает сведения о модуле для любого найденного заголовка образа.

**/v**
Отображает подробные сведения.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Если параметр /r **не используется**, отладчик выполняет поиск по всем регионам виртуальной памяти.

Команда **imgscan** отображает все заголовки изображений, которые он находит, и тип заголовка. Типы заголовков включают заголовки переносимого исполняемого файла (PE) и заголовки Microsoft MS-DOS MZ.

В следующем примере показана **команда .imgscan** .

dbgcmd

```
0:000> .imgscan
MZ at 00400000, prot 00000002, type 01000000 - size 2d000
MZ at 77f80000, prot 00000002, type 01000000 - size 7d000
  Name: ntdll.dll
MZ at 7c570000, prot 00000002, type 01000000 - size b8000
  Name: KERNEL32.dll
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .jdinfo (использование JIT_DEBUG_INFO)

Статья • 08.07.2024

Команда **Jdinfo** использует JIT_DEBUG_INFO структуру в качестве источника исключения и контекста для отладки JIT. Адрес структуры передается **команде Jdinfo** с помощью параметра %p, указанного в записи реестра AeDebug.

Дополнительные сведения об используемых разделах реестра см. в разделе "Включение отладки postmortem". Дополнительные сведения о контекстах регистрации см. в разделе "Изменение контекстов".

dbgcmd

```
.jdinfo Address
```

## Параметры

*Адрес*
Задает адрес структуры JIT_DEBUG_INFO. Адрес структуры передается **команде Jdinfo** с помощью параметра %p, указанного в записи реестра AeDebug.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Режим пользователя |
| **Целевые объекты** | Динамический, аварийный дампа |
| **Платформы** | Все |

## Пример

В этом примере показано, как можно настроить запись реестра AeDebug для использования WinDbg в качестве отладчика JIT.

dbgcmd

```
Debugger = "Path\WinDbg.EXE -p %ld -e %ld -c ".jdinfo 0x%p"
```

Затем при сбое вызывается настроенный отладчик JIT, а параметр %p используется для передачи адреса структуры **JIT_DEBUG_INFO команде Jdinfo** , выполняемой после запуска отладчика.

dbgcmd

```
nMicrosoft (R) Windows Debugger Version 10.0.10240.9 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

*** wait with pending attach
Executable search path is:
...
ModLoad: 00000000`68a20000 00000000`68ac3000
C:\WINDOWS\WinSxS\amd64_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.9247_n
one_08e394a1a83e212f\MSVCR90.dll
(153c.5d0): Break instruction exception - code 80000003 (first chance)
Processing initial command '.jdinfo 0x00000000003E0000'
ntdll!DbgBreakPoint:
00007ffc`81a986a0 cc              int     3
0:003> .jdinfo 0x00000000003E0000
----- Exception occurred on thread 0:15c8
ntdll!ZwWaitForMultipleObjects+0x14:
00007ffc`81a959a4 c3              ret

----- Exception record at 00000000`003e0028:
ExceptionAddress: 00007ff791d81014 (CrashAV_x64!wmain+0x0000000000000014)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000001
   Parameter[1]: 0000000000000000
Attempt to write to address 0000000000000000

----- Context record at 00000000`003e00c0:
rax=0000000000000000 rbx=0000000000000000 rcx=00007ffc81a954d4
rdx=0000000000000000 rsi=0000000000000000 rdi=0000000000000001
rip=00007ff791d81014 rsp=00000000006ff8b0 rbp=0000000000000000
 r8=00000000006ff808  r9=0000000000000000 r10=0000000000000000
r11=0000000000000000 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei pl zr na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b
efl=00010246
CrashAV_x64!wmain+0x14:
00007ff7`91d81014 45891b          mov     dword ptr [r11],r11d
ds:00000000`00000000=????????
```

# Замечания

Команда Jdinfo **использует сведения о реестре AeDebug**, представленные в Windows Vista. Дополнительные сведения об используемых разделах реестра см. в разделе "Включение отладки postmortem". Команда **Jdinfo** принимает адрес JIT_DEBUG_INFO, настроенный системой для **AeDebug** , и задает контекст исключения, вызвавшего сбой.

Вы можете использовать **команду Jdinfo** вместо **-g** в **AeDebug, чтобы отладчик задал состояние AeDebug** без необходимости выполнения.

Это состояние может быть выгодно, так как в обычных условиях при возникновении исключения в пользовательском режиме происходит следующая последовательность:

1. Операционная система Microsoft Windows останавливает выполнение приложения.
2. Запущен отладчик postmortem.
3. Отладчик подключается к приложению.
4. Отладчик выдает команду Go. (Эта команда вызвана ключом **-g в ключе AeDebug**.)
5. Целевой объект пытается выполнить и может или не столкнуться с тем же исключением.
6. Это исключение разбивается на отладчик.

Существует несколько проблем, которые могут возникнуть из-за этих событий:

- Исключения не всегда повторяются, возможно, из-за временных условий, которые больше не существуют при перезапуске исключения.
- Может произойти другое событие, например другое исключение. Нет способа знать, идентично ли оно исходному событию.
- Присоединение отладчика включает внедрение нового потока, который может быть заблокирован, если поток держит блокировку загрузчика. Внедрение нового потока может быть значительным нарушением процесса.

Если вы используете **-c .jdinfo** вместо **-g** в **ключе AeDebug** , выполнение не происходит. Вместо этого сведения об исключении извлекаются из структуры JIT_DEBUG_INFO с помощью переменной %p.

Например, рассмотрим следующий **ключ AeDebug** .

dbgcmd

```
ntsd -p %ld -e %ld -c ".jdinfo 0x%p"
```

Следующий пример является еще менее инвазивным. **Переключатель -pv** приводит к тому, что отладчик прикрепляет невиновно, что не внедряет новые потоки в целевой объект.

dbgcmd

```
ntsd -pv -p %ld -e %ld -c ".jdinfo 0x%p"
```

Если вы используете этот неинсивный параметр, выход из отладчика не завершает процесс. Для завершения процесса можно использовать **команду kill (Kill Process** ).

Если вы хотите использовать это для отладки файлов дампа, следует использовать **.dump /j** , чтобы добавить структуру JIT_DEBUG_INFO в файл дампа при создании файла дампа.

Структура JIT_DEBUG_INFO определяется следующим образом.

dbgcmd

```
typedef struct _JIT_DEBUG_INFO {
    DWORD dwSize;
    DWORD dwProcessorArchitecture;
    DWORD dwThreadID;
    DWORD dwReserved0;
    ULONG64 lpExceptionAddress;
    ULONG64 lpExceptionRecord;
    ULONG64 lpContextRecord;
} JIT_DEBUG_INFO, *LPJIT_DEBUG_INFO;
```

Для отображения структуры JIT_DEBUG_INFO можно использовать команду dt.

dbgcmd

```
0: kd> dt JIT_DEBUG_INFO
nt!JIT_DEBUG_INFO
   +0x000 dwSize           : Uint4B
   +0x004 dwProcessorArchitecture : Uint4B
   +0x008 dwThreadID       : Uint4B
  +0x00c dwReserved0      : Uint4B
   +0x010 lpExceptionAddress : Uint8B
```

```
   +0x018 lpExceptionRecord : Uint8B
   +0x020 lpContextRecord  : Uint8B
```

### Просмотр записи исключений, стека вызовов и LastEvent с помощью WinDbg

После использования команды jdinfo для задания контекста моменту сбоя можно просмотреть запись исключения, возвращенную jdinfo, стек вызовов и lastevent, как показано ниже, для изучения причины.

dbgcmd

```
0:000> .jdinfo  0x00000000003E0000
----- Exception occurred on thread 0:15c8
ntdll!NtWaitForMultipleObjects+0x14:
00007ffc`81a959a4 c3              ret

----- Exception record at 00000000`003e0028:
ExceptionAddress: 00007ff791d81014 (CrashAV_x64!wmain+0x0000000000000014)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000001
   Parameter[1]: 0000000000000000
Attempt to write to address 0000000000000000
...

0:000> k
  *** Stack trace for last set context - .thread/.cxr resets it
# Child-SP          RetAddr           Call Site
00 00000000`006ff8b0 00007ff7`91d811d2 CrashAV_x64!wmain+0x14
[c:\my\my_projects\crash\crashav\crashav.cpp @ 14]
01 00000000`006ff8e0 00007ffc`7fa38364 CrashAV_x64!__tmainCRTStartup+0x11a
[f:\dd\vctools\crt_bld\self_64_amd64\crt\src\crtexe.c @ 579]
02 00000000`006ff910 00007ffc`81a55e91 KERNEL32!BaseThreadInitThunk+0x14
03 00000000`006ff940 00000000`00000000 ntdll!RtlUserThreadStart+0x21

0:000> .lastevent
Last event: 153c.5d0: Break instruction exception - code 80000003 (first
chance)
  debugger time: Thu Sep  8 12:55:08.968 2016 (UTC - 7:00)
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .kdfiles (установка карты замены драйвера)

Статья • 08.07.2024

Команда **Kdfiles** считывает файл и использует его содержимое в качестве карты замены драйвера.

dbgcmd

```
.kdfiles MapFile
.kdfiles -m OldDriver NewDriver
.kdfiles -s SaveFile
.kdfiles -c
.kdfiles
```

## Параметры

*MapFile*
Указывает файл карты замены драйвера для чтения.

**-m**
Добавляет сопоставление замены драйвера в текущий список связей.

*OldDriver*
Указывает путь и имя файла предыдущего драйвера на целевом компьютере. Синтаксис OldDriver совпадает с синтаксисом первой строки после **сопоставления** в файле замены драйвера. Дополнительные сведения об этом синтаксисе см. в разделе "Файлы драйверов сопоставления".

*NewDriver*
Указывает путь и имя файла нового драйвера. Этот драйвер может находиться на хост-компьютере или в другом сетевом расположении. Синтаксис NewDriver *совпадает с синтаксисом* второй строки после **сопоставления** в файле замены драйвера. Дополнительные сведения об этом синтаксисе см. в разделе "Файлы драйверов сопоставления".

**-s**
Создает файл и записывает в этот файл связи на замену текущего драйвера.

*SaveFile*
Указывает имя создаваемого файла.

**-c**
Удаляет существующую карту замены драйвера. (Этот параметр не изменяет сам файл карты. Вместо этого этот параметр очищает текущие параметры карты отладчика.)

# Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Процессоры на основе x86 |

# Дополнительная информация

Дополнительные сведения о замене драйверов и замене других модулей в режиме ядра, описание формата файлов карты замены драйверов и ограничений для использования этой функции см. в разделе "Файлы драйверов сопоставления".

# Замечания

Если вы используете **команду Kdfiles** без параметров, отладчик отображает путь и имя текущего файла карты замены драйвера и текущий набор сопоставлений замены.

При выполнении этой команды указанный *файл MapFile*считывается. Если файл не найден или не содержит текст в правильном формате, отладчик отображает сообщение, которое указывает "Не удается загрузить связи файлов".

Если указанный файл находится в правильном формате файла карты замены драйвера, отладчик загружает содержимое файла и использует их в качестве карты замены драйвера. Эта карта остается до выхода отладчика или до тех пор, пока не будет выполнена другая **команда Kdfiles** .

После чтения файла карта замены драйвера не влияет на последующие изменения файла (если за этими изменениями не следует другая **команда Kdfiles** ).

## Замена файла в режиме пользователя

Замена файлов в режиме пользователя добавлена в Windows версии 2004. Эта поддержка позволяет заменить следующие файлы пользовательского режима на KDFILE.

- Библиотеки DLL в режиме пользователя (включая NTDLL и Известные значения)
- EXEs в пользовательском режиме, которые являются основным образом процесса для CreateProcess

Чтобы использовать поддержку kdfiles в пользовательском режиме, необходимо сначала включить загрузку символов ядра с помощью `!gflag +ksl` команды отладчика или настроить глобальные флаги ksl в реестре. Дополнительные сведения о gflag см. в разделе !gflag.

В следующих примерах показано общее использование.

dbgcmd

```
.kdfiles -m system32\userdll C:\myfiles\my_native_userdll.dll
.kdfiles -m system32\userdll \\server\share\my_native_userdll.dll
.kdfiles -m syswow64\ntdll.dll \\server\share\my_x86_wow64_ntdll.dll
.kdfiles -m system32\userbase.dll \\server\share\my_native_userbase.dll
```

Kdfiles в пользовательском режиме игнорирует ошибки в соответствии с файлом и не отображает сообщение об ошибке при возникновении сбоя.

Будьте осторожны, чтобы соответствующим образом квалифицировать пути kdfiles для KDfile в пользовательском режиме. Это плохая идея просто соответствовать ntdll.dll (вместо system32\ntdll.dll), так как в противном случае Wow64 NTDLL будет заменен собственным. Аналогичные ситуации могут возникнуть с другими неоднозначными совпадениями подстроки.

После сборки 20172 механизм Kdfiles в пользовательском режиме попытается извлечь файлы из отладчика до тех пор, пока одна попытка не завершится ошибкой; Затем имя файла, которое не удалось извлечь, не будет выполнено повторно для сеанса загрузки без ручного вмешательства от отладчика для изменения состояния целевой системы. В предыдущих сборках механизм Kdfiles пользовательского режима будет предпринять одну попытку (успешно или нет), чтобы извлечь заданное имя файла на сеанс загрузки. Эти политики снижают затраты на обмен данными с отладчиком для файлов, которые не находятся в списке kdfiles или недоступны для замены, например из-за нарушений общего доступа от процессов, которые, возможно, уже загружали заданный файл. Из-за

этого обычно рекомендуется настроить все файлы для извлечения в списке Kdfiles заранее, прежде чем они сначала будут ссылаться.

Помните об ограничениях, которые не удается заменить уже в файлах диска и т. д. Так как многие системные библиотеки DLL не будут легко переключаться после первоначальной загрузки, предустановите параметр gflags +ksl и используйте Kdfiles для замены любых двоичных файлов в режиме пользователя прямо во время загрузки.

Дополнительные сведения о включении отладки загрузки см. в статье BCDEdit /bootdebug.

Использование KDNET транспорта KDNET с высокой скоростью и низкой задержкой рекомендуется минимизировать влияние на производительность системы.

# Требования

Развернуть таблицу

| | |
|---|---|
| Версия | Поддерживается в Windows XP и более поздних версиях операционной системы Windows. |

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .kdtargetmac (ОТОБРАЖЕНИЕ ЦЕЛЕВОго MAC-адреса)

Статья • 01.02.2024

Отображение целевого MAC-адреса

```
dbgcmd
```

```
.kdtargetmac
```

# Параметры

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Только в режиме ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | все |

## Дополнительные сведения

Используйте команду .kdtargetmac, чтобы отобразить MAC-адрес (управление доступом к мультимедиа) целевой системы.

```
dbgcmd
```

```
0: kd> .kdtargetmac

The target machine MAC address in open-device format is: XXXXXXXXXXXX
```

Команда .kdtargetmac is доступна, если KDNET включена в целевой системе. Используйте команду BCDEdit с параметром /dbgsettings, чтобы отобразить конфигурацию в целевой системе. Тип отладки *NET* указывает, что настроенA сеть KDNET.

```
dbgcmd
```

```
C:\WINDOWS\system32>bcdedit /dbgsettings
key                     1.2.3.4
debugtype               NET
hostip                  192.168.1.10
port                    50000
dhcp                    Yes
The operation completed successfully.
```

Дополнительные сведения см. в разделе Настройка отладки ядра сети KDNET вручную.

# Комментарии

Знание MAC-адреса целевой системы может быть полезно для трассировки сети и других служебных программ.

# .kframes (Set Stack Length)

Статья • 26.10.2023

Команда **.kframes** задает длину по умолчанию для отображения трассировки стека.

```
dbgcmd
```

```
.kframes FrameCountDefault
```

## Параметры

*FrameCountDefault*

Указывает количество кадров стека, отображаемых при использовании команды трассировки стека.

### Среда

| **Среда** | |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Комментарии

Чтобы задать длину по умолчанию для отображения трассировки стека, можно использовать команду **.kframes** . Эта длина определяет количество кадров, отображаемых командами **k, kb, kp, kP и kv,** и числом DWORD_PTRs, отображаемых командой **kd** .

Эту длину по умолчанию можно переопределить с помощью параметров *FrameCount* или *WordCount* для этих команд.

Используйте команду **.kframes** без параметров, чтобы просмотреть текущее значение.

```
dbgcmd
```

```
.kframes
Default stack trace depth is 0n256 frames
```

Используйте следующую команду, чтобы задать глубину трассировки стека 0x2.

dbgcmd

```
.kframes 0x2
Default stack trace depth is 0n2 frames
```

Используйте следующие команды для отображения двух значений стека и двух необработанных значений стека.

dbgcmd

```
k
 # Child-SP          RetAddr               Call Site
00 00000054`b71ffb78 00007ffe`1ee672ae     ntdll!DbgBreakPoint
01 00000054`b71ffb80 00007ffe`1e2a3e2d     ntdll!DbgUiRemoteBreakin+0x4e

kd
00000054`b71ffb70  00000000
00000054`b71ffb74  00000000
```

Укажите *FrameCount* и *WordCount* для отображения дополнительных значений.

dbgcmd

```
k 3
 # Child-SP          RetAddr               Call Site
00 00000054`b71ffb78 00007ffe`1ee672ae     ntdll!DbgBreakPoint
01 00000054`b71ffb80 00007ffe`1e2a3e2d     ntdll!DbgUiRemoteBreakin+0x4e
02 00000054`b71ffbb0 00007ffe`1eddef48     KERNEL32!BaseThreadInitThunk+0x1d

kd 5
00000054`b71ffb70  00000000
00000054`b71ffb74  00000000
00000054`b71ffb78  1ee672ae
00000054`b71ffb7c  00007ffe
00000054`b71ffb80  00000000
```

# См. также раздел

**k, kb, kc, kd, kp, kP, kv (отображение стека backtrace)**

# .kill (Kill Process)

Статья • 05.04.2024

В пользовательском режиме **команда .kill** завершает процесс отладки.

В режиме **ядра команда .kill** завершает процесс на целевом компьютере.

Синтаксис пользовательского режима

```
dbgcmd

.kill [ /h | /n ]
```

Синтаксис режима ядра

```
dbgcmd

.kill Process
```

## Параметры

**/h**
(Только в пользовательском режиме) Любое выдающееся событие отладки будет продолжено и отмечено как обработанное. Это значение по умолчанию.

**/N**
(Только в пользовательском режиме) Любое выдающееся событие отладки будет продолжено без пометки как обработанное.

*Обработать*
Указывает адрес процесса, который необходимо завершить. Если *процесс* опущен или ноль, процесс по умолчанию для текущего состояния системы будет завершен.

## Среда

В режиме ядра эта команда поддерживается в Microsoft Windows Server 2003 и более поздних версиях Windows.

Развернуть таблицу

| | |
|---|---|
| **Режимы** | режим пользователя, режим ядра |

| | |
|---|---|
| **Целевые объекты** | только динамическая отладка |
| **Платформы** | all |

# Замечания

В пользовательском режиме эта команда завершает процесс отладки. Если отладчик подключен к дочернему процессу, можно использовать **.kill** , чтобы завершить дочерний процесс, не завершив родительский процесс. Дополнительные сведения см. в примерах.

В режиме ядра эта команда планирует выбранный процесс на целевом компьютере для завершения. При следующем запуске целевого объекта (например, с помощью команды g (Go) **указанный** процесс завершается.

Эту команду нельзя использовать во время локальной отладки ядра.

# Примеры

## Использование .childdbg

Предположим, что перед созданием дочернего процесса вы присоединяете отладчик к родительскому процессу (Parent.exe). Вы можете ввести команду **.childdbg 1** , чтобы сообщить отладчику присоединиться к любому дочернему процессу, который создает родительский файл.

dbgcmd

```
1:001> .childdbg 1
Processes created by the current process will be debugged
```

Теперь позвольте родительскому процессу запустить и разорвать его после создания дочернего процесса. **Использование | (Состояние процесса)** команда, чтобы просмотреть номера процесса для родительских и дочерних процессов.

dbgcmd

```
0:002> |*
.  0    id: 7f8 attach  name: C:\Parent\x64\Debug\Parent.exe
   1    id: 2d4 child   name: notepad.exe
```

В предыдущих выходных данных число дочерних процессов (notepad.exe) равно 1. Точка (.) в начале первой строки сообщает нам, что родительский процесс является текущим процессом. Чтобы сделать дочерний процесс текущим процессом, введите **|1s**.

dbgcmd

```
0:002> |1s
...
1:001> |*
#  0    id: 7f8 attach  name: C:\Parent\x64\Debug\Parent.exe
.  1    id: 2d4 child   name: notepad.exe
```

Чтобы убить дочерний процесс, введите команду **kill**. Родительский процесс продолжает выполняться.

dbgcmd

```
1:001> .kill
Terminated.  Exit thread and process events will occur.
1:001> g
```

## Использование параметра -o

При запуске WinDbg или CDB можно использовать **параметр -o** , чтобы сообщить отладчику, что он должен присоединиться к дочерним процессам. Например, следующая команда запускает WinDbg, которая запускается и подключается к Parent.exe. Когда Parent.exe создает дочерний процесс, WinDbg присоединяется к дочернему процессу.

**windbg -g -G -o -o Parent.exe**

Дополнительные сведения см. в разделе "**Параметры** командной строки WinDbg" и "**Параметры** командной строки CDB".

# Требования

Развернуть таблицу

| | |
|---|---|
| Версия | Версии:(Режим ядра) поддерживается в Windows Server 2003 и более поздних версиях. |

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .lastevent (отображение последнего события)

Статья • 26.10.2023

Команда **.lastevent** отображает последнее исключение или событие, которое произошло.

dbgcmd

```
.lastevent
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения об исключениях и событиях см. в разделе Управление исключениями и событиями.

# Комментарии

При переходе к отладчику всегда создается исключение. Всегда возникает *последнее событие* , когда отладчик принял входные данные команды. Если вы включаетесь в отладчик с помощью **клавиш CTRL+C**, **CTRL+BREAK** или Отладка | Break— создается код исключения 0x80000003.

# .line (поддержка исходной строки переключения)

Статья • 08.07.2024

Команда **.line** включает или отключает поддержку сведений о исходной строке.

```
dbgcmd
```

```
.lines [-e|-d|-t]
```

## Параметры

**-e**
Включает поддержку исходной строки.

**-d**
Отключает поддержку исходной строки.

**-t**
Включает или отключает поддержку исходной строки. Если параметры для **строк не указаны, поведение команды .line** по умолчанию — это переключение поддержки исходной строки.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения об отладке источников и связанных командах см. в разделе "Отладка в режиме источника".

# Замечания

Прежде чем выполнять отладку на уровне источника, необходимо включить поддержку исходной строки. Эта поддержка позволяет отладчику загружать символы исходной строки.

Вы можете включить поддержку исходной **строки с помощью команды .line или параметра командной строки -line**. Если поддержка исходной **строки уже включена, с помощью команды .line** эта поддержка отключается.

По умолчанию, если команда .line **не используется**, WinDbg включает поддержку исходной строки, а отладчики консоли (KD, CDB, NTSD) отключают поддержку. Дополнительные сведения об изменении этого параметра см. в разделе "Параметры символов".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .load, .loadby (DLL расширения загрузки)

Статья • 05.04.2024

**Команды load** и **.loadby** загружают новую библиотеку DLL расширения в отладчик.

dbgcmd

```
.load DLLName
!DLLName.load
.loadby DLLName ModuleName
```

## Параметры

*DLLName*
Указывает библиотеку DLL расширения отладчика для загрузки. Если вы используете команду .load**, *DLLName* должно содержать полный** путь. Если вы используете **команду loadby** , *dllName* должно содержать только имя файла.

*Имя модуля*
Указывает имя модуля, расположенного в том же каталоге, что и библиотека DLL *расширения, указываемая DLL* .

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о загрузке, выгрузке и управлении расширениями см. в разделе "Загрузка библиотек DLL расширений отладчика".

# Замечания

При использовании команды .load **необходимо указать полный** путь.

При использовании **команды .loadby** не укажите путь. Вместо этого отладчик находит модуль, который *указывает параметр ModuleName* , определяет путь этого модуля, а затем использует этот путь при загрузке отладчиком библиотеки DLL расширения. Если отладчик не может найти модуль или не удается найти библиотеку DLL расширения, вы получите сообщение об ошибке, указывающее проблему. Между указанным модулем и библиотекой DLL расширения не требуется никакой связи. **Поэтому использование команды .loadby** — это просто способ избежать ввода длинного пути.

**После завершения команды .load** или **.loadby** можно получить доступ к командам, хранящимся в загруженном расширении.

Чтобы загрузить библиотеку DLL расширения, можно выполнить одно из следующих действий:

- Используйте команду load или **.loadby**.
- Выполнение расширения путем выдачи полного **!***DLLName*.*Синтаксис ExtensionCommand* . Если отладчик еще не загружен *dllName*.dll, он загружает библиотеку DLL в данный момент, если он находится в текущем пути поиска DLL.

Используйте команду цепочки для отображения сведений о загруженной и текущей схеме поиска библиотеки DLL.

dbgcmd

```
0:000> .chain
Extension DLL search Path:
    C:\Program Files (x86)\Windows Kits\10\Debuggers\x64\WINXP;C:\Program
Files (x86)\Windows Kits\10\Debuggers\x64\winext;C:\Program Files
(x86)\Windows Kits\10\Debuggers\x64\winext\arcade;C:\Program Files
(x86)\Windows Kits\10\Debuggers\x64\pri;C:\Program Files (x86)\Windows
Kits\10\Debuggers\x64;
Extension DLL chain:
    C:\Windows\Microsoft.NET\Framework64\v4.0.30319\SOS.dll: image
4.8.4084.0, API 1.0.0, built Sun Nov 24 00:38:52 2019
```

Например, управляемый код SOS.dll не находится в пути поиска для библиотек DLL, показанных выше, поэтому используйте команду .load с полным путем для загрузки этой библиотеки DLL.

dbgcmd

```
0:000> .load C:\Windows\Microsoft.NET\Framework64\v4.0.30319\SOS.dll
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Языковой стандарт (настройка языкового стандарта)

Статья • 08.07.2024

Команда **locale** задает языковой стандарт или отображает текущий языковой стандарт.

```
dbgcmd
```

```
.locale [Locale]
```

## Параметры

*Место действия*
Указывает нужный языковой стандарт. Если этот параметр не указан, отладчик отображает текущий языковой стандарт.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о языковом стандарте см. на **странице справки по подпрограмме setlocale** .

## Замечания

Языковой стандарт управляет отображением строк Юникода.

В следующих примерах показана **команда locale** .

dbgcmd

```
kd> .locale
Locale: C

kd> .locale E
Locale: English_United States.1252

kd> .locale c
Locale: Catalan_Spain.1252

kd> .locale C
Locale: C
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .logappend (добавление файла журнала)

Статья • 08.07.2024

Команда **.logappend** добавляет копию событий и команд из окна команды отладчика к указанному файлу журнала.

dbgcmd

```
.logappend [/u] [FileName]
```

## Параметры

**/u**
Записывает файл журнала в формате Юникода. Если этот параметр не указан, отладчик записывает файл журнала в формате ASCII (ANSI).

**Обратите внимание** , что при добавлении к существующему файлу журнала следует использовать **параметр /u** , только если вы создали файл журнала с помощью **параметра /u** . В противном случае файл журнала будет содержать символы ASCII и Юникода, что может затруднить чтение.

*Имя файла*
Указывает имя файла журнала. Можно указать полный путь или только имя файла. Если имя файла содержит пробелы, заключите *Имя файла* в кавычки. Если путь не указан, отладчик использует текущий каталог. Если не указано *имя файла*, отладчик называет файл Dbgeng.log.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Если при выполнении **команды logappend** уже открыт файл журнала, отладчик закрывает файл журнала. Если указать имя файла, который уже существует, отладчик добавляет новые сведения в файл. Если файл не существует, отладчик создает его.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# LOGCLOSE (закрыть файл журнала)

Статья • 26.10.2023

Команда **.logclose** закрывает любой открытый файл журнала.

```
dbgcmd

    .logclose
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# .logfile (отображение состояния файла журнала)

Статья • 26.10.2023

Команда **.logfile** определяет, существует ли файл журнала, и отображает состояние файла.

```
dbgcmd

    .logfile
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# .logopen (открыть файл журнала)

Статья • 08.07.2024

Команда **.logopen** отправляет копию событий и команд из окна команды отладчика в новый файл журнала.

dbgcmd

```
.logopen [Options] [FileName]
.logopen /d
```

## Параметры

*Параметры* любого из следующих параметров:

**/t**
Добавляет идентификатор процесса с текущей датой и временем к имени файла журнала. Эти данные вставляются после имени файла и перед расширением имени файла.

**/u**
Записывает файл журнала в формате Юникода. Если этот параметр не указан, отладчик записывает файл журнала в формате ASCII (ANSI).

*Имя файла*
Указывает имя файла журнала. Можно указать полный путь или только имя файла. Если имя файла содержит пробелы, заключите *Имя файла* в кавычки. Если путь не указан, отладчик использует текущий каталог. Если не указано *имя файла*, отладчик называет файл Dbgeng.log.

**/d**
Автоматически выбирает имя файла на основе имени целевого процесса или целевого компьютера и состояния целевого объекта. Файл всегда имеет расширение имени файла .log.

## Среда

Развернуть таблицу

| Позиция | Description |
| --- | --- |
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Если при выполнении **команды logopen** уже открыт файл журнала, отладчик закрывает его. Если указать имя файла, которое уже существует, содержимое файла перезаписывается.

Команда **.logopen /t** добавляет идентификатор процесса, дату и время к имени файла журнала. В следующем примере идентификатор процесса в шестнадцатеричном формате — 0x02BC, дата — 28 февраля 2005 года, а время — 9:05:50.935.

dbgcmd

```
0:000> .logopen /t c:\logs\mylogfile.txt
Opened log file 'c:\logs\mylogfile_02BC_2005-02-28_09-05-50-935.txt'
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .netuse (управление сетевыми подключениями)

Статья • 20.12.2024

Команда **.netuse** добавляет подключение к сетевой общей папке.

```
dbgcmd
.netuse /a "[Local]" "Remote" "[User]" "[Password]"
```

## Параметры

**/a**
Добавляет новое подключение. Необходимо всегда использовать параметр **/a**.

локальный
Указывает букву диска, используемую для подключения. В кавычки необходимо заключить *local*. Если этот параметр не указан, необходимо включить пустую пару кавычки в качестве параметра.

*удаленных*
Указывает UNC-путь к подключенной общей папке. В кавычки необходимо заключить *удаленные*.

*user*
Указывает имя пользователя учетной записи, авторизованной для установления подключения. В кавычки необходимо заключить пользователя . Если этот параметр не указан, необходимо включить пустую пару кавычки в качестве параметра.

*паролей*
Указывает пароль, связанный с учетной записью пользователя . В кавычки необходимо заключить пароль. Если этот параметр не указан, необходимо включить пустую пару кавычки в качестве параметра.

## Окружающая среда

Развернуть таблицу

| Пункт | Описание |
| --- | --- |
| Режимы | Режим пользователя, режим ядра |
| Цели | Динамический, аварийный дампа |
| Платформ | Все |

## Замечания

Команда .netuse работает так, как команда net use Microsoft MS-DOS.

Если вы используете **.netuse** во время удаленного сеанса отладки, эта команда влияет на сервер отладки, а не клиент отладки.

В следующем примере показана эта команда.

dbgcmd

```
0:000> .netuse "m:" "\\myserver\myshare" "" ""
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .noshell (запретить команды оболочки)

Статья • 18.12.2024

Команда **.noshell** запрещает использование команд **.shell**.

```
dbgcmd
```

```
.noshell
```

## Окружающая среда

Развернуть таблицу

| Пункт | Описание |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Цели | Динамический, аварийный дампа |
| Платформ | Все |

## Дополнительные сведения

Дополнительные сведения о командной оболочке и других способах отключения команд оболочки см. в разделе Использование команд оболочки.

## Замечания

Если вы используете команду **.noshell**, вы не можете использовать команды **оболочки (командная оболочка**) до тех пор, пока отладчик запущен, даже если вы запускаете новый сеанс отладки.

При выполнении удаленной отладки эта команда полезна для целей безопасности.

> **Важно!**
>
> При использовании удаленной отладки необходимо учитывать важные аспекты безопасности. Дополнительные сведения о включении безопасного

режима см. в статье Вопросы безопасности для средств отладки Windows.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .noversion (отключение проверки версий)

Статья • 26.10.2023

Команда **.noversion** отключает проверку всех версий библиотек DLL расширений.

dbgcmd

```
.noversion
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

# Комментарии

Число сборок библиотек DLL расширений должно соответствовать номеру сборки отлаживаемого компьютера, так как библиотеки DLL компилируются и связаны с зависимостями от конкретных версий структур данных. Если версии не совпадают, обычно появляется следующее сообщение.

Консоль

```
*** Extension DLL(1367 Free) does not match target system(1552 Free)
```

# Ocommand (ожидайте команды из целевого объекта)

Статья • 08.07.2024

Команда **Ocommand** позволяет целевому приложению отправлять команды отладчику.

dbgcmd

```
.ocommand  String
.ocommand -d
.ocommand
```

## Параметры

*Струна*
Задает строку префикса команды. *Строка* может включать пробелы, но нельзя использовать символы управления стилем C, такие как **\"** и **\n**. Вы также можете заключить *строку* в кавычки. Однако если *строка* включает в себя точку с запятой, начальные пробелы или конечные пробелы, необходимо заключить *строку* в кавычки.

**-d**
Удаляет строку префикса команды.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения о **функциях OutputDebugString** и других функциях пользовательского режима, взаимодействующих с отладчиком, см. в документации по пакету SDK для Microsoft Windows.

## Замечания

При использовании **команды Ocommand** без параметров отладчик отображает текущую строку префикса команды. Чтобы очистить существующую строку, используйте **ocommand -d**.

При установке строки префикса команды выполняется сканирование любого целевого выходных данных (например, содержимого **команды OutputDebugString** ). Если этот результат начинается со строки префикса команды, текст выходных данных, следовать строке префикса, обрабатывается как командная строка отладчика и выполняется. При выполнении этого текста командная строка не отображается.

Целевой **объект может включать команду echo (Echo Comment)** в выходную строку, если требуется дополнительное сообщение. Целевые выходные данные, которые не начинаются со строки префикса, отображаются обычным образом.

После выполнения команд в командной строке целевой объект остается разбитым на отладчик, если только последняя команда не является **g (Go)**.

Сравнение строки префикса команды и целевого выходных данных не учитывает регистр. (Однако последующее использование **Ocommand** отображает строку, введенную с сохранением регистра.

В этом примере предположим, что в отладчике введите следующую команду.

```
dbgcmd
```

```
0:000> .ocommand magiccommand
```

Затем целевое приложение выполняет следующую строку.

```
dbgcmd
```

```
OutputDebugString("MagicCommand kb;g");
```

Отладчик распознает префикс командной строки и выполняет **кб; g** немедленно.

Однако следующая строка не приводит к выполнению каких-либо команд.

dbgcmd

```
OutputDebugString("Command on next line.\nmagiccommand kb;g");
```

В предыдущем примере нет команд, так как префикс командной строки не находится в начале выходных данных, даже если он начинает новую строку.

**Обратите внимание** , что следует выбрать префикс строки команд, который, скорее всего, не будет отображаться в любых целевых выходных данных, отличных от собственных команд.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .nvload (загрузка NatVis)

Статья • 26.10.2023

Команда .nvload загружает файл NatVis в среду отладчика. После загрузки визуализации она будет использоваться для отрисовки данных, определенных в визуализации.

```
dbgcmd
```

```
.nvload FileName|ModuleName
```

## Параметры

*FileName* | *Имя_модуля*

Указывает имя файла NatVis или имя модуля для загрузки.

**FileName** — это явное имя загружаемого NATVIS-файла. Можно использовать полный путь.

**ModuleName** — это имя модуля в отлаживаемом целевом процессе. Все файлы NatVis, внедренные в файл символов (PDB) именованного модуля, загружаются при наличии доступных.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения см. в разделе Создание настраиваемых представлений собственных объектов.

## См. также раздел

# dx (отображение выражения NatVis)

# .nvlist (список NatVis)

Статья • 05.04.2024

Команда Nvlist выводит файлы NatVis, загруженные в среду отладчика.

```
dbgcmd
```

```
.nvlist
```

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения см. в разделе "Создание настраиваемых представлений собственных объектов".

## См. также

dx (выражение Display NatVis)

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .nvunload (NatVis Unload)

Статья • 26.10.2023

Команда .nvunload выгружает файл NatVis из среды отладчика.

```
dbgcmd

.nvunload FileName|ModuleName
```

*FileName* | *Имя модуля*

Указывает имя файла NatVis или имя модуля для выгрузки.

**FileName** — это явное имя NATVIS-файла для выгрузки. Можно использовать полный путь.

**Имя модуля** — это имя модуля в отлаживаемом целевом процессе. Все файлы NatVis, внедренные в файл символов (PDB) именованного имени модуля, выгружаются.

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения см. в разделе Создание пользовательских представлений собственных объектов.

## См. также раздел

**dx (отображение выражения NatVis)**

# .nvunloadall (NatVis Unload All)

Статья • 26.10.2023

Команда .nvunloadall выгружает все файлы NatVis из среды отладчика.

```
dbgcmd
```

```
.nvunloadall
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения см. в разделе Создание настраиваемых представлений собственных объектов.

## См. также раздел

**dx (отображение выражения NatVis)**

# .ofilter (вывод целевого объекта фильтра)

Статья • 08.07.2024

**Команда ofilter** фильтрует выходные данные из целевого приложения или целевого компьютера.

dbgcmd

```
.ofilter [/!] String
.ofilter ""
.ofilter
```

## Параметры

**/!**
Изменяет фильтр таким образом, чтобы отладчик отображал только выходные данные, не содержащие *строку*. Если этот параметр не используется, отладчик отображает только выходные данные, содержащие *строку*.

*Струна*
Указывает строку для сопоставления в выходных данных целевого объекта. *Строка* может включать пробелы, но нельзя использовать символы управления стилем C, такие как **\"** и **\n**. *Строка* может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис подстановочных знаков строки".

Строки *можно заключить* в кавычки. Однако если *строка* содержит точку с запятой, начальные пробелы или конечные пробелы, необходимо использовать кавычки. Буквенно-цифровые символы в строке *преобразуются в* прописные буквы, но фактическое сопоставление шаблонов не учитывает регистр.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |

| Позиция | Description |
| --- | --- |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения о **outputDebugString** и других подпрограммах пользовательского режима см. в документации по пакету SDK для Microsoft Windows. Дополнительные сведения об **DbgPrint, DbgPrintEx** и других подпрограммах режима ядра см. в комплекте драйверов Windows (WDK).

# Замечания

Если вы используете **команду .ofilter** без параметров, отладчик отображает текущие критерии сопоставления шаблонов.

Чтобы очистить существующий фильтр, используйте **ofilter "".** Эта команда фильтрует все данные, отправленные подпрограммами пользовательского режима (например, OutputDebugString**) и подпрограммами режима ядра (например, DbgPrint**). Однако отладчик всегда отображает запросы, которые **отправляет DbgPrompt** .

**Подпрограммы DbgPrintEx и KdPrintEx** предоставляют другой метод фильтрации сообщений отладки, которые не нужны.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .open (файл с открытым исходным кодом)

Статья • 08.07.2024

Команда **open** выполняет поиск исходного пути к исходному файлу и открывает этот файл.

dbgcmd

```
.open [-m Address] FileName
.open -a Address
```

## Параметры

*Имя файла*
Указывает имя файла источника. Это имя может включать абсолютный или относительный путь. Если не указать абсолютный путь, путь интерпретируется как относительный к каталогу в исходном пути.

**-m** **** *Адрес*
Указывает адрес в исходном файле. Этот адрес должен содержаться в известном модуле. Параметр -m **** ***Address*** **следует использовать, если файл, указывающий** *Имя_файла*, не является уникальным. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

Параметр **-m** необходим, если используется исходный сервер для извлечения исходных файлов.

**-a** *Адрес*
Указывает адрес в исходном файле. Этот адрес должен содержаться в известном модуле. Если отладчик может найти исходный файл, отладчик загружает и открывает файл, а строка, соответствующая указанному адресу, выделена. Если отладчик не может найти исходный файл, адрес отображается в окне дизассембли. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

## Среда

Команду .open **можно использовать** только в WinDbg, и ее нельзя использовать в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
| --- | --- |
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения о исходных файлах и путях к источнику и других способах загрузки исходных файлов см. в разделе "Путь к источнику".

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .opendump (открытый файл дампа)

Статья • 08.07.2024

Команда **opendump открывает файл дампа** для отладки.

```
dbgcmd
```

```
.opendump DumpFile
.opendump /c "DumpFileInArchive" [CabFile]
```

## Параметры

*DumpFile*
Указывает имя открываемого файла дампа. *DumpFile* должен включать расширение имени файла (обычно .dmp или MDMP) и может включать абсолютный или относительный путь. Относительные пути относятся к каталогу, в который был запущен отладчик.

**/c** "*DumpFileInArchive*"
Указывает имя файла дампа для отладки. Этот файл дампа должен содержаться в архивном файле, который *указывает CabFile* . Файл DumpFileInArchive *необходимо заключить* в кавычки.

*CabFile*
Указывает имя открываемого архивного файла. *CabFile*должен включать расширение имени файла (обычно .cab) и может включать абсолютный или относительный путь. Относительные пути относятся к каталогу, в который был запущен отладчик. Если вы используете **параметр /c** для указания файла дампа в архиве, но не указан *CabFile*, отладчик повторно использует архивный файл, который вы недавно открыли.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | Режим пользователя, режим ядра |
| **Целевые объекты** | Только аварийный дамп (но вы можете использовать эту команду, если выполняются другие сеансы) |

| Платформы | Все |
| --- | --- |

# Замечания

После использования команды opendump **необходимо использовать команду g (Go), чтобы завершить загрузку файла дампа.**

При открытии архивного файла (например, CAB-файла) следует использовать **параметр /c** . Если этот параметр не используется и вы указываете архив для *DumpFile*, отладчик открывает первый файл с расширением MDMP или .dmp имени файла в этом архиве.

Можно использовать **opendump** , даже если сеанс отладки уже выполняется. Эта функция позволяет выполнять отладку нескольких аварийных дампов одновременно. Дополнительные сведения о том, как управлять сеансом с несколькими целевыми объектами, см. в разделе "Отладка нескольких целевых объектов".

**Обратите внимание** , что при отладке динамических целевых объектов и целевых объектов дампа возникают сложности, так как команды ведут себя по-разному для каждого типа отладки. Например, если вы используете **команду g (Go),** когда текущая система является файлом дампа, отладчик начинает выполняться, но вы не можете вернуться к отладчику, так как команда останова не распознается как допустимая для отладки файла дампа.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .outmask (маска вывода элемента управления)

Статья • 08.07.2024

Команда **outmask** управляет текущей маской вывода.

dbgcmd

```
.outmask[-] [/l] Expression
.outmask /a
.outmask /d
```

## Параметры

*Выражение*
Указывает флаги для добавления в маску. *Выражение* может быть любым значением ULONG, указывающим нужные биты флага. Список возможных флагов см. в таблице в разделе "Примечания".

**-**
Удаляет биты, которые *выражение* указывает из маски, а не добавляет их в маску.

**/l**
Сохраняет текущее значение маски вывода файла журнала. Если вы не включаете **/l**, маска вывода файла журнала совпадает с обычной маской вывода.

**/a**
Активирует все флаги маски. Этот параметр эквивалентен **0xFFFFFFFF .outmask**.

**/d**
Восстанавливает маску вывода до значения по умолчанию. Этот параметр эквивалентен **0x3F7 .outmask**.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |

| Позиция | Description |
| --- | --- |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Каждый флаг маски вывода позволяет отладчику отображать определенные выходные данные в командном окне отладчика. Если заданы все флаги маски, отображаются все выходные данные.

Вы должны удалить флаги маски вывода с осторожностью, так как вы не сможете считывать выходные данные отладчика.

Существуют следующие значения флагов.

Развернуть таблицу

| Значение | Значение по умолчанию | Description |
| --- | --- | --- |
| 1 | Вкл | Обычные выходные данные |
| 2 | Вкл | Вывод ошибок |
| 4 | Вкл | Предупреждения |
| 8 | Выключено | Дополнительные выходные данные |
| 0x10 | Вкл | Вывод запроса |
| 0x20 | Вкл | Регистрация дампа перед запросом |
| 0x40 | Вкл | Предупреждения, относящиеся к операции расширения |
| 0x80 | Вкл | Отладка выходных данных целевого объекта (например, **OutputDebugString** или **DbgPrint**) |
| 0x100 | Вкл | Отладочные входные данные, ожидаемые |

| Значение | Значение по умолчанию | Description |
| --- | --- | --- |
| | | целевым объектом (например, **DbgPrompt**) |
| 0x200 | Вкл | Сообщения символов (например, **!sym шумный**) |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .pagein (страница в памяти)

Статья • 08.07.2024

Страницы **команд .pagein** в указанном регионе памяти.

```
dbgcmd
.pagein [Options] Address
```

## Параметры

*Параметры*

Любой из следующих параметров:

**/p** *Process*

Указывает адрес процесса, который владеет памятью, в которую требуется выполнить страницу. (Точнее, этот параметр задает адрес блока EPROCESS для процесса.) Если вы опустите *процесс* или укажите ноль, отладчик использует текущий параметр процесса. Дополнительные сведения о параметре процесса см. в разделе .process (Set Process Context)

**/f**
Принудительно выводит память на страницу, даже если адрес находится в памяти ядра, а версия Windows не поддерживает это действие.

*Адрес*

Указывает адрес страницы.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра (но не во время локальной отладки ядра) |
| Целевые объекты | Только динамическая отладка |
| Платформы | Все |

## Замечания

После запуска **команды .pagein** необходимо использовать **команду g (Go)** для возобновления выполнения программы. Через некоторое время целевой компьютер автоматически переходит в отладчик.

На этом этапе указанный адрес будет отображаться на странице. Если вы используете **параметр /p** , контекст процесса также устанавливается в указанный процесс, точно так же, как если бы вы использовали **команду .process /i Process** .

Если адрес уже указан на странице, **команда .pagein** по-прежнему проверяет, включено ли адрес, а затем возвращается в отладчик. Если адрес недопустим, эта команда возвращается только в отладчик.

## Требования

Поддерживается в Windows XP и более поздних версиях Windows.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .pcmd (установка команды командной строки)

Статья • 05.04.2024

Команда **PCMD** заставляет отладчика выдавать команду всякий раз, когда целевой объект перестает выполнять и отображает запрос в окне команды отладчика с регистрацией или целевыми сведениями о состоянии.

dbgcmd

```
.pcmd -s CommandString
.pcmd -c
.pcmd
```

## Параметры

**-s** **** *CommandString*
Задает новую строку командной строки командной строки. При остановке выполнения целевого объекта отладчик выдает проблемы и немедленно запускает *команду CommandString* . Если *CommandString* содержит пробелы или точки с запятой, необходимо заключить его в кавычки.

**-c**
Удаляет любую существующую строку командной строки командной строки.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о командной строке отладчика см. в разделе "Использование команд отладчика".

# Замечания

При использовании **команды PCMD** без параметров отображается текущая команда командной строки.

При установке команды командной строки с помощью **pcmd-s** указанный *CommandString* выдается всякий раз, когда целевой объект перестает выполнять (например, когда **заканчивается команда g**, **p** или **t).** Команда *CommandString* не выдается при использовании команды без выполнения, если эта команда не отображает регистры или сведения о целевом состоянии.

В следующем примере первое использование **PCMD** задает фиксированную строку, которая отображается с запросом. Во-вторых, использование **PCMD** приводит к отображению отладчика текущего идентификатора процесса и идентификатора потока целевого объекта при каждом появлении запроса. Специальная строка не отображается после **использования команды Ttime** , так как эта команда не включает выполнение.

dbgcmd

```
0:000> .pcmd
No per-prompt command

0:000> .pcmd -s ".echo Execution is done."
Per-prompt command is '.echo Execution is done.'

0:000> t
Prymes!isPrime+0xd0:
004016c0 837dc400      cmp dword ptr [ebp-0x3c],0x0
ss:0023:0012fe70=00000002
Execution is done.

0:000> t
Prymes!isPrime+0xd4:
004016c4 7507            jnz     Prymes!isPrime+0xdd (004016cd)
 [br=1]
Execution is done.

0:000> .ttime
Created: Thu Aug 21 13:18:59 2003
Kernel:  0 days 0:00:00.031
User:    0 days 0:00:00.000

0:000> .pcmd -s "r $tpid, $tid"
Per-prompt command is 'r $tpid, $tid'
```

```
0:000> t
Prymes!isPrime+0xdd:
004016cd ebc0              jmp     Prymes!isPrime+0x9f (0040168f)
$tpid=0000080c $tid=00000514

0:000> t
Prymes!isPrime+0x9f:
0040168f 8b55fc            mov     edx,[ebp-0x4]
ss:0023:0012fea8=00000005
$tpid=0000080c $tid=00000514
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .pop (состояние отладчика восстановления)

Статья • 08.07.2024

**Команда** POP восстанавливает состояние отладчика в состояние, которое ранее было сохранено с помощью **команды PUSH (Сохранить состояние отладчика).**

dbgcmd

```
.pop
.pop /r
.pop /r /q
```

## Параметры

**/r**
Указывает, что сохраненные значения псевдорегистров $t 0 до $t 19 должны быть восстановлены. Если **параметр /r** не включен, эти значения не влияют на **команду POP** .

**/q**
Указывает, что команда выполняется тихо. То есть команда выполняется без отображения выходных данных.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Эта команда наиболее полезна при использовании с скриптами и программами команд отладчика, чтобы они могли работать с одним фиксированным состоянием.

Если команда выполнена успешно, выходные данные не отображаются.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .prefer_dml (предпочитать язык разметки отладчика)

Статья • 26.10.2023

Команда **.prefer_dml** задает поведение по умолчанию для команд, которые могут предоставлять выходные данные в формате языка разметки отладчика (DML).

dbgcmd

```
.prefer_dml 0
.prefer_dml 1
```

## Параметры

0
По умолчанию все команды предоставляют выходные данные в виде обычного текста.

1
По умолчанию команды, способные предоставлять выходные данные DML, предоставляют выходные данные DML.

## См. также

Команды языка разметки отладчика

# .process (задать контекст процесса)

Статья • 05.04.2024

Команда **.process** указывает, какой процесс используется для контекста процесса.

```
dbgcmd
```

```
.process [/i] [/p [/r]] [/P] [Process]
```

## Параметры

**/i**
Только динамическая отладка; не во время локальной отладки ядра) Указывает, что *процесс* должен быть отлаживаем *инвазивно*. Такой вид отладки означает, что операционная система целевого компьютера фактически делает указанный процесс активным. (Без этого параметра команда .process **изменяет выходные данные отладчика,** но не влияет на сам целевой компьютер.) При использовании /i **необходимо использовать команду g (Go) для выполнения целевого объекта.** Через несколько секунд целевой объект прерывается обратно в отладчик, а указанный *процесс* активен и используется для контекста процесса.

**/p**
Преобразует все записи таблицы страниц перехода (PTEs) для этого процесса на физические адреса перед доступом, если вы используете **/p** и *Process* является ненулевой. Этот перевод может привести к замедлению, так как отладчик должен найти физические адреса для всей памяти, используемой этим процессом. Кроме того, отладчику может потребоваться передать значительный объем данных по отладочному кабелю. (Это поведение совпадает с **. cache forcedecodeuser**.)

Если включен **параметр /p** и *процесс* равен нулю или не указан, перевод отключен. (Это поведение совпадает с **. cache noforcedecodeptes**.)

**/R**
Перезагружает символы пользовательского режима после установки контекста процесса, если вы используете **параметры /r** и **/p** . (Это поведение совпадает с **. перезагрузить /user**.)

**/P**
(Только динамические отладочные и полные дампы памяти) Преобразует все записи таблицы страниц перехода (PTEs) на физические адреса перед доступом, если вы используете **/P** и *Process* является ненулевой. **В отличие от параметра /p**,

**параметр /P** преобразует PTEs для всех процессов пользовательского режима и режима ядра, а не только для указанного процесса. Этот перевод может привести к замедлению, так как отладчик должен найти физические адреса для всей используемой памяти. Кроме того, отладчику может потребоваться передать большое количество данных по отладочному кабелю. (Это поведение совпадает с **.cache forcedecodeptes**.)

*Обработать*
Указывает адрес нужного процесса. (Точнее, этот параметр задает адрес блока EPROCESS для этого процесса. Контекст процесса задан для этого процесса. Если опустить *процесс* или указать ноль, контекст процесса сбрасывается в процесс по умолчанию для текущего состояния системы. (Если вы использовали параметр /i **для задания контекста процесса, необходимо использовать параметр /i** для сброса контекста процесса.)

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о контексте процесса и других параметрах контекста см. в разделе "Изменение контекстов".

# Замечания

Как правило, при отладке ядра единственным видимым адресным пространством в режиме пользователя является тот, который связан с текущим процессом.

Команда **.process** указывает отладчику ядра использовать определенный процесс пользовательского режима в качестве контекста процесса. Это использование имеет несколько эффектов, но самое важное заключается в том, что отладчик имеет доступ к виртуальному адресном пространству этого процесса. Отладчик

использует таблицы страниц для этого процесса для интерпретации всех адресов памяти в пользовательском режиме, чтобы можно было читать и записывать в эту память.

Команда **.context (Set User-Mode Address Context)** имеет аналогичный эффект. **Однако команда .context** задает контекст *адреса в пользовательском режиме для определенного каталога страницы, а* ***команда .process*** *задает* контекст процесса определенному процессу. На процессоре на **основе x86 контекст** и **процесс .имеют** почти тот же эффект. Однако на процессоре на основе Itanium один процесс может иметь несколько каталогов страниц. В этой ситуации команда .process **является более мощной**, так как она обеспечивает доступ ко всем каталогам страниц, связанным с процессом. Дополнительные сведения о контексте процесса см. в разделе "Контекст процесса".

**Обратите внимание** , что при выполнении динамической отладки следует использовать **параметр /i** или **/p** . Без одного из этих параметров невозможно правильно отображать память пользователя или сеанса.

Параметр **/i** активирует целевой процесс. При использовании этого параметра необходимо выполнить целевой объект один раз, чтобы эта команда ввела в силу. При повторном выполнении контекст процесса теряется.

Параметр **/p** включает параметр принудительного **кода.** (Если параметр принудительного **кодauser** уже активен, не нужно использовать **/p**.) Контекст процесса и состояние принудительного **кодauser** остаются только до тех пор, пока целевой объект не будет выполнен снова.

Если выполняется отладка аварийного дампа, **параметры /i** и **/p** недоступны. Однако вы не можете получить доступ к любой части виртуального адресного пространства процесса в пользовательском режиме, которые были страницы на диск при сбое.

Если вы хотите использовать отладчик ядра для задания точек останова в пользовательском пространстве, используйте **параметр /i** , чтобы переключить целевой объект в правильный контекст процесса.

В следующем примере показано, как использовать **расширение !process** для поиска адреса блока EPROCESS для требуемого процесса.

dbgcmd

```
kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS fe5039e0  SessionId: 0  Cid: 0008    Peb: 00000000  ParentCid: 0000
    DirBase: 00030000  ObjectTable: fe529b68  TableSize:  50.
```

```
    Image: System

.....

PROCESS fe3c0d60  SessionId: 0  Cid: 0208    Peb: 7ffdf000  ParentCid: 00d4
    DirBase: 0011f000  ObjectTable: fe3d0f48  TableSize:  30.
    Image: regsvc.exe
```

Теперь в примере используется команда .process **с этим адресом** процесса.

dbgcmd

```
kd> .process fe3c0d60
Implicit process is now fe3c0d60
```

Обратите внимание, что эта команда делает команду контекста ненужной. Контекст адреса в пользовательском режиме уже имеет требуемое значение.

dbgcmd

```
kd> .context
User-mode page directory base is 11f000
```

Это значение позволяет анализировать адресное пространство различными способами. Например, в следующем примере показаны выходные данные **расширения !peb** .

dbgcmd

```
kd> !peb
PEB at 7FFDF000
    InheritedAddressSpace:    No
    ReadImageFileExecOptions: No
    BeingDebugged:            No
    ImageBaseAddress:         01000000
    Ldr.Initialized: Yes
    Ldr.InInitializationOrderModuleList: 71f40 . 77f68
    Ldr.InLoadOrderModuleList: 71ec0 . 77f58
    Ldr.InMemoryOrderModuleList: 71ec8 . 77f60
        01000000 C:\WINNT\system32\regsvc.exe
        77F80000 C:\WINNT\System32\ntdll.dll
        77DB0000 C:\WINNT\system32\ADVAPI32.dll
        77E80000 C:\WINNT\system32\KERNEL32.DLL
        77D40000 C:\WINNT\system32\RPCRT4.DLL
        77BE0000 C:\WINNT\system32\secur32.dll
    SubSystemData:     0
    ProcessHeap:       70000
    ProcessParameters: 20000
        WindowTitle: "'C:\WINNT\system32\regsvc.exe'"
```

```
        ImageFile:    'C:\WINNT\system32\regsvc.exe'
        CommandLine:  'C:\WINNT\system32\regsvc.exe'
        DllPath:
'C:\WINNT\system32;.;C:\WINNT\System32;C:\WINNT\system;C:\WINNT;C:\WINNT\sys
tem32;C:\WINNT;C:\WINNT\System32\Wbem;C:\PROGRA~1\COMMON~1\AUTODE~1'
        Environment:  0x10000
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .prompt_allow (отображение запроса элемента управления)

Статья • 08.07.2024

Команда **.prompt_allow** управляет информацией, отображаемой во время пошагового выполнения и трассировки, и всякий раз, когда выполнение целевого объекта останавливается.

dbgcmd

```
.prompt_allow {+|-}Item [...]
.prompt_allow
```

## Параметры

**+**
Отображает указанный элемент на шаге, трассировке и запросе выполнения. Необходимо добавить пробел перед знаком плюса (+), но после него нельзя добавить пробел.

**-**
Запрещает отображение указанного элемента на шаге, трассировке и запросе на выполнение. Необходимо добавить пробел перед знаком минуса (-), но после него нельзя добавить пробел.

*Пункт*
Указывает элемент для отображения или не отображается. Можно указать любое количество элементов. Разделение нескольких элементов по пробелам. Перед каждым элементом необходимо добавить знак плюса (+) или знак минуса (-). Можно использовать следующие элементы:

**дис**
Распакуленные инструкции в текущем расположении.

**ea**
Действующий адрес текущей инструкции.

**reg**
Текущее состояние наиболее важных регистров. Вы можете отключить отображение регистрации с помощью **команды pr**, **tr** или .prompt_allow -reg. Все три из этих команд управляют одинаковым параметром, и вы можете использовать

любой из них для переопределения любого предыдущего использования этих команд.

Вы также можете отключить отображение регистрации с помощью команды l-os. Этот параметр отличается от других трех команд, как описано в следующем разделе примечаний. Чтобы управлять отображением регистров и флагов, используйте **команду rm (Register Mask).**

**src**
Исходная строка, соответствующая текущей инструкции. Вы можете отключить отображение исходной строки с помощью l-ls или .prompt_allow -src; Команды. Необходимо включить отображение исходной строки с помощью обоих механизмов для отображения.

**sym**
Символ текущей инструкции. Этот символ включает текущий модуль, имя функции и смещение.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения о командах, влияющих на выполнение, см. в разделе "Управление целевым объектом".

# Замечания

Команду .prompt_allow **можно использовать** без параметров, чтобы показать, какие элементы отображаются и не отображаются. При каждом запуске **.prompt_allow** отладчик изменяет состояние только указанных элементов.

По умолчанию отображаются все элементы.

Если вы использовали параметр l+os, этот параметр переопределяет любой из **.prompt_allow** параметров, отличных от **src**.

Можно также использовать сложную команду, например следующий пример.

dbgcmd

```
0:000> .prompt_allow -reg -dis +ea
Allow the following information to be displayed at the prompt:
(Other settings can affect whether the information is actually displayed)
   sym - Symbol for current instruction
    ea - Effective address for current instruction
   src - Source info for current instruction
Do not allow the following information to be displayed at the prompt:
   dis - Disassembly of current instruction
   reg - Register state
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .push (сохранение состояния отладчика)

Статья • 08.07.2024

Команда **PUSH** сохраняет текущее состояние отладчика.

dbgcmd

```
.push
.push /r
.push /r /q
```

## Параметры

**/r**
Указывает, что текущие значения в псевдорегистрирующих регистрах **$t 0** в **$t 19** должны быть сохранены. **Если параметр /r** не используется, эти значения не сохраняются командой **PUSH**.

**/q**
Указывает, что команда выполняется тихо. То есть команда выполняется без отображения выходных данных.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Эта команда наиболее полезна при использовании с скриптами и программами команд отладчика, чтобы они могли работать с одним фиксированным состоянием. Чтобы восстановить отладчик в состоянии, которое ранее было сохранено с

помощью этой команды, используйте **команду POP (Восстановление состояния отладчика).** Если команда выполнена успешно, выходные данные не отображаются.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .quit_lock (предотвращение случайного выхода)

Статья • 20.12.2024

Команда **.quit_lock** задает пароль, чтобы предотвратить случайное завершение сеанса отладки.

dbgcmd

```
.quit_lock /s NewPassword
.quit_lock /q Password
.quit_lock
```

## Параметры

**/s** **** *NewPassword*
Запрещает завершение сеанса отладки и сохранение *NewPassword*. Сеанс отладчика нельзя завершить, пока не будет использоваться команда **.quit_lock /q** вместе с этим паролем. *NewPassword* может быть любой строкой. Если он содержит пробелы, необходимо заключить *NewPassword* в кавычки.

**/q** **** *пароль*
Позволяет завершить сеанс отладки. *пароль* должен соответствовать паролю, заданному командой **.quit_lock /s**.

## Окружающая среда

Развернуть таблицу

| Пункт | Описание |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Цели | Динамический, аварийный дампа |
| Платформ | Все |

## Замечания

Без параметров **.quit_lock** отображает текущее состояние блокировки, включая полный текст пароля.

Чтобы изменить существующий пароль, можно повторить команду **.quit_lock /s**.

При использовании **.quit_lock /q**блокировка удаляется. Эта команда не закрывает отладчик. Вместо этого команда позволяет выйти из сеанса только в обычном режиме, когда требуется.

**Примечание** Пароль не является секретом. Любой удаленный пользователь, подключенный к сеансу отладки, может использовать **.quit_lock** для определения пароля. Эта команда предназначена для предотвращения случайного использования команды **q (Quit)**. Эта команда особенно полезна, если перезагрузка сеанса отладки может оказаться сложной (например, во время удаленной отладки).

Невозможно использовать команду **.quit_lock /s** в защищенном режиме. Если вы используете эту команду перед активацией безопасного режима, защита паролей остается, но изменить или удалить пароль нельзя.

> ⓘ **Важно!**
>
> При использовании удаленной отладки необходимо учитывать важные аспекты безопасности. Дополнительные сведения о включении безопасного режима см. в статье **Вопросы безопасности для средств отладки Windows**.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Readmem (чтение памяти из файла)

Статья • 08.07.2024

Команда **readmem** считывает необработанные двоичные данные из указанного файла и копирует данные в память целевого компьютера.

```
dbgcmd
```

```
.readmem FileName Range
```

## Параметры

*Имя файла*
Указывает имя файла для чтения. Можно указать полный путь или только имя файла. Если имя файла содержит пробелы, заключите *Имя файла* в кавычки. Если путь не указан, отладчик использует текущий каталог.

*Диапазон*
Задает диапазон адресов для размещения данных в памяти. Этот параметр может содержать начальный и конечный адрес или начальный адрес и число объектов. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Данные памяти копируются буквально на целевой компьютер. Отладчик не анализирует данные каким-либо образом. Например, **команда readmem myfile**

**1000 10** копирует 10 байт из файла Myfile и сохраняет их в памяти целевого компьютера, начиная с адреса 1000.

Команда **readmem** является противоположной **команде .writemem (запись памяти в файл).**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .reboot (Reboot Target Computer)

Статья • 26.10.2023

Команда **.reboot** перезапускает целевой компьютер.

```
dbgcmd
```

```
.reboot
```

## Среда

| | |
|---|---|
| **Режимы** | Только в режиме ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Все |

## Дополнительные сведения

Дополнительные сведения о связанных командах и объяснение того, как процесс перезагрузки влияет на отладчик, см. в разделе Сбой и перезагрузка целевого компьютера.

# .record_branches (включение записи ветвей)

Статья • 26.10.2023

Команда **.record_branches** включает запись ветвей, выполняемых кодом целевого объекта.

dbgcmd

```
.record_branches {1|0}
.record_branches
```

## Среда

| | |
|---|---|
| **Режимы** | Пользовательский режим, режим ядра |
| **Целевые объекты** | Только динамическая отладка |
| **Платформы** | Только для 64-разрядных процессоров |

# Комментарии

Команда **.record_branches 1** позволяет записывать ветви в коде целевого объекта. Команда **.record_branches 0** отключает эту запись.

Без параметров **.record_branches** отображает текущее состояние этого параметра.

# .reload (модуль перезагрузки)

Статья • 08.07.2024

Команда **reload** удаляет все сведения о символах для указанного модуля и перезагрузит эти символы по мере необходимости. В некоторых случаях эта команда также перезагрузит или выгрузит сам модуль.

dbgcmd

```
.reload [Options] [Module[=Address[,Size[,Timestamp]]]]
.reload -?
```

## Параметры

*Параметры* любого из следующих параметров:

**/d**
Перезагрузите все модули в списке модулей отладчика. (Если вы опустите все параметры, эта ситуация используется по умолчанию во время отладки в пользовательском режиме.)

**/f**
Принудительно отладчик немедленно загружает символы. Этот параметр переопределяет *отложенную загрузку* символов. Дополнительные сведения см. в разделе "Примечания".

**/i**
Игнорирует несоответствие в версиях PDB-файла. (Если этот параметр не включен, отладчик не загружает несогласованные файлы символов.) При использовании **/i также используется /f**, даже если вы не указываете его явным образом.

**/l**
Выводит список модулей, но не перезагрузит их символы. (В режиме ядра этот параметр предоставляет выходные данные, аналогичные выходным данным. Команда lm .)

**/n**
Перезагрузит только символы ядра. Этот параметр не перезагрузит символы пользователя. (Этот параметр можно использовать только во время отладки в режиме ядра.)

**/o**
Принудительно перезаписывает кэшированные файлы в нижнем хранилище сервера символов. При использовании этого флага также следует включить **/f**. По умолчанию подчиненные файлы хранилища никогда не перезаписываются.

Так как сервер символов использует отдельные имена файлов для символов каждой разной сборки двоичного файла, вам не нужно использовать этот параметр, если вы не считаете, что нижнее хранилище повреждено.

**/s**
Перезагрузит все модули в списке образов модуля системы. (Если вы опустите все параметры, эта ситуация используется по умолчанию во время отладки в режиме ядра.)

Если вы загружаете отдельный системный модуль по имени при отладке в пользовательском режиме, необходимо включить **/s**.

**/u**
Выгружает указанный модуль и все его символы. Отладчик выгрузит любой загруженный модуль, имя которого соответствует *module*, независимо от полного пути. Имена изображений также выполняются в поиске. Дополнительные сведения см. в заметке в следующем разделе "Примечания".

**/unl**
Перезагрузит символы на основе сведений о изображении в выгруженном списке модулей.

**/пользователь**
Перезагружает только символы пользователя. (Этот параметр можно использовать только во время отладки в режиме ядра.)

**/v**
Включает подробный режим.

**/w**
Обрабатывает *модуль* как литеральную строку. Это лечение запрещает отладчику расширять подстановочные знаки.

*Модуль*
Указывает имя изображения в целевой системе, для которой нужно перезагрузить символы на хост-компьютере. *Модуль* должен содержать расширение имени и имени файла. Если вы не используете **параметр /w** , *модуль* может содержать различные подстановочные знаки и описатели. Дополнительные сведения о синтаксисе см. в разделе "Синтаксис подстановочных знаков строки". Если вы

опустите *модуль, поведение* ***команды reload*** *зависит от того, какие параметры* вы используете.

*Адрес*
Указывает базовый адрес модуля. Как правило, этот адрес должен быть указан только в том случае, если заголовок изображения поврежден или выстраивается.

*Размер*
Указывает размер образа модуля. Во многих ситуациях отладчик знает правильный размер модуля. Если отладчик не знает правильный размер, следует указать *размер*. Этот размер может быть фактическим размером модуля или большим числом, но размер не должен быть меньше. Как правило, этот размер необходимо иметь только в том случае, если заголовок изображения поврежден или выстраивается.

*Метка времени*
Задает метку времени образа модуля. Во многих ситуациях отладчик знает правильную метку времени модуля. Если отладчик не знает метки времени, следует указать *метку времени*. Как правило, необходимо иметь эту метку времени, только если заголовок изображения поврежден или выстраивается.

**Обратите внимание,** что между параметрами *адреса*, *размера* и *метки* времени не должно быть пустого места.

**-?**
Отображает короткий текст справки для этой команды.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Дополнительные сведения об отложенной (отложенной) загрузке символов см. в разделе "Отложенная загрузка символов". Дополнительные сведения о других

параметрах символов см. в разделе "Параметры символов".

# Замечания

Команда **reload** не приводит к чтению сведений о символах. Вместо этого эта команда позволяет отладчику знать, что файлы символов, возможно, изменились или что новый модуль должен быть добавлен в список модулей. Эта команда приводит к тому, что отладчик пересматривает список модулей и удаляет сведения о символах для указанных модулей. Фактические сведения о символах не считываются из отдельных PDB-файлов до тех пор, пока не потребуется информация. (Этот тип загрузки называется *отложенной загрузкой* символов или *отложенной загрузкой* символов.)

Вы можете принудительно загружать символы с помощью **параметра /f** или выполнив **команду ld (Load Symbols).**

Команда **reload** полезна, если система перестает отвечать (т. е. сбои), что может привести к потере символов для отлаживаемого целевого компьютера. Команда также может быть полезной, если вы обновили дерево символов.

Если заголовок изображения неправильно по какой-либо причине, например выгрузочный модуль или выгружается, вы можете правильно загрузить символы с помощью **аргумента /unl** или указать *как адрес* , так и *размер*.

Команда **.reload /u** выполняет широкий поиск. Отладчик сначала пытается сопоставить *модуль* с точным именем модуля независимо от пути. Если отладчик не может найти это совпадение, *модуль* обрабатывается как имя загруженного образа. Например, если hal, который находится в памяти, имеет имя модуля halacpi.dll, обе из следующих команд выгрузают свои символы.

dbgcmd

```
kd> .reload /u halacpi.dll

kd> .reload /u hal
```

Если выполняется отладка в пользовательском режиме и требуется загрузить модуль, который не входит в список модулей целевого приложения, необходимо включить **параметр /s** , как показано в следующем примере.

dbgcmd

```
0:000> .reload /u ntdll.dll
Unloaded ntdll.dll
```

```
0:000> .reload /s /f ntdll.dll
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .remote (создание сервера Remote.exe)

Статья • 20.12.2024

Команда **.remote** запускает Remote.exe server, что позволяет удаленно подключиться к текущему сеансу отладки.

```
dbgcmd

.remote session
```

## Параметры

сеанса
Указывает имя, которое вы предоставляете сеансу отладки.

## Окружающая среда

Вы можете использовать команду **.remote** в KD и CDB, но ее нельзя использовать в WinDbg.

Развернуть таблицу

| Пункт | Описание |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Цели | Динамический, аварийный дампа |
| Платформ | Все |

## Дополнительные сведения

Дополнительные сведения об использовании серверов Remote.exe и клиентов Remote.exe см. в Remote.exeудаленной отладки.

> **Важно!**

При использовании удаленной отладки необходимо учитывать важные аспекты безопасности. Дополнительные сведения о включении безопасного режима см. в статье **Вопросы безопасности для средств отладки Windows**.

## Замечания

Команда **.remote** создает процесс Remote.exe и преобразует текущий отладчик в сервер Remote.exe. Этот сервер позволяет клиенту Remote.exe подключаться к текущему сеансу отладки.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .remote_exit (клиент отладки выхода)

Статья • 19.12.2024

Команда **.remote_exit** выходит из клиента отладки, но не завершает сеанс отладки.

```
dbgcmd
```

```
.remote_exit [FinalCommands]
```

## Параметры

*FinalCommands*
Указывает командную строку для передачи на сервер отладки. Следует разделить несколько команд с запятой. Эти команды передаются на сервер отладки, а соединение затем нарушается.

## Окружающая среда

Команду **.remote_exit** можно использовать только в файле скрипта. Его можно использовать в KD и CDB, но его нельзя использовать в WinDbg.

Развернуть таблицу

| Пункт | Описание |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Цели | Динамический, аварийный дампа |
| Платформ | Все |

## Дополнительные сведения

Дополнительные сведения о файлах скриптов см. в разделе Использование файлов скриптов. Дополнительные сведения об отладке клиентов и серверах отладки см. в удаленной отладки с помощьюотладчика.

**Важно!**

При использовании удаленной отладки необходимо учитывать важные аспекты безопасности. Дополнительные сведения о включении безопасного режима см. в статье **Вопросы безопасности для средств отладки Windows**.

## Замечания

Если вы используете KD или CDB напрямую, а не с помощью сценария, вы можете выйти из клиента отладки с помощью клавиши CTRL+B .

Вы не можете выйти из клиента отладки с помощью скрипта, выполняемого в WinDbg.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .restart (Restart Target Application)

Статья • 26.10.2023

Команда **.restart** перезапускает целевое приложение.

Не путайте эту команду с командой **.restart (Restart Kernel Connection),** которая работает только в режиме ядра.

dbgcmd

```
.restart
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Только пользовательский режим |
| Цели | Только динамическая отладка |
| Платформы | все |

## Дополнительные сведения

Дополнительные сведения о том, как выполнить эту команду, и общие сведения о связанных командах см. в разделе Управление целевым объектом.

# Комментарии

CDB и WinDbg могут перезапустить целевое приложение, если отладчик изначально создал приложение. Вы можете использовать команду **.restart** , даже если целевое приложение уже закрыто.

Однако если приложение запущено, а отладчик позже подключен к процессу, команда **.restart** не будет действовать.

После перезапуска процесс немедленно переходит в отладчик.

В WinDbg используйте **представление | Команда командной строки WinDbg** , если вы запустили целевой объект из командной строки WinDbg и хотите просмотреть эту командную строку, прежде чем решать, следует ли использовать **.restart**.

# .restart (перезапуск ядра Подключение ion)

Статья • 05.04.2024

Команда **.restart перезагрузит** подключение ядра.

Не путайте эту команду с командой перезапуска **(перезапуск целевого приложения),** которая работает только в пользовательском режиме.

dbgcmd

```
.restart
```

## Среда

Команду .restart **можно использовать** только в KD.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения о повторном подключении к целевому объекту см. в разделе Синхронизация с целевым компьютером.

## Замечания

Команда **перезапуска** аналогична **команде CTRL+R (повторной синхронизации),** за исключением того, что **перезагрузка** еще более обширна в его эффекте. Эта команда эквивалентна окончанию отладчика, а затем присоединению нового отладчика к целевому компьютеру.

Команда **перезапуска** наиболее полезна при выполнении удаленной отладки с помощью remote.exe и завершения и перезапуска отладчика может оказаться сложной. Однако при выполнении удаленной отладки через отладчик нельзя использовать **перезапуск** из клиента отладки.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .rrestart (регистрация для перезапуска)

Статья • 26.10.2023

Команда **.rrestart** регистрирует сеанс отладки для перезапуска в случае перезагрузки или сбоя приложения.

dbgcmd

```
.rrestart
```

## Комментарии

Эта команда не работает для сеансов отладчика с повышенными привилегиями.

## См. также

.urestart

# .scroll_prefs (параметры прокрутки источника управления)

Статья • 08.07.2024

Команда **.scroll_prefs** управляет расположением источника в окне источника при прокрутке до строки.

dbgcmd

```
.scroll_prefs Before After
.scroll_prefs
```

## Параметры

*Перед*
Указывает, сколько исходных строк перед прокручиваемой линией должно отображаться в окне "Источник".

*После*
Указывает, сколько исходных строк после прокручиваемой строки должно отображаться в окне "Источник".

## Среда

Эта команда доступна только в WinDbg и не может использоваться в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Если эта команда используется без параметров, отображаются текущие параметры прокрутки источника.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .scriptdebug (отладка JavaScript)

Статья • 05.04.2024

**Используйте команду .scriptdebug** для отладки скриптов JavaScript.

```
dbgcmd
```

```
.scriptdebug FileName
```

## Параметры

*FileName*

Указывает имя скрипта JavaScript отладчика для отладки.

## Среда

Развернуть таблицу

| | |
|---|---|
| **Режимы** | all |
| **Целевые объекты** | all |
| **Платформы** | all |

## Дополнительная информация

Общие сведения об отладке JavaScript см. в статье "Скрипты отладчика JavaScript" — отладка JavaScript.

> **Примечание**
>
> Чтобы использовать отладку JavaScript с WinDbg, запустите отладчик как Администратор istrator.

## Замечания

Перед отладкой JavaScript выполните следующие действия.

1. Загрузите пример скрипта.

```
dbgcmd

0:000> .scriptload C:\MyScripts\DebuggableSample.js
```

Чтобы начать активную отладку скрипта, используйте **команду scriptdebug** .

```
dbgcmd

0:000> .scriptdebug C:\MyScripts\DebuggableSample.js
>>> ****** DEBUGGER ENTRY DebuggableSample ******
           No active debug event!

>>> Debug [DebuggableSample <No Position>] >
```

Когда появится запрос `>>> Debug [DebuggableSample <No Position>] >` и запрос на ввод, вы находитесь в отладчике скрипта.

**Используйте команду .help** или **?** для отображения списка команд в среде отладки JavaScript.

```
dbgcmd

>>> Debug [DebuggableSample <No Position>] >.help
Script Debugger Commands (*NOTE* IDs are **PER SCRIPT**):
    ? ................................ Get help
    ? <expr>  ........................ Evaluate expression <expr> and
display result
    ?? <expr>  ....................... Evaluate expression <expr> and
display result
    |  ................................ List available scripts
    |<scriptid>s  ..................... Switch context to the given script
    bc \<bpid\>  ....................... Clear breakpoint by specified \
<bpid\>
    bd \<bpid\>  ....................... Disable breakpoint by specified \
<bpid\>
    be \<bpid\>  ....................... Enable breakpoint by specified \
<bpid\>
    bl  ............................... List breakpoints
    bp <line>:<column>  ............... Set breakpoint at the specified
line and column
    bp <function-name>  ............... Set breakpoint at the (global)
function specified by the given name
    bpc  .............................. Set breakpoint at current location
    dv  ............................... Display local variables of current
frame
    g  ................................ Continue script
    gu   .............................. Step out
    k  ................................ Get stack trace
```

```
    p  ............................... Step over
    q  ............................... Exit script debugger (resume
execution)
    sx  .............................. Display available events/exceptions
to break on
    sxe <event>  ..................... Enable break on <event>
    sxd <event>  ..................... Disable break on <event>
    t  ............................... Step in
    .attach <scriptId>  ............... Attach debugger to the script
specified by <scriptId>
    .detach [<scriptId>]  ............. Detach debugger from the script
specified by <scriptId>
    .frame <index>  ................... Switch to frame number <index>
    .f+  .............................. Switch to next stack frame
    .f-  .............................. Switch to previous stack frame
    .help  ............................ Get help
```

## События

**Используйте команду отладчика скрипта sx**, чтобы просмотреть список событий, которые могут быть захвачены.

dbgcmd

```
>>> Debug [DebuggableSample <No Position>] >sx
sx
    ab  [   inactive] .... Break on script abort
    eh  [   inactive] .... Break on any thrown exception
    en  [   inactive] .... Break on entry to the script
    uh  [     active] .... Break on unhandled exception
```

**Используйте команду отладчика скрипта sxe**, чтобы включить любое поведение останова. Например, чтобы включить разрыв для записи, чтобы скрипт заловился в отладчик скрипта, как только любой код в нем выполняется, используйте эту команду.

dbgcmd

```
>>> Debug [DebuggableSample <No Position>] >sxe en
sxe en
Event filter 'en' is now active
```

**Используйте команду отладчика скрипта sxd**, чтобы отключить любое поведение точки останова.

dbgcmd

```
>>> Debug [DebuggableSample 34:5] >sxd en
sxd en
Event filter 'en' is now inactive
```

## Трассировка стека

**Используйте команду k** для отображения трассировки стека.

dbgcmd

```
>>> Debug [DebuggableSample 34:5] >k
k
    ##  Function                          Pos    Source Snippet
-> [00] throwAndCatch                     034:05 (var curProc =
host.currentProcess)
   [01] outer                             066:05 (var foo = throwAndCatch())
   [02] outermost                         074:05 (var result = outer())
```

## Перечисление переменных

Используйте **??** для перечисления значений переменных JavaScript.

dbgcmd

```
>>> Debug [DebuggableSample 34:5] >??someObj
??someObj
someObj          : {...}
    __proto__        : {...}
    a                : 0x63
    b                : {...}
```

## Точки останова

Используйте следующие команды точки останова для работы с дополнительными точками останова.

**bpid<>**: Установка точки останова

**bd <bpid>**: отключение точки останова

**be <bpid>**: Включить точку останова

**bc <bpid>**: очистка точки останова

**bpc**: установка точки останова в текущей строке

**bl**: вывод списка точек останова

## Управление потоком — навигация

Используйте следующие команды, чтобы перейти вперед в скрипте.

**p**: шаг

**t**: шаг в

**g**. Продолжить скрипт

**gu**: выход

## Кадров

Используйте следующие команды для работы с кадрами.

**Индекс**> .frame: переключение на индекс номера кадров <<>

**.f+**: переключение на следующий кадр стека

**.f+**: переключение на предыдущий кадр стека

## Выход из программы

**Используйте команду .detach** для отсоединения отладчика JavaScript.

```
dbgcmd
```

```
>>> Debug [DebuggableSample 34:5] >.detach
.detach
Debugger has been detached from script!
```

**Используйте команду q**, чтобы выйти из отладчика JavaScript.

```
dbgcmd
```

```
>>> Debug [<NONE> ] >q
q
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .scriptlist (список загруженных скриптов)

Статья • 05.04.2024

Команда **.scriptlist** содержит загруженные скрипты.

```
dbgcmd

.scriptlist
```

## Параметры

нет

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Команда .scriptlist перечисляет все скрипты, загруженные с помощью команды .scriptload.

Если TestScript был успешно загружен с помощью .scriptload, команда .scriptlist будет отображать имя загруженного скрипта.

```
dbgcmd

0:000> .scriptlist
Command Loaded Scripts:
    JavaScript script from 'C:\WinDbg\Scripts\TestScript.js'
```

## Требования

Перед использованием любой из команд скрипта необходимо загрузить поставщика сценариев.

# См. также

Скрипты отладчика JavaScript

**.scriptload (Load Script)**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .scriptload (Load Script)

Статья • 05.04.2024

Команда **.scriptload** загружает и выполняет указанный файл скрипта.

```
dbgcmd

.scriptload ScriptFile
```

## Параметры

*ScriptFile*
Указывает имя загружаемого файла скрипта. *ScriptFile* должен включать расширение имени файла .js. Можно использовать абсолютные или относительные пути. Относительные пути относятся к каталогу, в который был запущен отладчик. Пути к файлам, содержащие пробелы, не поддерживаются.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Команда .scriptload загружает скрипт и выполняет скрипт. Следующая команда показывает успешную загрузку TestScript.js.

```
dbgcmd

0:000> .scriptload C:\WinDbg\Scripts\TestScript.js
JavaScript script successfully loaded from 'C:\WinDbg\Scripts\TestScript.js'
```

Если в начальной загрузке и выполнении скрипта возникли ошибки, ошибки будут отображаться в консоли, включая номер строки и сообщение об ошибке.

dbgcmd

```
0:000:x86> .scriptload C:\WinDbg\Scripts\TestScript.js
0:000> "C:\WinDbg\Scripts\TestScript.js" (line 11 (@ 1)): Error
(0x80004005): Syntax error
Error: Unable to execute JavaScript script 'C:\WinDbg\Scripts\TestScript.js'
```

Команда .scriptload выполнит следующую команду в JavaScript.

- корневой код
- функция intializeScript (если она присутствует в скрипте)

Когда скрипт загружается с помощью команды .scriptload, функция intializeScript и корневой код скрипта выполняются, имена, которые присутствуют в скрипте, мостятся в корневое пространство имен отладчика (dx Debugger) и скрипт остается в памяти до тех пор, пока он не выгрузится и все ссылки на его объекты будут освобождены.

Скрипт может предоставлять новые функции для вычислителя выражений отладчика, изменять объектную модель отладчика или работать в качестве визуализаторов так же, как визуализатор NatVis. Дополнительные сведения о NavVis и отладчике см. в dx **(Display NatVis Expression).**

Дополнительные сведения о работе с JavaScript см. в статье "Скрипт отладчик JavaScript". Дополнительные сведения о объектах отладчика см. в разделе "Собственные объекты" в расширениях JavaScript.

## Требования

Перед использованием любой из команд скрипта необходимо загрузить поставщика сценариев. Используйте команду scriptproviders, чтобы убедиться, что поставщик JavaScript загружен.

dbgcmd

```
0:000> .scriptproviders
Available Script Providers:
    NatVis (extension '.NatVis')
    JavaScript (extension '.js')
```

## См. также

**.scriptunload (выгрузка скрипта)**

Скрипты отладчика JavaScript

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .scriptproviders (поставщики скриптов списка)

Статья • 05.04.2024

Команда **.scriptproviders выводит список активных поставщиков скриптов** .

```
dbgcmd
```

```
.scriptproviders
```

## Параметры

нет

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Команда .scriptproviders выводит список всех языков скриптов, которые в настоящее время понимаются отладчиком и расширением, в котором они зарегистрированы. Любой файл, заканчивающийся в ". NatVis " понимается как скрипт NatVis, и любой файл, заканчивающийся на ".js", понимается как скрипт JavaScript. Любой тип скрипта можно загрузить с помощью команды .scriptload.

В приведенном ниже примере загружаются поставщики JavaScript и NatVis.

```
dbgcmd
```

```
0:000> .scriptproviders
Available Script Providers:
    NatVis (extension '.NatVis')
    JavaScript (extension '.js')
```

## Требования

Перед использованием любой из команд скрипта необходимо загрузить поставщика сценариев.

# См. также

Скрипты отладчика JavaScript

**.scriptload (Load Script)**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .scriptrun (запуск скрипта)

Статья • 05.04.2024

Команда .scriptrun загружает и запускает JavaScript.

```
dbgcmd
```

```
.scriptrun ScriptFile
```

## Параметры

*ScriptFile*
Указывает имя файла скрипта для загрузки и выполнения. *ScriptFile* должен включать расширение имени файла .js. Можно использовать абсолютные или относительные пути. Относительные пути относятся к каталогу, в который был запущен отладчик. Пути к файлам, содержащие пробелы, не поддерживаются.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Команда .scriptrun загружает скрипт и выполняет следующий код.

- root
- intializeScript
- invokeScript

При загрузке и выполнении кода отображается сообщение подтверждения.

```
dbgcmd
```

```
0:000> .scriptrun C:\WinDbg\Scripts\helloWorld.js
JavaScript script successfully loaded from 'C:\WinDbg\Scripts\helloWorld.js'
Hello World!  We are in JavaScript!
```

Любые манипуляции с объектной моделью, сделанные скриптом, будут оставаться на месте, пока скрипт не будет впоследствии выгружен или снова запущен с другим содержимым.

В этой таблице приведены сведения о функциях, выполняемых с помощью .scriptload и .scriptrun.

Развернуть таблицу

| | .scriptload | .scriptrun |
|---|---|---|
| root | yes | yes |
| initializeScript | yes | yes |
| invokeScript | | yes |
| uninitializeScript | | |

Этот код можно использовать для просмотра функций, которые вызываются с помощью команды выполнения скрипта.

dbgcmd

```
// Root of Script
host.diagnostics.debugLog("***>; Code at the very top (root) of the script
is always run \n");


function initializeScript()
{
    // Add code here that you want to run every time the script is loaded.
    // We will just send a message to indicate that function was called.
    host.diagnostics.debugLog("***>; initializeScript was called \n");
}

function invokeScript()
{
    // Add code here that you want to run every time the script is executed.
    // We will just send a message to indicate that function was called.
    host.diagnostics.debugLog("***>; invokeScript was called \n");
}
```

Дополнительные сведения о работе с JavaScript см. в статье "Скрипт отладчик JavaScript". Дополнительные сведения о объектах отладчика см. в разделе "Собственные объекты" в расширениях JavaScript.

## Требования

Перед использованием любой из команд скрипта необходимо загрузить поставщика сценариев. **Используйте команду .load (Load Extension DLL)** для загрузки библиотеки DLL поставщика JavaScript.

dbgcmd

```
0:000> .load C:\ScriptProviders\jsprovider.dll
```

# См. также

**.scriptload (Load Script)**

Скрипты отладчика JavaScript

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .scriptunload (выгрузка скрипта)

Статья • 05.04.2024

Команда **.scriptunload** выгрузит указанный скрипт.

```
dbgcmd
```

```
.scriptunload ScriptFile
```

## Параметры

*ScriptFile*
Указывает имя файла скрипта для выгрузки. *ScriptFile* должен включать расширение имени файла .js. Можно использовать абсолютные или относительные пути. Относительные пути относятся к каталогу, в который был запущен отладчик. Пути к файлам, содержащие пробелы, не поддерживаются.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Команда .scriptunload выгружает загруженный скрипт. Используйте следующий синтаксис команды для выгрузки скрипта

```
dbgcmd
```

```
0:000:x86> .scriptunload C:\WinDbg\Scripts\TestScript.js
JavaScript script unloaded from 'C:\WinDbg\Scripts\TestScript.js'
```

Если в скрипте имеются невыполненные ссылки на объекты, содержимое скрипта будет отсоединяться, но скрипт может оставаться в памяти до тех пор, пока не будут освобождены все такие ссылки.

Дополнительные сведения о работе с JavaScript см. в статье "Скрипт отладчик JavaScript". Дополнительные сведения о объектах отладчика см. в разделе "Собственные объекты" в расширениях JavaScript.

## Требования

Перед использованием любой из команд скрипта необходимо загрузить поставщика сценариев. **Используйте команду .load (Load Extension DLL)** для загрузки библиотеки DLL поставщика JavaScript.

dbgcmd

```
0:000> .load C:\ScriptProviders\jsprovider.dll
```

# См. также

**.scriptload (Load Script)**

Скрипты отладчика JavaScript

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .secure (активация безопасного режима)

Статья • 05.04.2024

Команда **.secure** активирует или отображает состояние безопасного режима.

dbgcmd

```
.secure 1
.secure
```

## Среда

Безопасный режим можно включить только в то время как отладчик неактивен. Безопасный режим применяется только к сеансам в режиме ядра, так как, по определению, безопасный режим предотвращает операции отладки в пользовательском режиме.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения см. в разделе "Безопасный режим".

## Замечания

Чтобы активировать безопасный режим, используйте команду **.secure 1** (или **.secure** , за которой следует любое ненулевое значение).

**Команда .secure** показывает, активен ли режим безопасности в данный момент.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .send_file (отправка файла)

18.06.2025

Команда **.send_file** копирует файлы. При удаленной отладке с помощью сервера обработки он отправляет файл с компьютера интеллектуального клиента на компьютер сервера обработки.

dbgcmd

```
.send_file [-f] Source Destination
.send_file [-f] -s Destination
```

## Параметры

**-f**
Принудительное создание файла. По умолчанию **.send_file** не перезаписывает существующие файлы. Если используется параметр -f, то целевой файл всегда будет создан, и любой существующий файл с тем же именем будет перезаписан.

*Источник*
Указывает полный путь и имя файла для отправки. Если выполняется отладка с помощью сервера обработки, этот файл должен находиться на компьютере, на котором работает смарт-клиент.

*Назначение*
Указывает каталог, в котором должен быть записан файл. Если выполняется отладка с помощью сервера обработки, это имя каталога вычисляется на компьютере, на котором выполняется сервер обработки.

**-s**
Позволяет отладчику копировать все загруженные файлы символов.

## Окружающая среда

Развернуть таблицу

| Товар | Описание |
|---|---|
| Режимы | Режим пользователя |
| Цели | В прямом эфире |

| Товар | Описание |
| --- | --- |
| Платформы | Все |

# Замечания

Эта команда особенно полезна при выполнении удаленной отладки с помощью сервера обработки, но хотите начать отладку локально. В этом случае можно использовать команду .send_file -s для копирования всех файлов символов, которые отладчик использовал на сервер обработки. Затем эти файлы символов можно использовать отладчиком, работающим на локальном компьютере.

# .server (создание сервера отладки)

Статья • 19.12.2024

Команда **.server** запускает сервер отладки, позволяя удаленному подключению к текущему сеансу отладки.

> ⓘ **Важно!**
>
> При использовании удаленной отладки необходимо учитывать важные аспекты безопасности. Дополнительные сведения о включении безопасного режима см. в статье **Вопросы безопасности для средств отладки Windows**.

*Рекомендуемые методы подключения с некоторыми дополнительными* безопасности

dbgcmd

```
.server spipe:proto=Protocol,
{certuser=Cert|machuser=Cert},pipe=PipeName[,hidden][,password=Password]
.server ssl:proto=Protocol,
{certuser=Cert|machuser=Cert},port=Socket[,hidden][,password=Password]
.server ssl:proto=Protocol,
{certuser=Cert|machuser=Cert},port=Socket,clicon=Client[,password=Password]
```

*небезопасные методы подключения*

dbgcmd

```
.server npipe:pipe=PipeName[,hidden][,password=Password][,IcfEnable]
.server tcp:port=Socket[,hidden][,password=Password][,ipversion=6]
[,IcfEnable]
.server tcp:port=Socket,clicon=Client[,password=Password][,ipversion=6]
.server com:port=COMPort,baud=BaudRate,channel=COMChannel[,hidden]
[,password=Password]
```

# Параметры

*PipeName*
Если используется протокол NPIPE или SPIPE, *PipeName* — это строка, которая будет служить именем канала. Каждое имя канала должно определять уникальный сервер отладки. При попытке повторного использования имени канала появится сообщение об ошибке. *PipeName* не должны содержать пробелы или кавычки.

*PipeName* может включать числовый код **printf**-style format, например %x или %d. Отладчик заменит его идентификатором процесса отладчика. Второй такой код будет заменен идентификатором потока отладчика.

*Сокет*
Если используется протокол TCP или SSL, сокетом является номером порта сокета.

Также можно указать диапазон портов, разделенных двоеточием. Отладчик проверяет каждый порт в этом диапазоне, чтобы узнать, является ли он бесплатным. Если он находит бесплатный порт и не возникает ошибок, будет создан сервер отладки. Клиент отладки должен указать фактический порт, используемый для подключения к серверу. Чтобы определить фактический порт, используйте любой из методов, описанных в **поиск серверов отладки**; При отображении этого сервера отладки за портом будут следовать два числа, разделенные двоеточием. Первое число будет фактическим используемым портом; Второй можно игнорировать. Например, если порт был указан как порт=51:60, а порт 53 фактически использовался, результаты поиска будут отображать "port=53:60". (Если для установки обратного подключения используется параметр clicon clicon, клиент отладки может указать диапазон портов таким образом, а сервер должен указать фактически используемый порт.)

**clicon**=*клиента*
Если используется протокол TCP или SSL и указан параметр clicon **clicon**, откроется *обратное* подключение. Это означает, что сервер отладки попытается подключиться к клиенту отладки, а не позволить клиенту инициировать контакт. Это может быть полезно, если у вас есть брандмауэр, который предотвращает подключение в обычном направлении. *клиент* указывает сетевое имя компьютера, на котором существует клиент отладки или будет создан. Два начальных обратных косых шеи (`\\`) являются необязательными.

При использовании **clicon** лучше всего запустить клиент отладки перед созданием сервера отладки, хотя обычный порядок (сервер перед клиентом) также разрешен. Сервер обратного подключения не будет отображаться, когда другой отладчик отображает все активные серверы.

*COMPort*
При использовании com-протокола *COMPort* указывает используемый com-порт. Префикс COM является необязательным (например, "com2" и "2" допустимы).

*BaudRate*
Если используется протокол COM, *BaudRate* указывает скорость выполнения соединения. Разрешена любая скорость baud, поддерживаемая оборудованием.

*COMChannel*
Если используется протокол COM, *COMChannel* указывает com-канал, который будет использоваться при взаимодействии с клиентом отладки. Это может быть любое значение от 0 до 254 включительно.

протокола
Если используется протокол SSL или SPIPE, протокола указывает протокол Secure Channel (S-Channel). Это может быть любой из строк tls1, p1, ssl2 или ssl3.

*сертификат*
Если используется протокол SSL или SPIPE, *сертификат* указывает сертификат. Это может быть имя сертификата или отпечаток сертификата (строка шестнадцатеричных цифр, заданных оснастки сертификата). Если используется синтаксис **certuser**=*сертификат Cert*, отладчик будет искать сертификат в системном хранилище (хранилище по умолчанию). Если используется синтаксис **machuser**=*сертификатов*, отладчик будет искать сертификат в хранилище компьютеров. Указанный сертификат должен поддерживать проверку подлинности сервера.

**скрытые**
Предотвращает отображение сервера, когда другой отладчик отображает все активные серверы.

**password**=*Password*
Требуется, чтобы клиент отладки предоставлял указанный пароль для подключения к сеансу отладки. *пароль* может быть любой буквенно-цифровой строкой, длиной до двенадцати символов.

**ipversion=6**
(Только средства отладки для Windows 6.6.07 и более ранних версий)
Принудительно отладчик использовать IP-адрес версии 6, а не версию 4 при использовании TCP для подключения к Интернету. В Windows Vista и более поздних версиях отладчик пытается автоматически использовать IP-версию 6, что делает этот параметр ненужным.

**IcfEnable**
Вызывает отладчик включить необходимые подключения портов для tcp или именованного канала связи, если брандмауэр подключения к Интернету активен. По умолчанию брандмауэр подключения к Интернету отключает порты, используемые этими протоколами. Если **IcfEnable** используется с TCP-подключением, отладчик заставляет Windows открыть порт, указанный параметром Socket. Если **IcfEnable** используется с именованным подключением к каналу, отладчик заставляет Windows открывать порты, используемые для именованных

каналов (порты 139 и 445). Отладчик не закрывает эти порты после завершения подключения.

## Окружающая среда

Развернуть таблицу

| Пункт | Описание |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Цели | Динамический, аварийный дампа |
| Платформ | Все |

## Дополнительные сведения

Полные сведения о запуске сервера отладки см. в **активации сервера отладки**. Примеры см. в примерахклиента и сервера.

## Замечания

Эта команда преобразует текущий отладчик в сервер отладки. Это позволяет запускать сервер после запуска отладчика, в то время как -server параметр командной строки может быть выдан только при запуске отладчика.

Это позволяет клиенту отладки подключаться к текущему сеансу отладки. Обратите внимание, что можно запустить несколько серверов с помощью различных параметров, что позволяет различным типам отладчика клиентов присоединяться к сеансу.

> ⓘ **Важно!**
>
> Использование пароля с протоколом TCP, NPIPE или COM обеспечивает только небольшое количество защиты, так как пароль не шифруется. При использовании пароля вместе с протоколом SSL или SPIPE пароль шифруется. Если вы хотите установить безопасный удаленный сеанс, необходимо использовать протокол SSL или SPIPE.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .servers (перечисление серверов отладки)

Статья • 26.10.2023

Команда **.servers** выводит список всех серверов отладки, которые были установлены этим отладчиком.

```
dbgcmd
```

```
.servers
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Подробные сведения об отладке серверов см. в разделе Удаленная отладка с помощью отладчика.

# Комментарии

Выходные данные команды **.servers** перечисляют все серверы отладки, запущенные отладчиком, на котором выполняется эта команда. Выходные данные форматируются таким образом, что их можно использовать буквально в качестве аргумента для параметра командной строки -remote или вставить в диалоговое окно WinDbg.

Каждый сервер отладки идентифицируется по уникальному идентификатору. Этот идентификатор можно использовать в качестве аргумента для команды **.endsrv (End Debugging Server),** если вы хотите завершить работу сервера отладки.

Команда **.servers** не выводит список серверов отладки, запущенных на этом компьютере разными экземплярами отладчика, а также не перечисляет серверы

обработки или серверы подключения KD.

# .setdll (set Default Extension DLL)

Статья • 05.04.2024

Команда **setdll** изменяет библиотеку DLL расширения по умолчанию для отладчика.

```
dbgcmd

.setdll DLLName
!DLLName.setdll
```

## Параметры

*DLLName*
Имя и путь библиотеки DLL расширения. Если полный путь был указан при загрузке библиотеки DLL, необходимо также указать полный путь.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о загрузке, выгрузке и управлении расширениями см. в разделе "Загрузка БИБЛИОТЕК DLL расширений отладчика". Дополнительные сведения о выполнении команд расширения см. в разделе "Использование команд расширения отладчика".

## Замечания

Отладчик поддерживает библиотеку DLL расширения по умолчанию, которая неявно загружается при запуске отладчика. Это позволяет пользователю указывать команду расширения, не загружая библиотеку DLL расширения. Эта команда

позволяет изменять библиотеку DLL, загруженную в виде библиотеки DLL по умолчанию.

Когда команда выдана, отладчик сначала ищет его в расширении по умолчанию. Если совпадение не найдено, все остальные загруженные библиотеки DLL расширения выполняются в том порядке, в котором они были загружены.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .shell (командная оболочка)

Статья • 05.04.2024

Команда **.shell** запускает процесс оболочки и перенаправляет выходные данные в отладчик или в указанный файл.

dbgcmd

```
.shell [Options] [ShellCommand]
.shell -i InFile [-o OutFile [-e ErrFile]] [Options] ShellCommand
```

# Параметры

*InFile*
Указывает путь и имя файла, который будет использоваться для ввода. Если вы планируете не предоставлять входные данные после начальной команды, можно указать один дефис (-) вместо *InFile* без пробела перед дефисом.

*OutFile*
Указывает путь и имя файла, которые будут использоваться для стандартных выходных данных. Если **значение -o** **** *OutFile* опущено, выходные данные отправляются в окно команды отладчика. Если вы не хотите, чтобы эти выходные данные отображались или сохранены в файле, можно указать один дефис (-) вместо *OutFile* без пробела перед дефисом.

*ErrFile*
Указывает путь и имя файла, который будет использоваться для вывода ошибок. Если значение -e ErrFile опущено, выходные данные ошибок отправляются в то же место, что и стандартные выходные данные. Если вы не хотите, чтобы эти выходные данные отображались или сохранены в файле, можно указать один дефис (-) вместо *ErrFile* без пробела перед дефисом.

*Параметры* могут быть любым количеством следующих параметров:

**-ci** "*Команды*"
Обрабатывает указанные команды отладчика, а затем передает выходные данные в виде входного файла в запущенный процесс. Команды могут быть любым количеством команд отладчика, разделенных точкой с запятой, и заключены в кавычки.

**-x**
Приводит к тому, что отладчик полностью отсоединяется от отладчика. Это позволяет создавать процессы, которые будут продолжать выполняться даже после завершения сеанса отладки.

*ShellCommand*
Указывает командную строку приложения или команду Microsoft MS-DOS, выполняемую.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Дополнительная информация

Другие способы доступа к командной оболочке см. в разделе "Использование команд оболочки".

# Замечания

**Команда .shell** не поддерживается, если выходные данные отладчика пользовательского режима перенаправляются в отладчик ядра. Дополнительные сведения о перенаправлении выходных данных в отладчик ядра (иногда называется NTSD по KD), см. в разделе "Управление отладчиком пользовательского режима" из отладчика ядра.

Вся строка после **команды .shell** будет интерпретирована как команда Windows (даже если она содержит точку с запятой). Эта строка не должна быть заключена в кавычки. Между оболочкой **и *ShellCommand* должно быть пространство** (дополнительные ведущие пробелы игнорируются).

Выходные данные команды будут отображаться в окне команды отладчика, если **не используется параметр -o** **** *OutFile*.

**Выполнение команды .shell** без параметров активирует оболочку и оставляет ее открытой. Все последующие команды будут интерпретированы как команды Windows. В течение этого времени отладчик будет отображать сообщения, считывающие **<процесс .shell, может потребовать входных данных>**, и запрос WinDbg будет заменен запросом **ввода>** . Иногда появляется отдельное окно командной строки, когда отладчик покидает оболочку. Это окно должно игнорироваться; все входные и выходные данные будут выполнены в окне команды отладчика.

Чтобы закрыть эту оболочку и вернуться к самому отладчику, введите **выход** или **.shell_quit**. (Команда .shell_quit **более мощна**, так как она работает, даже если оболочка заморожена.)

Эту команду нельзя использовать при отладке CSRSS, так как новые процессы не могут быть созданы без активного использования CSRSS.

Флаг -ci можно использовать для выполнения одной или нескольких команд отладчика, а затем передать выходные данные процессу оболочки. Например, вы можете передать выходные данные из **команды !process 0 7** в скрипт Perl с помощью следующей команды:

```
dbgcmd

0:000> .shell -ci "!process 0 7" perl.exe parsemyoutput.pl
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .settings (Настройка параметров отладки)

Статья • 08.07.2024

Наборы **команд .settings** , изменяет, отображает, загружает и сохраняет параметры в пространстве имен Debugger.Settings.

dbgcmd

```
.settings set  namespace.setting=value
.settings set namespace.setting+=value
.settings save [file path]
.settings load file path
.settings list [namespace][-v]
.settings help
```

# Параметры

### Настройка параметров .settings

**namespace.setting=value**
Задает или изменяет параметр. При указании путей к файлам используйте экранирование косой черты, например C:\\Symbols\\.

Примеры:

`.settings set Display.PreferDMLOutput=false`

`.settings set Sources.DisplaySourceLines=true`

`.settings set Symbols.Sympath="C:\\Symbols\\"`

**namespace.setting+=value**
Указывает, что новое значение будет добавлено (вместо замены) предыдущего значения.

Пример:

`.settings set Extensions.ExtensionSearchPath+=";C:\\MyExtension\\"`

### Параметр сохранения параметров .setting

**Путь к файлу**
Сохраняет все значения в пространстве имен Debugger.Settings в указанный XML-файл.

**никакой**
Если путь к файлу не указан, параметры будут сохранены в последнем файле, который был загружен или сохранен. Если предыдущий файл не существует, файл с именем config.xml будет создан в каталоге, из который был загружен исполняемый файл отладчика.

**Параметры загрузки .setting**

**Путь к файлу**
Загружает все параметры из XML-файла параметров. Параметры загрузки изменят только параметры, определенные в этом файле. Все ранее загруженные или измененные параметры, которые не отображаются в этом файле, не будут изменены. Этот файл будет рассматриваться как путь сохранения по умолчанию до следующей операции сохранения или загрузки.

**Параметры списка .setting**

**пространство имен**
Вывод списка всех параметров в заданном пространстве имен и их значениях.

**-v**
Флаг –v приводит к отображению описания параметра.

**Параметры справки .setting**

**Никакой**
Выводит список всех параметров в пространстве имен отладчика и их описание.

**Пространство имен**
Выводит список всех параметров в заданном пространстве имен и их описание.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |

| Позиция | Description |
| --- | --- |
| Платформы | Все |

## Дополнительная информация

При запуске отладчик загружает все параметры из config.xml в каталоге исполняемого файла отладчика. В сеансе отладки можно изменить параметры с помощью предыдущей команды параметров (например, Sympath или .prefer_dml) или новых команд .settings. Для сохранения параметров в файле конфигурации параметров можно использовать параметр .settings save. Чтобы включить автосохранение, можно использовать следующую команду.

```
.settings set AutoSaveSettings=true
```

При включении автоматического сохранения параметры в пространстве имен Debugger.Settings будут автоматически сохранены при выходе из отладчика.

## Замечания

Вы можете обмениваться файлами параметров xml отладки с другими пользователями, чтобы дублировать их параметры отладки.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .show_read_failures

Статья • 05.04.2024

Команда **.show_read_failures** включает или отключает отображение сбоев чтения.

```
dbgcmd
```

```
.show_read_failures /v
.show_sym_failures /V
```

## Параметры

**/v**
Включает отображение сбоев чтения.

**/V**
Отключает отображение сбоев чтения.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .show_sym_failures

Статья • 05.04.2024

Команда **.show_sym_failures** включает или отключает отображение сбоев подстановки символов и сбоев подстановки типов.

dbgcmd

```
.show_sym_failures /s
.show_sym_failures /S
.show_sym_failures /t
.show_sym_failures /T
```

## Параметры

**/s**
Включает отображение сбоев подстановки символов.

**/S**
Отключает отображение сбоев подстановки символов.

**/T**
Включает отображение сбоев подстановки типа.

**/T**
Отключает отображение сбоев подстановки типа.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Спящий режим (приостановка отладчика)

Статья • 05.04.2024

Команда **спящего** режима приводит к приостановке работы отладчика пользовательского режима и целевому компьютеру. Эта команда используется только при управлении отладчиком пользовательского режима из отладчика ядра.

dbgcmd

```
.sleep milliseconds
```

## Параметры

*milliseconds*
Указывает длину паузы в миллисекундах.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | управление отладчиком пользовательского режима из отладчика ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения и сведения о пробуждении отладчика в спячем режиме см. в разделе "Управление отладчиком пользовательского режима" из отладчика ядра.

## Замечания

При управлении отладчиком пользовательского режима из отладчика ядра и запросом отладчика пользовательского режима отображается в отладчике ядра,

эта команда активирует спящий режим. Отладчик ядра, отладчик пользовательского режима и целевое приложение будут зависать, но остальная часть целевого компьютера станет активной.

Если вы используете эту команду в любом другом сценарии, она просто заморозит отладчик в течение определенного периода времени.

Время сна находится в миллисекундах и интерпретируется в соответствии с радиксом по умолчанию, если не используется префикс, например **0n** . Таким образом, если по умолчанию радикс равен 16, следующая команда вызовет около 65 секунд сна:

dbgcmd

```
0:000> .sleep 10000
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .sound_notify (использование звука уведомления)

Статья • 08.07.2024

Команда **.sound_notify** приводит к воспроизведению звука при входе WinDbg в состояние ожидания для команды.

dbgcmd

```
.sound_notify /ed
.sound_notify /ef File
.sound_notify /d
```

## Параметры

**/Эд**
Вызывает звук оповещений Windows по умолчанию, когда WinDbg вступает в состояние ожидания для команды.

**/эф** **** *Файл*
Вызывает звук, содержащийся в указанном файле, когда WinDbg вводит состояние ожидания для команды.

**/d**
Отключает воспроизведение звуков.

## Среда

Эта команда доступна только в WinDbg и не может использоваться в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .srcfix, .lsrcfix (использование исходного сервера)

Статья • 05.04.2024

**Команды SRCFIX и .lsrcfix** автоматически задают исходный путь, чтобы указать, что исходный сервер будет использоваться.

dbgcmd

```
.srcfix[+] [Paths]
.lsrcfix[+] [Paths]
```

## Параметры

+
Приводит к сохранению существующего исходного пути и **;srv*** к концу. + Если он не используется, то путь к существующему источнику заменяется.

*Paths*
Указывает один или несколько дополнительных путей для добавления в конец нового исходного пути.

## Среда

Команда **SRCFIX** доступна во всех отладчиках. **Команда Lsrcfix** доступна только в WinDbg и не может использоваться в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о настройке локального пути к источнику для удаленного клиента см. в разделе "**Параметры** командной строки WinDbg". Дополнительные сведения об SrcSrv см. в разделе "Использование исходного сервера". Дополнительные сведения о пути к источнику и локальном исходном пути см. в разделе "Исходный путь". Дополнительные сведения о командах, которые можно использовать при выполнении удаленной отладки с помощью отладчика, см. в разделе "Управление сеансом удаленной отладки".

## Замечания

При добавлении `srv*` в исходный путь отладчик использует SrcSrv для получения исходных файлов из расположений, указанных в файлах символов целевых модулей. Использование `srv*` в исходном пути существенно отличается от использования `srv*` в пути символов. В пути к символам можно указать расположение сервера символов вместе с `srv*` (например, `.sympath SRV*https://msdl.microsoft.com/download/symbols`). В исходном пути srv* стоит отдельно, отделяя от всех остальных элементов точкой с запятой.

При выполнении этой команды из клиента **отладки SRCFIX задает исходный путь для использования исходного сервера на сервере отладки, а Lsrcfix** выполняет то же самое на локальном компьютере.

Эти команды совпадают с **командами Srcpath (Set Source Path) и .lsrcpath (Set Local Source Path)** и элементом исходного **пути srv*** . Таким образом, следующие две команды эквивалентны:

dbgcmd

```
.srcfix[+] [Paths]
.srcpath[+] srv*[;Paths]
```

Аналогичным образом, следующие две команды эквивалентны:

dbgcmd

```
.lsrcfix[+] [Paths]
.lsrcpath[+] srv*[;Paths]
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .srcnoisy (шумная загрузка источника)

Статья • 05.04.2024

**Команда Srcnoisy** управляет уровнем детализации для загрузки исходного файла.

```
dbgcmd
```

```
.srcnoisy [Options]
```

## Параметры

*Параметры* могут быть одним из следующих вариантов:

0
Отключает отображение дополнительных сообщений.

1
Отображает сведения о ходе загрузки и выгрузки исходного файла.

2
Отображает сведения о ходе загрузки и выгрузки файла символов.

3
Отображает все сведения, отображаемые параметрами 1 и 2.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Без параметров **Srcnoisy** отображает текущее состояние шумной загрузки источника.

Шумная загрузка источника не должна быть путана с шумной загрузкой символов, которая управляется **расширением !sym шумный** и другими средствами управления параметром SYMOPT_DEBUG.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .srcpath, .lsrcpath (задать исходный путь)

Статья • 05.04.2024

Команды **Srcpath и .lsrcpath** задают или отображают путь поиска исходного файла.

dbgcmd

```
.srcpath[+] [Directory [; ...]]
.lsrcpath[+] [Directory [; ...]]
```

## Параметры

+
Указывает, что новые каталоги будут добавлены к (а не замене) предыдущего пути поиска исходного файла.

*Каталог*
Указывает один или несколько каталогов, которые нужно поместить в путь поиска. Если *каталог* не указан, отображается текущий путь. Разделите несколько каталогов с запятой.

## Среда

Команда **Srcpath** доступна во всех отладчиках. **Команда Lsrcpath** доступна только в WinDbg и не может использоваться в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения и другие способы изменения этого пути см. в разделе "Исходный путь". Дополнительные сведения о командах, которые можно использовать при выполнении удаленной отладки с помощью отладчика, см. в разделе "Управление сеансом удаленной отладки".

> **Примечание**
>
> Исходный путь в WinDbg поддерживает извлечение файлов с помощью source Link 1.0 и начиная с версии 1.2104, извлечения файлов с помощью сервера Source Link 2.0 или DebugInfoD. Дополнительные сведения о синтаксисе пути к исходному коду см. в разделе "Расширенный доступ к исходному коду".

## Замечания

При включении `srv*` в исходный путь отладчик использует SrcSrv для получения исходных файлов из расположений, указанных в файлах символов целевых модулей. Дополнительные сведения об использовании srv* в пути к источнику см. в разделе "Использование исходного сервера и **SRCFIX**".

При выполнении этой команды из клиента **отладки srcpath** задает исходный путь на сервере отладки, а **Lsrcpath** задает исходный путь на локальном компьютере.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .step_filter (настройка фильтра шагов)

Статья • 08.07.2024

Команда **.step_filter** создает список функций, пропущенных (перешагнутых) при трассировке. Это позволяет отслеживать код и пропускать только определенные функции. Его также можно использовать в исходном режиме для управления шагами при наличии нескольких вызовов функций в одной строке.

dbgcmd

```
.step_filter "FilterList"
.step_filter /c
.step_filter
```

## Параметры

**FilterList**
Указывает символы, связанные с функциями, которые необходимо выполнить. *FilterList* может содержать любое количество текстовых шаблонов, разделенных точкой с запятой. Каждый из этих шаблонов может содержать различные подстановочные знаки и описатели; Дополнительные сведения см. в синтаксисе подстановочных знаков строки. Функция, символ которой соответствует хотя бы одному из этих шаблонов, будет перешагон во время трассировки. Каждый раз при **использовании FilterList** все предыдущие списки фильтров удаляются и полностью заменяются новым списком.

**/c**
Очищает список фильтров.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Без параметров .step_filter отображает текущий список фильтров.

Как правило, команда трассировки (например, **t** или отладка windbg | **шаг в]** трассировки в вызов функции. Однако если символ, связанный с вызываемой функцией, соответствует шаблону, указанному *FilterList*, функция будет перемещена, как если бы использовалась команда шага (например, **p**).

Если указатель инструкции находится в коде, указанном в списке фильтров, любые команды трассировки или шага будут выходить из этой функции, например команду gu **или кнопку "Шаг выхода WinDbg".** Конечно, этот фильтр предотвратит трассировку такого кода в первую очередь, поэтому это произойдет только в том случае, если вы изменили фильтр или попали в точку останова.

Например, следующая команда приведет к пропуску команд трассировки во всех вызовах CRT:

dbgcmd

```
.step_filter "msvcrt!*"
```

Команда **.step_filter** наиболее полезна при отладке в исходном режиме, так как в одной исходной строке может быть несколько вызовов функций. Команды **p** и **t** нельзя использовать для разделения этих вызовов функций.

Например, в следующей строке команда t **будет выполняться как в GetTickCount,** так и в printf, а **команда p** будет выполнять оба вызова функции:

dbgcmd

```
printf( "%x\n", GetTickCount() );
```

Команда **.step_filter** позволяет отфильтровать один из этих вызовов при трассировке в другую.

Так как функции идентифицируются по символам, один фильтр может включать весь модуль. Это позволяет отфильтровать функции платформы, например вызовы Microsoft Foundation Classes (MFC) или Active Template Library (ATL).

При отладке в режиме сборки каждый вызов находится в другой строке, поэтому можно выбрать, следует ли выполнять пошаговые или трассировки по строкам. Поэтому **.step_filter** не очень полезно в режиме сборки.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .suspend_ui (приостановка интерфейса WinDbg)

Статья • 05.04.2024

Команда **.suspend_ui** приостанавливает обновление окон сведений об отладке WinDbg.

dbgcmd

```
.suspend_ui 0
.suspend_ui 1
.suspend_ui
```

## Параметры

**0**
Приостанавливает обновление окон сведений об отладке WinDbg.

**1**
Включает обновление окон сведений об отладке WinDbg.

## Среда

Эта команда доступна только в WinDbg и не может использоваться в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Замечания

Без каких-либо параметров .suspend_ui отображает, приостановлены ли в настоящее время окна сведений отладки.

По умолчанию окна сведений отладки обновляются при каждом отображении информации.

Приостановка обновления этих окон может ускорить WinDbg во время последовательности быстрых операций, например при трассировке или пошаговом выполнении много раз в быстром последовательности.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Symfix (задать путь к хранилищу символов)

Статья • 05.04.2024

Команда **Symfix** автоматически задает путь к символам, указывающим на хранилище символов Майкрософт.

```
dbgcmd
```

```
.symfix[+] [LocalSymbolCache]
```

## Параметры

+
Приводит к добавлению пути в хранилище символов Майкрософт к существующему пути символов. Если это не включено, то путь к существующему символу заменяется.

*LocalSymbolCache*
Указывает каталог, используемый в качестве локального кэша символов. Если этот каталог не существует, он будет создан при запуске копирования файлов на сервере символов. Если *LocalSymbolCache* опущен, будет использоваться подкаталог каталога установки отладчика.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения см. в разделе "Использование серверов символов и хранилищ символов".

## Замечания

В следующем примере показано, как использовать **Symfix** для задания нового пути символа, указывающего на хранилище символов Майкрософт.

dbgcmd

```
3: kd> .symfix c:\myCache
3: kd> .sympath
Symbol search path is: srv*
Expanded Symbol search path is:
cache*c:\myCache;SRV*https://msdl.microsoft.com/download/symbols
```

В следующем примере показано, как использовать **Symfix+** для добавления существующего пути символа с путем, указывающим на хранилище символов Майкрософт.

dbgcmd

```
3: kd> .sympath
Symbol search path is: c:\someSymbols
Expanded Symbol search path is: c:\somesymbols
3: kd> .symfix+ c:\myCache
3: kd> .sympath
Symbol search path is: c:\someSymbols;srv*
Expanded Symbol search path is:
c:\somesymbols;cache*c:\myCache;SRV*https://msdl.microsoft.com/download/symb
ols
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .symopt (задать параметры символов)

Статья • 05.04.2024

Наборы **команд Symopt** или отображение параметров символов.

```
dbgcmd
```

```
.symopt+ Flags
.symopt- Flags
.symopt
```

## Параметры

+
Приводит к настройке параметров символов, *заданных флагами* . Если **symopt** используется с *флагами* , но без знака плюса или минуса, предполагается знак плюса.

-
Приводит к очистке параметров символов, *указанных флагами* .

*Flags*
Указывает параметры символов, которые необходимо изменить. *Флаги* должны быть суммой битовых флагов этих параметров символов.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Список и описание каждого параметра символа, его битового флага и других методов настройки и очистки этих параметров см. в разделе "Параметры

символов".

## Замечания

Без аргументов symopt отображает текущие параметры символов.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .sympath (задать путь к символам)

Статья • 05.04.2024

Команда **Sympath** задает или изменяет путь к символам. Путь к символам указывает расположения, в которых отладчик ищет файлы символов.

dbgcmd

```
.sympath[+] [Path [; ...]]
```

## Параметры

+
Указывает, что новые расположения будут добавлены (вместо замены) предыдущего пути поиска символов.

*Путь*
Полный путь или список полных путей. Несколько путей разделены точкой с запятой. Если *путь* опущен, отображается текущий путь символа.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения и другие способы изменения этого пути см. в разделе "Путь к символам".

## Замечания

Новые сведения о символах не загружаются при изменении пути к символам. Для перезагрузки символов можно использовать **команду reload (reload Module).**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .thread (настройка контекста регистрации)

Статья • 05.04.2024

Команда **.thread** указывает, какой поток будет использоваться для контекста регистрации.

```
dbgcmd
```

```
.thread [/p [/r] ] [/P] [/w] [Thread]
```

## Параметры

**/p**
(только для динамической отладки) Если этот параметр включен и *поток* не является ненулевой, все записи таблицы страницы перехода (PTEs) для процесса владения этим потоком будут автоматически преобразованы в физические адреса перед доступом. Это может привести к замедлению, так как отладчику придется искать физические адреса для всей памяти, используемой этим процессом, и значительное количество данных может потребоваться передать по отладочному кабелю. (Это поведение совпадает с **поведением. cache forcedecodeuser**.)

**Если параметр /p** включен и *поток* равен нулю или опущен, этот перевод будет отключен. (Это поведение совпадает с **поведением. cache noforcedecodeuser**.)

**/R**
(только для динамической отладки) **Если параметр /r** включен вместе с параметром **/p** , символы пользовательского режима для процесса владения этим потоком будут перезагружены после установки контекстов процесса и регистрации. (Это поведение совпадает с **поведением. перезагрузить /user**.)

**/P**
(только для динамической отладки) Если этот параметр включен и *поток* не является ненулевой, все записи таблицы страниц перехода (PTEs) будут автоматически преобразованы в физические адреса перед доступом. **В отличие от параметра /p**, это преобразует PTEs для всех процессов пользовательского и ядра, а не только для процесса владения этим потоком. Это может привести к замедлению, так как отладчику придется искать физические адреса для всей используемой памяти, и может потребоваться передать огромное количество

данных по отладочному кабелю. (Это поведение совпадает с **поведением. cache forcedecodeptes**.)

**/w**
(только 64-разрядная отладка ядра) Изменяет активный контекст для потока в 32-разрядный контекст WOW64. Указанный поток должен выполняться в процессе с состоянием WOW64.

*Поток*
Адрес потока. Если это опущено или ноль, контекст потока сбрасывается в текущий поток.

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

# Дополнительная информация

Дополнительные сведения о контексте регистрации и других параметрах контекста см. в разделе "Изменение контекстов".

# Замечания

Как правило, при отладке ядра только видимые регистры связаны с текущим потоком.

Команда **.thread** указывает отладчику ядра использовать указанный поток в качестве контекста регистра. После выполнения этой команды отладчик получит доступ к наиболее важным регистрам и трассировке стека для этого потока. Этот контекст регистра сохраняется, пока целевой объект не будет выполнять или использовать другую команду контекста регистра (**.thread**, **CXR** или **.trap).** Полные сведения см. в разделе "Регистрировать контекст ".

Параметр **/w** можно использовать только в 64-разрядных сеансах отладки ядра в потоке, работающем в процессе с состоянием WOW64. Полученный контекст будет последним контекстом, запоминаемым WOW64; Обычно это последний код пользовательского режима, *выполняемый Thread*. Этот параметр можно использовать только в том случае, если целевой объект находится в собственном режиме компьютера. Например, если целевой объект работает на 64-разрядном компьютере, который эмулирует процессор на основе x86 с помощью WOW64, этот параметр нельзя использовать. **Использование параметра /w** приведет к автоматическому переключении режима компьютера на процессор на основе x86.

Эта команда фактически не изменяет текущий поток. Другими словами, расширения, такие как **!thread** и **!teb** по-прежнему по умолчанию для текущего потока, если с ними не используются аргументы.

Ниже приведен пример. **Используйте расширение !process** для поиска адреса требуемого потока. (В этом случае **!process 0 0** используется для перечисления всех процессов, то **!process** используется второй раз для перечисления всех потоков для требуемого процесса.)

dbgcmd

```
kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS fe5039e0  SessionId: 0  Cid: 0008    Peb: 00000000  ParentCid: 0000
    DirBase: 00030000  ObjectTable: fe529a88  TableSize: 145.
    Image: System

.....

PROCESS ffaa5280  SessionId: 0  Cid: 0120    Peb: 7ffdf000  ParentCid: 01e0
    DirBase: 03b70000  ObjectTable: ffaa4e48  TableSize:  23.
    Image: winmine.exe

kd> !process ffaa5280
PROCESS ffaa5280  SessionId: 0  Cid: 0120    Peb: 7ffdf000  ParentCid: 01e0
    DirBase: 03b70000  ObjectTable: ffaa4e48  TableSize:  23.
    Image: winmine.exe
    VadRoot ffaf6e48 Clone 0 Private 50. Modified 0. Locked 0.
    DeviceMap fe502e88
    Token                             e1b55d70

.....

        THREAD ffaa43a0  Cid 120.3a4  Teb: 7ffde000  Win32Thread: e1b4fea8
WAIT: (WrUserRequest) UserMode Non-Alertable
            ffadc6a0  SynchronizationEvent
        Not impersonating
        Owning Process ffaa5280
        WaitTime (seconds)      24323
```

```
        Context Switch Count    494                 LargeStack

.....
```

Теперь используйте **команду .thread** с адресом требуемого потока. Это задает контекст регистрации и позволяет проверять важные регистры и стек вызовов для этого потока.

dbgcmd

```
kd> .thread ffaa43a0
Using context of thread ffaa43a0

kd> r
Last set context:
eax=00000000 ebx=00000000 ecx=00000000 edx=00000000 esi=00000000
edi=00000000
eip=80403a0d esp=fd581c2c ebp=fd581c60 iopl=0         nv up di pl nz na pe
nc
cs=0000  ss=0000  ds=0000  es=0000  fs=0000  gs=0000
efl=00000000
0000:3a0d ??              ???

kd> k
  *** Stack trace for last set context - .thread resets it
ChildEBP RetAddr
fd581c38 8042d61c ntoskrnl!KiSwapThread+0xc5
00001c60 00000000 ntoskrnl!KeWaitForSingleObject+0x1a1
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .time (отображаемое системное время)

Статья • 05.04.2024

Команда **.time** отображает сведения о системных переменных времени.

dbgcmd

```
.time [-h Hours]
```

## Параметры

**-h** **** *Hours*
Указывает смещение от среднего времени Greenwich в часах. Отрицательное значение *часов* должно быть заключено в скобки.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Системные переменные времени управляют поведением счетчика производительности.

Ниже приведен пример в режиме ядра:

dbgcmd

```
kd> .time
Debug session time: Wed Jan 31 14:47:08 2001
System Uptime: 0 days 2:53:56
```

Ниже приведен пример в пользовательском режиме:

dbgcmd

```
0:000> .time
Debug session time: Mon Apr 07 19:10:50 2003
System Uptime: 4 days 4:53:56.461
Process Uptime: 0 days 0:00:08.750
  Kernel time: 0 days 0:00:00.015
  User time: 0 days 0:00:00.015
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Tlist (идентификаторы процесса списка)

Статья • 08.07.2024

Команда **Tlist** выводит список всех процессов, выполняемых в системе.

```
dbgcmd

.tlist [Options][FileNamePattern]
```

## Параметры

*Параметры* могут быть любым количеством следующих параметров:

**-v**
Приводит к детализации отображения. Это включает номер сеанса, имя пользователя процесса и командную строку, используемую для запуска процесса.

**-c**
Ограничивает отображение только текущим процессом.

*FileNamePattern*
Указывает отображаемое имя файла или шаблон, который должен соответствовать имени файла процесса. Будут отображаться только те процессы, имена файлов которых соответствуют этому шаблону. *FileNamePattern* может содержать различные подстановочные знаки и описатели; дополнительные сведения см. в разделе "Синтаксис подстановочных знаков строки". Это совпадение выполняется только в отношении фактического имени файла, а не пути.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Сведения о других методах отображения процессов см. в разделе "Поиск идентификатора процесса".

## Замечания

Каждый идентификатор процесса отображается с **префиксом 0n** , чтобы подчеркнуть, что PID является десятичным числом.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .trap (отображение кадра ловушки)

Статья • 05.04.2024

Команда **.trap отображает состояние регистра кадра ловушки** , а также задает контекст регистрации.

```
dbgcmd
```

```
.trap [Address]
```

## Параметры

*Адрес*
Шестнадцатеричный адрес кадра ловушки в целевой системе. Опущение адреса не отображает сведения о кадре ловушки, но он сбрасывает контекст регистра.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения о контексте регистрации и других параметрах контекста см. в разделе "Изменение контекстов".

## Замечания

Команда **.trap** отображает важные регистры для указанного кадра ловушки.

Эта команда также указывает отладчику ядра использовать указанную запись контекста в качестве контекста регистрации. После выполнения этой команды отладчик получит доступ к наиболее важным регистрам и трассировке стека для

этого потока. Этот контекст регистра сохраняется, пока целевой объект не будет выполнять или использовать другую команду контекста регистра (.thread, **CXR или .trap).** Полные сведения см. в разделе "Регистрировать контекст ".

Это расширение часто используется при отладке ошибок проверка 0xA и 0x7F. Дополнительные сведения и пример см. в разделе **"Проверка ошибок" 0xA** (IRQL_NOT_LESS_OR_EQUAL).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .tss (сегмент состояния задачи отображения)

Статья • 05.04.2024

Команда **TSS** отображает форматируемое представление сохраненных сведений о сегменте состояния задач (TSS) для текущего процессора.

```
dbgcmd
```

```
.tss [Address]
```

## Параметры

*Адрес*
Адрес TSS.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Только режим ядра |
| Целевые объекты | live, аварийное дампа |
| Платформы | Только x86 |

## Замечания

Адрес TSS можно найти, проверив выходные данные **расширения !pcr** .

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .ttime (отображение времени потока)

Статья • 05.04.2024

Команда **Ttime** отображает время выполнения для потока.

```
dbgcmd
```

```
.ttime
```

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | только в пользовательском режиме |
| Целевые объекты | live, аварийное дампа |
| Платформы | Только x86 |

## Замечания

Эта команда работает только в пользовательском режиме. В режиме ядра вместо этого следует использовать !thread. Эта команда работает с мини-модулями в пользовательском режиме до тех пор, пока они были созданы с параметрами /mt или /ma; **дополнительные сведения см. в разделе "Файлы дампа в режиме** пользователя".

Команда **Ttime** показывает время создания потока, а также время его выполнения в режиме ядра и в пользовательском режиме.

Рассмотрим пример:

```
dbgcmd
```

```
0:000> .ttime
Created: Sat Jun 28 17:58:42 2003
Kernel:  0 days 0:00:00.131
User:    0 days 0:00:02.109
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .typeopt (задать параметры типа)

Статья • 08.07.2024

Наборы **команд .typeopt** или отображение параметров типа.

dbgcmd

```
.typeopt +Flags
.typeopt -Flags
.typeopt +FlagName
.typeopt -FlagName
.typeopt
```

## Параметры

**+**
Приводит к настройке параметров типа, *заданных флагами* или *флагами*.

**-**
Приводит к очистке параметров типа, *указанных флагами* или *флагами* .

*Флаги*
Указывает параметры типа, которые необходимо изменить. *Флаги* могут быть суммой любого из следующих значений (по умолчанию нет):

**0x1**
Отображает 16-разрядные целые значения без знака во всех окнах контрольных значений и в окне "Локальные" как тип данных UNICODE.

**0x2**
Отображает 32-разрядные целые значения во всех окнах контрольных значений и в окне "Локальные" в виде целых чисел без знака в радиксе по умолчанию.

**0x4**
Отображает целое число со знаком всех размеров во всех окнах контрольных значений и в окне "Локальные" в качестве неподписанных значений в радиксе по умолчанию.

**0x8**
Позволяет отладчику выбрать соответствующий символ с наибольшим размером, если окно "Локальные" или "Просмотр" ссылается на символ по имени, но имеется несколько символов, которые соответствуют этому имени. Размер символа определяется следующим образом: если символ является именем функции, его

размер — это размер функции в памяти. В противном случае размер символа — это размер типа данных, который он представляет.

*FlagName*
Указывает параметры типа, которые необходимо изменить. *FlagName* может быть любой из следующих строк (по умолчанию нет):

**уни**
Отображает 16-разрядные целые значения без знака во всех окнах контрольных значений и в окне "Локальные" как тип данных UNICODE. (Это имеет тот же эффект, что и **0x1**.)

**longst**
Отображает 32-разрядные целые значения во всех окнах контрольных значений и в окне "Локальные" в виде целых чисел без знака в радиксе по умолчанию. (Это имеет тот же эффект, что и **0x2**.)

**radix**
Отображает целое число со знаком всех размеров во всех окнах контрольных значений и в окне "Локальные" в качестве неподписанных значений в радиксе по умолчанию. (Это имеет тот же эффект, что и **0x4**.)

**size**
Позволяет отладчику выбрать соответствующий символ с наибольшим размером, если окно "Локальные" или "Просмотр" ссылается на символ по имени, но имеется несколько символов, которые соответствуют этому имени. Размер символа определяется следующим образом: если символ является именем функции, его размер — это размер функции в памяти. В противном случае размер символа — это размер типа данных, который он представляет. (Это имеет тот же эффект, что и **0x8**.)

# Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

# Замечания

Без аргументов .typeopt **отображает текущие параметры символов**.

Чтобы изменить радикс по умолчанию, используйте **команду n (Set Number Base).**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .выгрузка (DLL расширения выгрузки)

Статья • 05.04.2024

Команда **"Выгрузка" выгружает** библиотеку DLL расширения из отладчика.

```
dbgcmd
```

```
.unload DLLName
!DLLName.unload
```

## Параметры

*DLLName*
Указывает имя файла библиотеки DLL расширения отладчика для выгрузки. Если полный путь был указан при загрузке библиотеки DLL, необходимо также указать полный путь. Если *значение DLLName* опущено, то текущая библиотека DLL расширения выгружается.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Дополнительная информация

Дополнительные сведения о загрузке, выгрузке и управлении расширениями см. в разделе "Загрузка библиотек DLL расширений отладчика".

## Замечания

Эта команда полезна при тестировании создаваемого расширения. При повторной компиляции расширения необходимо выгрузить и загрузить новую библиотеку DLL.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .unloadall (выгрузка всех библиотек DLL расширения)

Статья • 26.10.2023

Команда **.unloadall** выгружает все библиотеки DLL расширения из отладчика в системе узла.

dbgcmd

```
.unloadall
```

## Среда

| Элемент | Описание |
|---|---|
| Режимы | Пользовательский режим, режим ядра |
| Цели | Динамический аварийный дамп |
| Платформы | Все |

## Дополнительные сведения

Дополнительные сведения о загрузке, выгрузке и управлении расширениями см. в разделе Загрузка библиотек DLL расширения отладчика.

# .urestart (отмена регистрации для перезапуска)

Статья • 26.10.2023

Команда **.urestart** отменяет регистрацию сеанса отладки для перезапуска в случае перезагрузки или сбоя приложения.

dbgcmd

```
.urestart
```

## Комментарии

Эта команда не работает для сеансов отладчика с повышенными привилегиями.

## См. также

**.rrestart**

# .wake (отладчик пробуждения)

Статья • 05.04.2024

**Команда пробуждения** приводит к концу спящего режима. Эта команда используется только при управлении отладчиком пользовательского режима из отладчика ядра.

dbgcmd

```
.wake PID
```

## Параметры

*PID*
Идентификатор процесса отладчика пользовательского режима.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | управление отладчиком пользовательского режима из отладчика ядра |
| Целевые объекты | только динамическая отладка |
| Платформы | all |

## Дополнительная информация

Дополнительные сведения см. в разделе "Управление отладчиком пользовательского режима" из отладчика ядра. Сведения о том, как найти идентификатор процесса отладчика, см. в разделе "Поиск идентификатора процесса".

## Замечания

При управлении отладчиком пользовательского режима из отладчика ядра и система находится в спячем режиме, эта команда может использоваться для

пробуждения отладчика до истечения времени ожидания таймера сна.

Эта команда не выдается в отладчике пользовательского режима на целевом компьютере или в отладчике ядра на хост-компьютере. Он должен быть выдан из третьего отладчика (KD, CDB или NTSD), работающего на целевом компьютере.

Этот отладчик может быть запущен явно для этой цели или может быть другим отладчиком, который выполняется. Однако если еще нет отладчика, проще просто использовать CDB с параметром командной **строки -wake**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .write_cmd_hist (журнал команд записи)

Статья • 05.04.2024

Команда **.write_cmd_hist** записывает всю историю окна команды отладчика в файл.

dbgcmd

```
.write_cmd_hist Filename
```

## Параметры

*Имя файла*
Указывает путь и имя файла, который будет создан.

## Среда

Эта команда доступна только в WinDbg и не может использоваться в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .writemem (запись памяти в файл)

Статья • 08.07.2024

Команда **writemem** записывает раздел памяти в файл.

```
dbgcmd
```

```
.writemem FileName Range
```

## Параметры

*Имя файла*
Указывает имя создаваемого файла. Можно указать полный путь и имя файла или просто имя файла. Если имя файла содержит пробелы, *имя файла* должно быть заключено в кавычки. Если путь не указан, используется текущий каталог.

*Диапазон*
Указывает диапазон памяти для записи в файл. Сведения о синтаксисе см. в разделе "Синтаксис адреса и диапазон адресов".

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Память копируется буквально в файл. Он не анализируется каким-либо образом.

Команда **writemem** является противоположностью команде **readmem (чтение памяти из файла).**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# .wtitle (задать название окна)

Статья • 08.07.2024

Команда **Wtitle** задает заголовок в главном окне WinDbg или в окне NTSD, CDB или KD.

```
dbgcmd
```

```
.wtitle Title
```

## Параметры

*Титул*
Заголовок, используемый для окна.

## Среда

Эта команда не может использоваться в файлах скриптов.

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Для CDB, NTSD или KD, если **команда WTitle** не использовалась, заголовок окна соответствует командной строке, используемой для запуска отладчика.

Для WinDbg, если **wtitle** не использовался, заголовок главного окна содержит имя целевого объекта. Если сервер отладки активен, его строка подключения также отображается. Если активны несколько серверов отладки, отображается только последний сервер.

При **использовании *WTitle*** *заголовок* заменяет все эти сведения. Даже если сервер отладки запущен позже, *заголовок* не изменится.

Номер версии WinDbg всегда отображается в строке заголовка окна независимо от того, используется ли эта команда.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# General Extensions

18.07.2025

В этом разделе описываются команды расширения, которые часто используются при отладке в пользовательском режиме и в режиме ядра.

Отладчик автоматически загружает соответствующую версию этих команд расширения. Если вы вручную не загружали другую версию, вам не нужно отслеживать используемые версии DLL. Дополнительные сведения о порядке поиска модуля по умолчанию см. в разделе "Использование команд расширения отладчика". Дополнительные сведения о загрузке модулей расширения см. в разделе "Загрузка библиотек DLL расширения отладчика".

В каждом разделе ссылки на команды расширения перечислены библиотеки DLL, предоставляющие следующую команду. Используйте следующие правила, чтобы определить правильный каталог для загрузки библиотеки DLL этого расширения:

- Если целевой компьютер работает под управлением Microsoft Windows XP или более поздней версии Windows, используйте winxp\kdexts.dll, winxp\ntsdexts.dll, winxp\exts.dll, winext\ext.dll, or dbghelp.dll.

# !Acl

Статья • 11.11.2023

Расширение **!acl** форматирует и отображает содержимое списка управления доступом (ACL).

Синтаксис

dbgcmd

```
!acl Address [Flags]
```

# Параметры

*Адрес*

Указывает шестнадцатеричный адрес списка ACL.

*Flags*

Отображает понятное имя списка ACL, если значение *Flags* равно 1. Это понятное имя включает тип идентификатора безопасности (SID), а также имя домена и пользователя для идентификатора безопасности.

## DLL

Exts.dll

## Дополнительные сведения

Дополнительные сведения о списках управления доступом см. в разделах **!sid**, **!sd** и Определение ACL объекта. См. также *статью Microsoft Windows Internals (Microsoft Windows Internals* ) Марка Руссиновича (Mark Russinovich) и Дэвида Соломона (David Solomon).

# Комментарии

В следующем примере показано расширение **!acl** .

Консоль

```
kd> !acl e1bf35d4 1
ACL is:
ACL is: ->AclRevision: 0x2
ACL is: ->Sbz1       : 0x0
ACL is: ->AclSize    : 0x40
ACL is: ->AceCount   : 0x2
ACL is: ->Sbz2       : 0x0
ACL is: ->Ace[0]: ->AceType: ACCESS_ALLOWED_ACE_TYPE
ACL is: ->Ace[0]: ->AceFlags: 0x0
ACL is: ->Ace[0]: ->AceSize: 0x24
ACL is: ->Ace[0]: ->Mask : 0x10000000
ACL is: ->Ace[0]: ->SID: S-1-5-21-518066528-515770016-299552555-2981724
(User: MYDOMAIN\myuser)

ACL is: ->Ace[1]: ->AceType: ACCESS_ALLOWED_ACE_TYPE
ACL is: ->Ace[1]: ->AceFlags: 0x0
ACL is: ->Ace[1]: ->AceSize: 0x14
ACL is: ->Ace[1]: ->Mask : 0x10000000
ACL is: ->Ace[1]: ->SID: S-1-5-18 (Well Known Group: NT AUTHORITY\SYSTEM)
```

# !Адрес

Статья • 05.04.2024

Расширение **!address** отображает сведения о памяти, используемой целевым процессом или целевым компьютером.

Режим пользователя

```
dbgcmd
```

```
!address Address
!address -summary
!address [-f:F1,F2,...] {[-o:{csv | tsv | 1}] | [-c:"Command"]}
!address -? | -help
```

Режим ядра

```
dbgcmd
```

```
!address Address
!address
```

# Параметры

*Адрес*
Отображает только регион адресного пространства, содержащего *адрес*.

**-Сводка**
Отображает только сводную информацию.

**-f**:*F1*, *F2*, ...
Отображает только регионы, указанные фильтрами *F1*, *F2* и т. д.

Следующие значения фильтров указывают области памяти таким образом, как целевой процесс использует их.

Развернуть таблицу

| Значение фильтра | Отображаемые области памяти |
|---|---|
| VAR | Занятые регионы. Эти регионы включают все блоки виртуального выделения, кучу SBH, память от пользовательских распределителей и все остальные регионы |

| Значение фильтра | Отображаемые области памяти |
| --- | --- |
| | адресного пространства, которые не попадают в другую классификацию. |
| Бесплатно | Свободная память. Сюда входят все памяти, которые не были зарезервированы. |
| Изображения | Память, сопоставленная с файлом, который является частью исполняемого образа. |
| Стек | Память, используемая для стеков потоков. |
| Teb | Память, используемая для блоков среды потоков (TEOB). |
| Peb | Память, используемая для блока среды обработки (PEB). |
| Куча | Память, используемая для куч. |
| Pageheap | Область памяти, используемая для полностраничной кучи. |
| CSR | Общая память CSR. |
| Actx | Память, используемая для данных контекста активации. |
| NLS | Память, используемая для таблиц поддержки национальных языков (NLS). |
| FileMap | Память, используемая для сопоставленных с памятью файлов. Этот фильтр применяется только во время динамической отладки. |

Следующие значения фильтра указывают области памяти по типу памяти.

Развернуть таблицу

| Значение фильтра | Отображаемые области памяти |
| --- | --- |
| MEM_IMAGE | Память, сопоставленная с файлом, который является частью исполняемого образа. |
| MEM_MAPPED | Память, сопоставленная с файлом, который не является частью исполняемого образа. Это включает память, сопоставленную с файлом разбиения на страницы. |

| Значение фильтра | Отображаемые области памяти |
| --- | --- |
| MEM_PRIVATE | Частная память. Эта память не предоставляется другим процессом и не сопоставляется с любым файлом. |

Следующие значения фильтра указывают области памяти по состоянию памяти.

Развернуть таблицу

| Значение фильтра | Отображаемые области памяти |
| --- | --- |
| MEM_COMMIT | Зафиксированная память. |
| MEM_FREE | Свободная память. Сюда входят все памяти, которые не были зарезервированы. |
| MEM_RESERVE | Зарезервированная память. |

Следующие значения фильтров указывают области памяти защитой, примененной к памяти.

Развернуть таблицу

| Значение фильтра | Отображаемые области памяти |
| --- | --- |
| PAGE_NOACCESS | Доступ к памяти, к которому не удается получить доступ. |
| PAGE_READONLY | Память, доступная для чтения, но не записываемая и не исполняемая. |
| PAGE_READWRITE | Память, доступная для чтения и записи, но не исполняемая. |
| PAGE_WRITECOPY | Память, которая имеет поведение копирования по записи. |
| PAGE_EXECUTE | Память, которая является исполняемой, но нечитаемой и не записываемой. |
| PAGE_EXECUTE_READ | Память, которая является исполняемой и доступной для чтения, но не доступной для записи. |
| PAGE_EXECUTE_READWRITE | Память, которая является исполняемой, читаемой и записываемой. |

| Значение фильтра | Отображаемые области памяти |
|---|---|
| PAGE_EXECUTE_WRITECOPY | Память, которая является исполняемым и имеет поведение копирования во время записи. |
| PAGE_GUARD | Память, которая выступает в качестве охранной страницы. |
| PAGE_NOCACHE | Память, которая не кэшируется. |
| PAGE_WRITECO МБ INE | Память с включенным доступом для записи. |

**-o:{csv | tsv | 1}**
Отображает выходные данные в соответствии с одним из следующих параметров.

Развернуть таблицу

| Вариант | Формат вывода |
|---|---|
| csv | Отображает выходные данные в виде значений, разделенных запятыми. |
| tsv | Отображает выходные данные в виде значений, разделенных вкладками. |
| 1 | Отображает выходные данные в формате без системы. Этот формат хорошо подходит, если !address **используется в качестве входных данных** для foreach. |

**-c**:"*Command*"
Выполняет пользовательскую команду для каждого региона памяти. Для представления выходных полей расширения !address **можно использовать следующие заполнители в команде**.

Развернуть таблицу

| Заполнитель | Поле "Output" (Выходные данные) |
|---|---|
| 1% | Базовый адрес |
| 2% | Конечный адрес + 1 |
| %3 | Размер региона |
| %4 | Тип |

| Заполнитель | Поле "Output" (Выходные данные) |
|---|---|
| 5% | State |
| %6 | Защита |
| 7% | Использование |

Например, `!address -f:Heap -c:".echo %1 %3 %5"` отображает базовый адрес, размер и состояние для каждого региона памяти типа **Кучи**.

Кавычки в команде должны предшествовать косой черте (\). Например, !address -f:Heap -c:"s -a %1 %2 \"pad\" выполняет поиск в каждой области памяти типа **Кучи** для строки "pad".

Несколько команд, разделенных точкой с запятой, не поддерживаются.

**-?**
Отображает минимальный текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Ext.dll

# Дополнительная информация

Дополнительные сведения о том, как отображать и искать память, см. в разделе "Чтение и запись памяти". Дополнительные расширения, отображающие свойства памяти, см. в разделе **!vm** (режим ядра) и **!vprot** (режим пользователя).

# Замечания

Без каких-либо параметров **расширение !address** отображает сведения обо всем адресном пространстве. Команда **!address -summary** отображает только сводку.

В режиме ядра этот расширение выполняет поиск только памяти ядра, даже если вы использовали **.process (Set Process Context)** для указания виртуального адресного пространства данного процесса. В пользовательском режиме **расширение !address** всегда ссылается на память, принадлежащую целевому процессу.

В пользовательском режиме **адрес !address** показывает характеристики региона, к которому принадлежит указанный адрес. Без параметров **!address** отображает характеристики всех регионов памяти. К этим характеристикам относятся использование памяти, тип памяти, состояние памяти и защита памяти. Дополнительные сведения о значении этой информации см. в более ранних таблицах в описании **параметра -f** .

В следующем примере используется **!address** для получения сведений о области памяти, сопоставленной с kernel32.dll.

Консоль

```
0:000> !address 75831234
Usage:                  Image
Base Address:           75831000
End Address:            758f6000
Region Size:            000c5000
Type:                   01000000MEM_IMAGE
State:                  00001000MEM_COMMIT
Protect:                00000020PAGE_EXECUTE_READ
More info:              lmv m kernel32
More info:              !lmi kernel32
More info:              ln 0x75831234
```

В этом примере используется *значение адреса* 0x75831234. На экране показано, что этот адрес находится в области памяти, которая начинается с адреса 0x75831000 и заканчивается адресом 0x758f6000. Регион имеет образ использования**, тип MEM_IMAGE, состояние MEM_COMMIT и защиту PAGE_EXECUTE_READ.** (Дополнительные сведения о значении этих значений см. в предыдущих таблицах.) На экране также перечислены три другие команды отладчика, которые можно использовать для получения дополнительных сведений об этом адресе памяти.

Если вы начинаете с адреса и пытаетесь определить сведения об этом, сведения об использовании часто являются наиболее ценными. После получения сведений об использовании можно использовать дополнительные расширения для получения дополнительных сведений об этой памяти. Например, если используется **куча, можно использовать** расширение !heap для получения дополнительных сведений.

В следующем примере команда s (Search Memory) **используется** для поиска каждого региона памяти типа **Image** для строки "Примечание".

Консоль

```
!address /f:Image /c:"s -u %1 %2 \"Note\""

*** Executing: s -u 0xab0000 0xab1000 "Note"
```

```
*** Executing: s -u 0xab1000 0xabc000 "Note"
00ab2936  004e 006f 0074 0065 0070 0061 0064 0000  N.o.t.e.p.a.d...
00ab2f86  004e 006f 0074 0065 0070 0061 0064 005c  N.o.t.e.p.a.d.\.
00ab32e4  004e 006f 0074 0065 0070 0061 0064 0000  N.o.t.e.p.a.d...
*** Executing: s -u 0xabc000 0xabd000 "Note"
. . .
```

В режиме ядра выходные данные !address **похожи на выходные данные** пользовательского режима, но содержат меньше информации. В следующем примере показаны выходные данные режима ядра.

Консоль

```
kd> !address
  804de000 - 00235000
 Usage       KernelSpaceUsageImage
          ImageName   ntoskrnl.exe

  80c00000 - 001e1000
          Usage       KernelSpaceUsagePFNDatabase

....

  f85b0000 - 00004000
          Usage       KernelSpaceUsageKernelStack
          KernelStack 817b4da0 : 324.368

 f880d000 - 073d3000
          Usage       KernelSpaceUsageNonPagedPoolExpansion
```

Значение "использование" совпадает с пользовательским режимом. ImageName указывает модуль, связанный с этим адресом. "KernelStack" отображает адрес блока ETHREAD этого потока (0x817B4DA0), идентификатор процесса (0x324) и идентификатор потока (0x368).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !analyze (WinDbg)

Статья • 08.07.2024

Расширение **!analyze** отображает сведения о текущем исключении или проверке ошибок.

**Режим пользователя**

dbgcmd

```
!analyze [-v[0..99]] [-f | -hang]
!analyze [-v[0..99]] -xml [-xmi] [-xcs] [-xmf OutputXmlFile]
!analyze -c [-load KnownIssuesFile | -unload | -help ]
```

**Режим ядра**

dbgcmd

```
!analyze [-v[0..99]] [-f | -hang]
!analyze -show BugCheckCode [BugParameters]
!analyze [-v[0..99]] -xml [-xmi] [-xcs] [-xmf OutputXmlFile]
!analyze -c [-load KnownIssuesFile | -unload | -help ]
```

# Общие параметры

**-v[0.99]**

Отображает подробные выходные данные. Дополнительные сведения можно отобразить, указав число от 0 до 99. Если число не указано, значение по умолчанию равно 1. Вы также можете указать очень подробные сведения (**-vv**) для отображения всех доступных сведений.

В пользовательском режиме -v6 **отображает то,** что было обнаружено глобально и в каждом потоке.

**-f**

Создает выходные **данные исключения !analyze** . Используйте этот параметр для просмотра анализа исключений, даже если отладчик не обнаруживает исключение.

**-вешать**

Создает выходные **данные <a1/>.** Используйте этот параметр, если целевой объект испытал проверку ошибок или исключение. Тем не менее, анализ того, почему приложение зависается более актуально для вашей проблемы. В режиме ядра !analyze **-hang анализирует блокировки,** которые система держит, а затем сканирует цепочку очередей DPC. В пользовательском режиме !analyze **-hang анализирует стек потоков, чтобы определить,** блокируют ли потоки другие потоки.

Прежде чем запускать это расширение в пользовательском режиме, рассмотрите возможность изменения текущего потока на поток, который, по вашему мнению, перестал отвечать (т. е. висел). Это изменение необходимо сделать, так как исключение может изменить текущий поток на другой.

## Показать параметр

**-показывать** `BugCheckCode` `[BugParameters]`

Отображает сведения о проверке ошибок, указанной в параметре `BugCheckCode`. `BugParameters` задает до четырех параметров проверки ошибок, разделенных пробелами. Эти параметры позволяют дополнительно уточнить поиск.

## Продолжить параметры выполнения

**-c**

Продолжает выполнение, когда отладчик сталкивается с известной проблемой. Если проблема не является известной проблемой, отладчик остается разбитым на целевой объект.

Параметр -c **можно использовать** со следующими подпараметрами. Эти подпараметры настраивают список известных проблем. Они не вызывают само собой выполнение. Пока вы не запускаете **!analyze -c-load** **** по крайней мере один раз, **!analyze -c** не действует.

**-груз** `KnownIssuesFile`
Загружает указанный файл известных проблем. `KnownIssuesFile` указывает путь и имя файла этого файла. Этот файл должен быть в формате XML.

Список известных проблем в `KnownIssuesFile` файле используется для всех последующих **команд -c**, пока не будет использоваться **-c-выгрузить или до тех пор, пока не будет использоваться -c -load** еще раз (в какой момент новые данные заменяют старые данные).

**-разгружать**

Выгрузит текущий список известных проблем.

**-help**

Отображает справку по расширению **команд расширения !analyze -c** в окне команд отладчика.

## Параметры параметра загрузки XML

**-xml**

Создает выходные данные анализа в формате XML.

**-xmi**

Добавляет сведения о модуле в выходные данные XML. Для этого параметра требуется -xml или -xmf.

**-xcs**

Добавляет контекст и кадры стека вызовов в выходные данные XML. Для этого параметра требуется -xml или -xmf.

**-xmf** `OutputXmlFile`

Записывает анализ в указанный `OutputXmlFile` в формате XML. Файл будет перезаписан, если он уже существует. Выходные данные анализа не будут создаваться в консоли или журнале, если параметр -xml также не указан.

## DLL-библиотеки

ext.dll

# Дополнительная информация

Пример анализа исключения в пользовательском режиме и ошибки остановки в режиме ядра (то есть сбоя), а также дополнительные сведения о том, как **!analysis** использует файл triage.ini, см. в разделе "Использование расширения !analysis".

# Замечания

В пользовательском режиме **!analyze** отображает сведения о текущем исключении.

В режиме **ядра !analyze** отображает сведения о последней проверке ошибок. Если возникает ошибка, **автоматически создается отображение !analyze** . Вы можете использовать **!analyze -v** для отображения дополнительных сведений. Если вы хотите просмотреть только основные параметры проверки ошибок, можно использовать команду .bugcheck (отображение данных проверки ошибок).

Для драйверов, использующих среда выполнения платформы драйвера режима пользователя (UMDF) версии 2.15 или более поздней, **!analyze** предоставляет сведения о сбоях проверки UMDF и необработанных исключениях. Эта функция доступна при отладке в динамическом режиме ядра и при анализе файла дампа памяти в режиме пользователя. При сбое **драйвера UMDF выполняется анализ** попыток идентификации ответственного драйвера.

## См. также

- Использование расширения !analyze
- Справочник по коду проверки ошибок
- Анализ аварийного дампа с помощью отладчиков Windows (WinDbg)
- Анализ файла дампа в режиме ядра с помощью WinDbg

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Asd

Статья • 05.04.2024

Расширение **!asd** отображает указанное количество записей анализа сбоев из кэша данных, начиная с указанного адреса.

```
dbgcmd
```

```
    !asd Address DataUsed
```

## Параметры

*Адрес*
Указывает адрес первой записи анализа сбоев, отображаемой.

*DataUsed*
Определяет количество отображаемых маркеров.

## DLL-библиотеки

Ext.dll

## Дополнительная информация

Для отладки расширения !dumpfa можно использовать **расширение !dumpfa.**

## Замечания

**Расширение !asd** полезно только при отладке **расширения !analyze**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Атом

Статья • 05.04.2024

Расширение **!atom отображает форматированную таблицу атомов** для указанного атома или для всех атомов текущего процесса.

dbgcmd

```
!atom [Address]
```

## Параметры

*Адрес*
Указывает шестнадцатеричный виртуальный адрес атома для отображения. Если этот параметр не указан или указан ноль, отображается таблица atom для текущего процесса. В этой таблице перечислены все атомы для процесса.

## DLL-библиотеки

Exts.dll

## Дополнительная информация

Дополнительные сведения об атомах и таблицах atom см. в документации по пакету SDK для Microsoft Windows.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bitcount

Статья • 05.04.2024

Расширение **!bitcount** подсчитывает количество "1" битов в диапазоне памяти.

```
dbgcmd
```

```
!bitcount StartAddress TotalBits
```

## Параметры

*StartAddress*
Указывает начальный адрес диапазона памяти, чей "1" бит будет подсчитываться.

*TotalBits*
Задает размер диапазона памяти в битах.

**-?**
Отображает текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Exts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !blackboxbsd

Статья • 05.04.2024

Расширение **!blackboxbsd отображает кэшированные сведения о состоянии загрузки (BSD** ), когда он доступен в файле дампа режима ядра. Данные извлекаются из кэшированных данных в дамбе режима ядра, который был сохранен во время возникновения ошибки проверка и не всегда может быть доступен.

Синтаксис

```
dbgcmd
```

```
!blackboxbsd
```

## Параметры

*Не допускается*

## DLL-библиотеки

ext.dll

## Замечания

Разработчики драйверов могут добавлять дополнительные сведения о загрузке для дампа файлов. Разработчики драйверов (и ОС) могут решить, когда добавить эти сведения в файл дампа. Это означает, что не все файлы дампа режима ядра содержат дополнительные сведения о загрузке. Дополнительные сведения см. в статье "Написание процедуры обратного вызова проверки ошибки".

### Пример выходных данных команды

```
dbgcmd
```

```
0: kd> !ext.blackboxbsd
Version: 136
Product type: 1

Auto advanced boot: FALSE
```

```
Advanced boot menu timeout: 30
Last boot succeeded: TRUE
Last boot shutdown: FALSE
Sleep in progrees: FALSE

Power button timestamp: 0
System running: TRUE
Connected standby in progress: FALSE
User shutdown in progress: FALSE
System shutdown in progress: FALSE
Sleep in progress: 6
Connected standby scenario instance id: 5
Connected standby entry reason: 12
Connected standby exit reason: 31
System sleep transitions to on: 8
Last reference time: 0x1d3645f716b79e3
Last reference time checksum: 0xb6ae84b7
Last update boot id: 6

Boot attempt count: 1
Last boot checkpoint: TRUE
Checksum: 0x7f
Last boot id: 6
Last successful shutdown boot id: 2
Last reported abnormal shutdown boot id: 5

Error info boot id: 0
Error info repeat count: 0
Error info other error count: 0
Error info code: 0
Error info other error count: 0

Power button last press time: 0x1d365b105eb9e3b
Power button cumulative press count: 6
Power button last press boot id: 6
Power button last power watchdog stage: 0x20
Power button watchdog armed: FALSE
Power button shutdown in progress: FALSE
Power button last release time: 0x1d36112993b1191
Power button cumulative release count: 5
Power button last release boot id: 6
Power button error count: 0
Power button current connected standby phase: 1
Power button transition latest checkpoint id: 9
Power button transition latest checkpoint type: 0
Power button transition latest checkpoint sequence number: 77
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !blackboxscm

Статья • 05.04.2024

Расширение **!blackboxscm** отображает сведения из Service Control Manager (SCM), когда он доступен в файле дампа режима ядра. Расширение отобразит имена всех служб, у которых есть невыполненные запросы на управление службами. Данные извлекаются из кэшированных данных в дамбе режима ядра, который был сохранен во время возникновения ошибки проверка и не всегда может быть доступен.

Синтаксис

dbgcmd

```
!blackboxscm
```

## Параметры

*Не допускается*

## DLL-библиотеки

ext.dll

## Замечания

Команды были отправлены в функцию обратного вызова LPHANDLER_FUNCTION_EX конкретной службы.
Невыполненные запросы, такие как SERVICE_CONTROL_SHUTDOWN или SERVICE_CONTROL_PRESHUTDOWN, могут отложить упорядоченное завершение работы или перезагрузку компьютера.

### Пример выходных данных команды

Во многих файлах дампа возвращается только одна служба.

dbgcmd

```
2: kd> !ext.blackboxscm
    Name: gpsvc
```

```
    Code: 15
```

Возвращаемые данные содержат сведения о двух полях.

*Имя* — *имя* службы, активной в момент возникновения дампа.

*Код* — десятичное значение dwControl, которое является выдающимся

В этом примере код 15 (или 0x0000000F) определяется как SERVICE_CONTROL_PRESHUTDOWN.

## Несколько служб

При перечислении нескольких служб только первая служба, которая указана, обычно является интересом для анализа сбоев. Это связано с тем, что SCM (Service Control Manager) ожидает последовательного завершения этих запросов, поэтому только первая служба фактически получила запрос на управление.

Дополнительные сведения о SCM см . в диспетчере управления службами.

# Дополнительная информация

Значения dwControl определяются в winsvc.h и документируются как параметры для LPHANDLER_FUNCTION_EX функции обратного вызова.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !chksym

Статья • 05.04.2024

Расширение **!chksym** проверяет допустимость модуля в файле символов.

```
dbgsyntax

    !chksym Module [Symbol]
```

## Параметры

*Модуль*
Указывает имя модуля по имени или базовому адресу.

*Символ*
Указывает имя файла символов.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Рекомендации недоступны |
| **Windows XP** | Рекомендации недоступны |
| **Windows Vista и более поздние версии** | Dbghelp.dll |

## Замечания

Если не указать символ, загрузочный символ проверяется. В противном случае, если указать путь к файлу символов PDB или DBG, загруженный символ проверяется в загруженном модуле.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !chkimg

Статья • 11.11.2023

Расширение **!chkimg** обнаруживает повреждения изображений исполняемых файлов, сравнивая их с копией в хранилище символов или другом репозитории файлов.

dbgsyntax

```
!chkimg [Options] [-mmw LogFile LogOptions] [Module]
```

# Параметры

*Параметры* Любое сочетание следующих параметров:

**-P** **** *SearchPath*
Рекурсивно выполняет поиск файла в *SearchPath* перед доступом к серверу символов.

**-F**
Исправляет ошибки в образе. Всякий раз, когда сканирование обнаруживает различия между файлом в хранилище символов и изображением в памяти, содержимое файла в хранилище символов копируется на изображение. При динамической отладке можно создать файл дампа перед выполнением расширения **!chkimg -f** .

**-Nar**
Предотвращает перемещение сопоставленного изображения файла на сервере символов. По умолчанию, когда копия файла находится на сервере символов и сопоставлена с памятью, **!chkimg** перемещает изображение файла на сервере символов. Однако при использовании параметра **-nar** изображение файла с сервера не перемещается.

Исполняемый образ, который уже находится в памяти (то есть проверяемый), перемещается, так как отладчик всегда перемещает загружаемые образы.

Этот параметр полезен, только если операционная система уже переместит исходный образ. Если образ не был перемещен, **!chkimg** и отладчик переместят образ. Этот переключатель используется редко.

**-Ss** **** *Имя раздела*
Ограничивает сканирование теми разделами, имена которых содержат строку *Имя_раздела*. Проверка будет включать любой неотбрасываемый раздел, имя которого содержит эту строку. *Значение SectionName* учитывает регистр и не может превышать 8 символов.

**-Как**
Вызывает включение в сканирование всех разделов изображения, за исключением разделов, допускающих удаление. По умолчанию (если вы не используете **-as** или **-ss**) при сканировании пропускаются разделы, доступные для записи, разделы, которые не являются исполняемыми, разделы с именем PAGE, и отбрасываемые разделы.

**-R** **** *StartAddress* **** *EndAddress*
Ограничивает сканирование диапазоном памяти, который начинается с *StartAddress* и заканчивается *на EndAddress*. В этом диапазоне проверяются все разделы, которые обычно проверяются. Если раздел частично перекрывается с этим диапазоном, сканируется только часть раздела, перекрывающаяся с этим диапазоном. Проверка ограничена этим диапазоном, даже если вы также используете параметр **-as** или **-ss** .

**-nospec**
Вызывает включение зарезервированных разделов Hal.dll и Ntoskrnl.exe. По умолчанию **!chkimg** не проверка некоторые части этих файлов.

**-noplock**
Отображает области, которые не совпадают по значению байтов 0x90 (инструкция **nop** ) и байтового значения 0xF0 (инструкция **блокировки** ). По умолчанию эти несоответствия не отображаются.

**-Np**
Вызывает распознавание исправленных инструкций.

**-D**
Отображает сводку по всем несовпадениям областей во время сканирования. Дополнительные сведения об этом сводном тексте см. в разделе Примечания.

**-Db**
Отображение несовпадения областей в формате, аналогичном **команде отладчика db**. Поэтому в каждой отображаемой строке отображается адрес первого байта в строке, за которым следует до 16 шестнадцатеричных байтовых значений. За значениями байтов сразу же следуют соответствующие значения ASCII. Все

непечатаемые символы, такие как возврат каретки и каналы строк, отображаются в виде точек (.). Несовпадение байтов помечается звездочкой (*).

**-Ло** **** *Линии*
Ограничивает число выходных строк, отображаемых **в формате -d** или **-db** , числом строк.

**-V**
Отображает подробные сведения.

**-mmw**
Создает файл журнала и записывает в этот файл действие **!chkimg** . Каждая строка файла журнала представляет собой одно несоответствие.

*Logfile*
Указывает полный путь к файлу журнала. Если указать относительный путь, путь будет относительным по отношению к текущему пути.

*LogOptions*
Указывает содержимое файла журнала. *LogOptions* — это строка, состоящая из объединения различных букв. Каждая строка в файле журнала содержит несколько столбцов, разделенных запятыми. Эти столбцы включают элементы, указанные в следующих буквах параметров в порядке, в котором буквы отображаются в строке *LogOptions* . Следующие параметры можно включить несколько раз. Необходимо включить хотя бы один вариант.

| **Параметр "Журнал"** | **Сведения, включенные в файл журнала** |
|---|---|
| v | Виртуальный адрес несоответствия |
| r | Смещение (относительный адрес) несоответствия в модуле |
| s | Символ, соответствующий адресу несоответствия. |
| S | Имя раздела, содержащего несоответствие |
| й | Правильное значение, ожидаемое в расположении несоответствия |
| w | Неверное значение, которое находилось в расположении несоответствия |

*LogOptions* также может включать некоторые или нет из следующих дополнительных параметров.

| Параметр "Журнал" | Действие |
| --- | --- |
| o | Если файл с именем *LogFile* уже существует, существующий файл перезаписывается. По умолчанию отладчик добавляет новые сведения в конец любого существующего файла. |
| *Строка* t | Добавляет дополнительный столбец в файл журнала. Каждая запись в этом столбце содержит *string*. Параметр **t***String* удобен, если вы добавляете новые сведения в существующий файл журнала и необходимо отличать новые записи от старых. Невозможно добавить пробел между **t** и *String*. Если вы используете параметр **t**l*String* , он должен быть последним параметром в *LogOptions*, так как *строка* принимается для включения всех символов, которые присутствуют до следующего пробела. |

Например, если *LogOptions* имеет значение **rSewo**, каждая строка файла журнала содержит относительный адрес и имя раздела расположения несоответствия, а также ожидаемые и фактические значения в этом расположении. Этот параметр также приводит к перезаписи всех предыдущих файлов. Параметр **-mmw** можно использовать несколько раз, если требуется создать несколько файлов журнала с разными параметрами. Одновременно можно создать до 10 файлов журналов.

*Модуль*
Указывает модуль для проверка. *Модуль* может быть именем модуля, начальным адресом модуля или любым адресом, содержащимся в модуле. Если опустить *module*, отладчик использует модуль, содержащий текущий указатель инструкции.

## DLL

| | |
| --- | --- |
| **Windows XP и более поздних версий.** | Ext.dll |

# Комментарии

При использовании **!chkimg** сравнивает изображение исполняемого файла в памяти с копией файла, который находится в хранилище символов.

Сравниваются все разделы файла, за исключением разделов, которые можно отменить, которые доступны для записи, которые не являются исполняемыми, с

именем "PAGE" или из INITKDBG. Это поведение можно изменить с помощью параметров **-ss**, **-as** или **-r** .

**!chkimg** отображает несоответствие между изображением и файлом как ошибку изображения со следующими исключениями:

- Адреса, занятые таблицей импорта адресов (IAT), не проверяются.
- Некоторые конкретные адреса в Hal.dll и Ntoskrnl.exe не проверяются, так как при загрузке этих разделов происходят определенные изменения. Чтобы проверка эти адреса, включите параметр **-nospec**.
- Если байтовое значение 0x90 присутствует в файле и если значение 0xF0 присутствует в соответствующем байте изображения (или наоборот), эта ситуация считается совпадением. Как правило, сервер символов содержит одну версию двоичного файла, которая существует как в однопроцессорных, так и в многопроцессорных версиях. На процессоре на базе x86 инструкция **блокировки** 0xF0, и эта инструкция соответствует инструкции **nop** (0x90) в версии uniprocessor. Если вы хотите, чтобы **!chkimg** отображал эту пару как несоответствие, установите параметр **-noplock** .

**Примечание** Если вы используете параметр **-f** для исправления несоответствий изображений, **!chkimg** исправит только те несоответствия, которые считаются ошибками. Например, **!chkimg** не изменяет 0x90 байт на 0xF0 байт, если не включен **параметр -noplock**.

Если включить параметр **-d** , **!chkimg** отображает сводку по всем несовпадениям областей во время сканирования. Каждое несоответствие отображается в двух строках. Первая строка включает начало диапазона, конец диапазона, размер диапазона, имя символа и смещение, соответствующее началу диапазона, а также количество байтов с момента последней ошибки (в скобках). Вторая строка заключена в квадратные скобки и включает ожидаемые шестнадцатеричные значения байтов, двоеточие, а затем шестнадцатеричные байтовые значения, которые фактически встречались на изображении. Если диапазон длиннее 8 байт, до двоеточия и после двоеточия отображаются только первые 8 байт. В следующем примере показана эта ситуация.

dbgcmd

```
be000015-be000016  2 bytes - win32k!VeryUsefulFunction+15 (0x8)
     [ 85 dd:95 23 ]
```

Иногда драйвер изменяет часть ядра Microsoft Windows с помощью перехватчиков, перенаправления или других методов. Даже драйвер, который больше не находится в стеке, мог изменить часть ядра. Расширение **!chkimg** можно использовать в качестве средства сравнения файлов, чтобы определить, какие части ядра Windows (или любой другой образ) изменяются драйверами и как именно изменяются части. Это сравнение наиболее эффективно для полных файлов дампа.

## Проверка каждого загруженного модуля

Вы также можете использовать **!chkimg** вместе с **расширением !for_each_module** для проверка образа каждого загруженного модуля. В следующем примере показана эта ситуация.

dbgcmd

```
!for_each_module !chkimg @#ModuleName
```

## Пример !analyze

Предположим, что вы столкнулись с ошибкой проверка, например, и для начала **используете !analyze**.

dbgcmd

```
kd> !analyze
....
BugCheck 1000008E, {c0000005, bf920e48, baf75b38, 0}
Probably caused by : memory_corruption
CHKIMG_EXTENSION: !chkimg !win32k
....
```

В этом примере выходные данные **!analyze** предполагают повреждение памяти и включают строку CHKIMG_EXTENSION, предполагающую, что Win32k.sys может быть поврежденным модулем. (Даже если эта строка отсутствует, можно рассмотреть возможные повреждения в модуле поверх стека.) Начните с использования **!chkimg** без параметров, как показано в следующем примере.

dbgcmd

```
kd> !chkimg win32k
Number of different bytes for win32k: 31
```

В следующем примере показано, что действительно есть повреждения памяти. Используйте **!chkimg -d** , чтобы отобразить все ошибки для модуля Win32k.

dbgcmd

```
kd> !chkimg win32k -d
    bf920e40-bf920e46  7 bytes - win32k!HFDBASIS32::vSteadyState+1f
        [ 78 08 d3 78 0c c2 04:00 00 00 00 00 01 00 ]
    bf920e48-bf920e5f  24 bytes - win32k!HFDBASIS32::vHalveStepSize (+0x08)
        [ 8b 51 0c 8b 41 08 56 8b:00 00 00 00 00 00 00 00 ]
Number of different bytes for win32k: 31
```

При попытке дизассемблировать поврежденное изображение второго раздела, который находится в списке, могут возникнуть следующие выходные данные.

dbgcmd

```
kd> u  win32k!HFDBASIS32::vHalveStepSize
win32k!HFDBASIS32::vHalveStepSize:
bf920e48 0000             add     [eax],al
bf920e4a 0000             add     [eax],al
bf920e4c 0000             add     [eax],al
bf920e4e 0000             add     [eax],al
bf920e50 7808            js win32k!HFDBASIS32::vHalveStepSize+0x12
(bf920e5a)
bf920e52 d3780c           sar     dword ptr [eax+0xc],cl
bf920e55 c20400           ret     0x4
bf920e58 8b510c           mov     edx,[ecx+0xc]
```

Затем используйте **!chkimg -f** , чтобы устранить повреждение памяти.

dbgcmd

```
kd> !chkimg win32k -f
Warning: Any detected errors will be fixed to what we expect!
Number of different bytes for win32k: 31 (fixed)
```

Теперь можно разобрать исправленное представление и просмотреть внесенные изменения.

dbgcmd

```
kd> u  win32k!HFDBASIS32::vHalveStepSize
win32k!HFDBASIS32::vHalveStepSize:
bf920e48 8b510c           mov     edx,[ecx+0xc]
bf920e4b 8b4108           mov     eax,[ecx+0x8]
bf920e4e 56               push    esi
bf920e4f 8b7104           mov     esi,[ecx+0x4]
```

```
bf920e52 03c2              add     eax,edx
bf920e54 c1f803            sar     eax,0x3
bf920e57 2bf0              sub     esi,eax
bf920e59 d1fe              sar     esi,1
```

## Анализ повреждения хранилища и памяти

Случайное повреждение файла и памяти может быть трудно исследовать. Одним из средств, которые следует учитывать в некоторых ситуациях, является включение дополнительной проверки памяти, например с помощью средства проверки драйверов. Сведения о средство проверки драйверов см. в разделе Средство проверки драйверов.

Для тестирования физической памяти используйте средство диагностики памяти Windows. Его использование и другие общие методы описаны в разделе Данные синего экрана.

Используйте служебную программу сканирования диска для выявления ошибок файловой системы. Выберите и удерживайте (или щелкните правой кнопкой мыши) диск, который требуется проверить, и выберите **Свойства**. Выберите **Сервис**. Нажмите кнопку **Проверить сейчас** .

# !cppexr

Статья • 05.04.2024

Расширение **!cppexr** отображает содержимое записи исключений C++.

```
dbgsyntax

    !cppexr Address
```

## Параметры

*Адрес*
Указывает адрес отображаемой записи исключений C++.

## DLL-библиотеки

Ext.dll

## Дополнительная информация

Дополнительные сведения об исключениях см. в документации по управлению исключениями и событиями, комплекте драйверов Windows (WDK), документации по пакету SDK для Windows и *внутренним* службам Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном. **Используйте команду EXR** для отображения других записей исключений.

## Замечания

Расширение **!cppexr** отображает сведения, связанные с исключением C++, с которыми сталкивается целевой объект, включая код исключения, адрес исключения и флаги исключений. Это исключение должно быть одним из стандартных исключений C++, определенных в Msvcrt.dll.

Обычно параметр Address можно получить *с помощью **команды !analyze -v***.

**Расширение !cppexr** полезно для определения типа исключения C++.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Cpuid

Статья • 05.04.2024

Расширение **!cpuid** отображает сведения о процессорах в системе.

```
dbgsyntax
```

```
!cpuid [Processor]
```

## Параметры

*Процессор*
Указывает обработчик, сведения которого будут отображаться. Если этот параметр не указан, отображаются все процессоры.

## DLL-библиотеки

Ext.dll

## Дополнительная информация

Дополнительные сведения об отладке многопроцессорных компьютеров см. в разделе "Синтаксис многопроцессора".

## Замечания

**Расширение !cpuid** работает во время динамической отладки пользовательского режима или режима ядра, отладки локального ядра и отладки файлов дампа. Однако мини-файлы в режиме пользователя содержат только сведения о активном процессоре.

При отладке в пользовательском режиме **расширение !cpuid** описывает компьютер, на котором запущено целевое приложение. В режиме ядра он описывает целевой компьютер.

В следующем примере показано это расширение.

```
dbgcmd
```

```
kd> !cpuid
CP  F/M/S  Manufacturer        MHz
 0  6,5,1  GenuineIntel        700
 1  8,1,5  AuthenticAMD        700
```

Столбец **CP** дает номер процессора. (Эти числа всегда являются последовательными, начиная с нуля). Столбец **производителя указывает изготовителя** процессора. Столбец **MHz** указывает скорость процессора, если она доступна.

Для процессора на основе x86 или процессора на основе **x64 столбец F** отображает номер семейства процессоров, **столбец M** отображает номер модели процессора, а **столбец S** отображает размер шага.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Cs

Статья • 05.04.2024

Расширение **!cs** отображает один или несколько критически важных разделов или целое дерево критически важных разделов.

dbgsyntax

```
!cs [-s] [-l] [-o]
!cs [-s] [-o] Address
!cs [-s] [-l] [-o] StartAddress EndAddress
!cs [-s] [-o] -d InfoAddress
!cs [-s] -t [TreeAddress]
!cs -?
```

# Параметры

Развернуть таблицу

| Параметр | Описание |
|---|---|
| **-s** | Отображает трассировку стека инициализации каждого критического раздела, если эта информация доступна. |
| **-L** | Отображение только заблокированных критических разделов. |
| **-o** | Отображает стек владельца для любого заблокированного критического раздела, отображаемого. |
| *Адрес* | Указывает адрес отображаемого критического раздела. Если этот параметр не указан, отладчик отображает все критические разделы в текущем процессе. |
| *StartAddress* | Указывает начало диапазона адресов для поиска критически важных разделов. |
| *EndAddress* | Указывает конец диапазона адресов для поиска критически важных разделов. |
| **-d** | Отображает критические разделы, связанные с DebugInfo. |
| *InfoAddress* | Указывает адрес DebugInfo. |
| **-t** | Отображает дерево критически важных разделов. Прежде чем использовать **параметр -t** , необходимо активировать средство проверки приложений для целевого процесса и выбрать **параметр "Проверить использование** блокировки". |

| Параметр | Описание |
| --- | --- |
| *TreeAddress* | Указывает адрес корневого дерева критически важных разделов. Если этот параметр опущен или указать ноль, отладчик отображает дерево критически важных разделов для текущего процесса. |
| -? | Отображает текст справки для этого расширения в окне команды отладчика. |

## DLL-библиотеки

Exts.dll

# Дополнительная информация

Другие команды и расширения, которые могут отображать критически важные сведения о разделе, см. в разделе "Отображение критического раздела". Дополнительные сведения о критически важных разделах см. в документации по пакету SDK для Microsoft Windows, документации по комплекту драйверов Windows (WDK) и *внутренним* службам Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

### Замечания

Расширение **!cs** требует полных символов (включая сведения о типе) для отлаживаемого процесса и для Ntdll.dll.

В следующих примерах показано, как использовать **!cs**. Следующая команда отображает сведения о критическом разделе по адресу 0x7803B0F8 и отображает трассировку стека инициализации.

dbgcmd

```
0:001> !cs -s 0x7803B0F8
Critical section   = 0x7803B0F8 (MSVCRT!__app_type+0x4)
DebugInfo          = 0x6A262080
NOT LOCKED
LockSemaphore      = 0x0
SpinCount          = 0x0

Stack trace for DebugInfo = 0x6A262080:

0x6A2137BD: ntdll!RtlInitializeCriticalSectionAndSpinCount+0x9B
0x6A207A4C: ntdll!LdrpCallInitRoutine+0x14
```

```
0x6A205569: ntdll!LdrpRunInitializeRoutines+0x1D9
0x6A20DCE1: ntdll!LdrpInitializeProcess+0xAE5
```

Следующая команда отображает сведения о критическом разделе, в котором DebugInfo находится в адресе 0x6A262080.

dbgcmd

```
0:001> !cs -d 0x6A262080
DebugInfo          = 0x6A262080
Critical section   = 0x7803B0F8 (MSVCRT!__app_type+0x4)
NOT LOCKED
LockSemaphore      = 0x0
SpinCount          = 0x0
```

Следующая команда отображает сведения обо всех активных критических разделах в текущем процессе.

dbgcmd

```
## 0:001> !cs

DebugInfo          = 0x6A261D60
Critical section   = 0x6A262820 (ntdll!RtlCriticalSectionLock+0x0)
LOCKED
LockCount          = 0x0
OwningThread       = 0x460
RecursionCount     = 0x1
LockSemaphore      = 0x0
## SpinCount          = 0x0

DebugInfo          = 0x6A261D80
Critical section   = 0x6A262580 (ntdll!DeferedCriticalSection+0x0)
NOT LOCKED
LockSemaphore      = 0x7FC
## SpinCount          = 0x0

DebugInfo          = 0x6A262600
Critical section   = 0x6A26074C (ntdll!LoaderLock+0x0)
NOT LOCKED
LockSemaphore      = 0x0
## SpinCount          = 0x0

DebugInfo          = 0x77fbde20
Critical section   = 0x77c8ba60 (GDI32!semColorSpaceCache+0x0)
LOCKED
LockCount          = 0x0
OwningThread       = 0x00000dd8
RecursionCount     = 0x1
LockSemaphore      = 0x0
## SpinCount          = 0x00000000
```

```
...
```

Следующая команда отображает дерево критически важных разделов.

dbgcmd

```
0:001> !cs -t

Tree root 00bb08c0

Level     Node       CS    Debug  InitThr EnterThr  WaitThr TryEnThr
LeaveThr EnterCnt  WaitCnt
##


    0 00bb08c0 77c7e020 77fbcae0      4c8      4c8        0        0
4c8        c        0
 1 00dd6fd0 0148cfe8 01683fe0      4c8      4c8        0        0      4c8
2        0
 2 00bb0aa0 008e8b84 77fbcc20      4c8        0        0        0        0
0        0
    3 00bb09e0 008e8704 77fbcba0      4c8        0        0        0
0        0        0
    4 00bb0a40 008e8944 77fbcbe0      4c8        0        0        0
0        0        0
    5 00bb0a10 008e8824 77fbcbc0      4c8        0        0        0
0        0        0
    5 00bb0a70 008e8a64 77fbcc00      4c8        0        0        0
0        0        0
    3 00bb0b00 008e8dc4 77fbcc60      4c8        0        0        0
0        0        0
    4 00bb0ad0 008e8ca4 77fbcc40      4c8        0        0        0
0        0        0
    4 00bb0b30 008e8ee4 77fbcc80      4c8        0        0        0
0        0        0
    5 00dd4fd0 0148afe4 0167ffe0      4c8        0        0        0
0        0        0
    2 00bb0e90 77c2da98 00908fe0      4c8      4c8        0        0
4c8       3a        0
 3 00bb0d70 77c2da08 008fcfe0      4c8        0        0        0        0
0        0
```

Следующие элементы отображаются в этом **отображении !cs -t** :

- **InitThr** — это идентификатор потока для потока, который инициализировал CS.
- **EnterThr** — это идентификатор потока, который в последний раз вызывал **ВводCriticalSection** .

- **WaitThr** — это идентификатор потока, который нашел критически важный раздел, принадлежащий другому потоку, и ждал его в последний раз.
- **TryEnThr** — это идентификатор потока, который в последний раз вызывал **TryEnterCriticalSection** .
- **LeaveThr** — это идентификатор потока, который в последний раз называется **LeaveCriticalSection**
- **EnterCnt** — это число **enterCriticalSection**.
- **WaitCnt** — это количество спорных запросов.

## См. также

**!ntsdexts.locks**

Отображение критического раздела

Критические истечение времени ожидания разделов (режим пользователя)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Cxr

Статья • 05.04.2024

Команда **расширения !cxr** устарела. **Вместо этого используйте команду CXR (отображаемая запись** контекста).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Dh

Статья • 05.04.2024

Расширение **!dh** отображает заголовки для указанного изображения.

```
dbgcmd
```

```
!dh [Options] Address
!dh -h
```

## Параметры

*Параметры* любого из следующих параметров:

**-f**
Отображает заголовки файлов.

**-s**
Отображает заголовки разделов.

**-a**
Отображает все сведения о заголовке.

*Адрес*
Указывает шестнадцатеричный адрес изображения.

**-h**
Отображает текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Dbghelp.dll Kdextx86.dll Ntsdexts.dll |
| **Windows XP и более поздние версии** | Dbghelp.dll |

## Замечания

Расширение **!lmi** извлекает наиболее важные сведения из заголовка изображения и отображает его в кратком формате сводки. Это расширение чаще всего полезно,

чем **!dh**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Библиотеки dll

Статья • 05.04.2024

Расширение **!dlls** отображает записи таблицы всех загруженных модулей или всех модулей, которые используют указанный поток или процесс.

dbgcmd

```
!dlls [Options] [LoaderEntryAddress]
!dlls -h
```

# Параметры

*Параметры* определяют уровень выходных данных. Этот параметр может быть любым сочетанием следующих значений:

**-f**
Отображает заголовки файлов.

**-s**
Отображает заголовки разделов.

**-a**
Отображает полные сведения о модуле. (Этот параметр эквивалентен -f -s.)

**-c** **** *ModuleAddress*
Отображает модуль, содержащий *ModuleAddress*.

**-i**
Сортирует отображение по порядку инициализации.

**-L**
Сортирует отображение по заказу загрузки. Эта ситуация используется по умолчанию.

**-m**
Сортирует отображение по порядку памяти.

**-v**
Отображение сведений о версии. Эти сведения взяты из раздела ресурсов каждого модуля.

*LoaderEntryAddress*
Указывает адрес записи загрузчика для модуля. Если этот параметр включен, отладчик отображает только этот конкретный модуль.

**-h**
Отображает текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Exts.dll

# Замечания

Список модулей включает все точки входа в каждый модуль.

Расширение **DLLS** работает только в динамической отладке (не с анализом аварийного дампа).

В режиме ядра это расширение отображает модули для текущего контекста процесса. Вы не можете использовать **библиотеки** !dll вместе с системным процессом или процессом простоя.

В следующих примерах показано, как использовать **расширение !dlls** .

dbgcmd

```
kd> !dlls -c 77f60000
Dump dll containing 0x77f60000:

0x00091f38: E:\WINDOWS\System32\ntdll.dll
      Base   0x77f60000  EntryPoint  0x00000000  Size        0x00097000
      Flags  0x00004004  LoadCount   0x0000ffff  TlsIndex    0x00000000
             LDRP_IMAGE_DLL
             LDRP_ENTRY_PROCESSED

kd> !dlls -a 91ec0

0x00091ec0: E:\WINDOWS\system32\winmine.exe
      Base   0x01000000  EntryPoint  0x01003e2e  Size        0x00020000
      Flags  0x00005000  LoadCount   0x0000ffff  TlsIndex    0x00000000
             LDRP_LOAD_IN_PROGRESS
             LDRP_ENTRY_PROCESSED

File Type: EXECUTABLE IMAGE
FILE HEADER VALUES
     14C machine (i386)
       3 number of sections
```

```
3A98E856 time date stamp Sun Feb 25 03:11:18 2001

       0 file pointer to symbol table
       0 number of symbols
      E0 size of optional header
     10F characteristics
            Relocations stripped
            Executable
            Line numbers stripped
            Symbols stripped
            32 bit word machine

OPTIONAL HEADER VALUES
     10B magic #
    7.00 linker version
    3A00 size of code
   19E00 size of initialized data
       0 size of uninitialized data
    3E2E address of entry point
    1000 base of code
         ----- new -----
01000000 image base
    1000 section alignment
     200 file alignment
       2 subsystem (Windows GUI)
    5.01 operating system version
    5.01 image version
    4.00 subsystem version
   20000 size of image
     400 size of headers
   21970 checksum
00040000 size of stack reserve
00001000 size of stack commit
00100000 size of heap reserve
00001000 size of heap commit
01000100 Opt Hdr
       0 [       0] address [size] of Export Directory
    40B4 [      B4] address [size] of Import Directory
    6000 [   19170] address [size] of Resource Directory
       0 [       0] address [size] of Exception Directory
       0 [       0] address [size] of Security Directory
       0 [       0] address [size] of Base Relocation Directory
    11B0 [      1C] address [size] of Debug Directory
       0 [       0] address [size] of Description Directory
       0 [       0] address [size] of Special Directory
       0 [       0] address [size] of Thread Storage Directory
       0 [       0] address [size] of Load Configuration Directory
     258 [      A8] address [size] of Bound Import Directory
    1000 [     1B0] address [size] of Import Address Table Directory
       0 [       0] address [size] of Reserved Directory
       0 [       0] address [size] of Reserved Directory
       0 [       0] address [size] of Reserved Directory


SECTION HEADER #1
```

```
   .text name
    3992 virtual size
    1000 virtual address
    3A00 size of raw data
     400 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
60000020 flags
         Code
         (no align specified)
         Execute Read


Debug Directories(1)
        Type       Size     Address  Pointer

        cv           1c        13d0      7d0    Format: NB10, 3a98e856, 1,
winmi
ne.pdb

SECTION HEADER #2
   .data name
     BB8 virtual size
    5000 virtual address
     200 size of raw data
    3E00 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
C0000040 flags
         Initialized Data
         (no align specified)
         Read Write

SECTION HEADER #3
   .rsrc name
   19170 virtual size
    6000 virtual address
   19200 size of raw data
    4000 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
40000040 flags
         Initialized Data
         (no align specified)
         Read Only
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !dml_proc

Статья • 26.10.2023

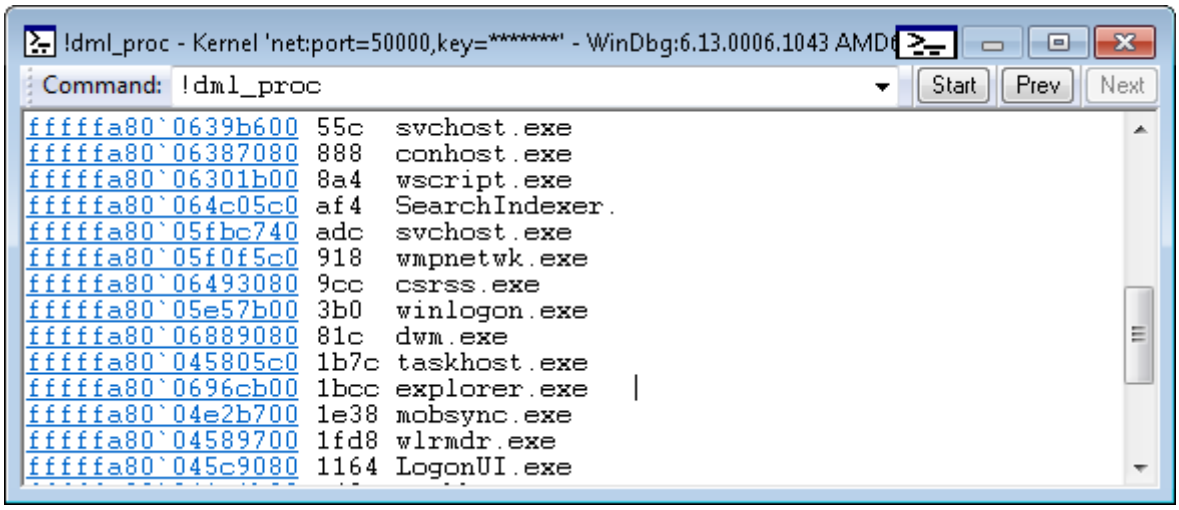
Расширение **!dml_proc** отображает список процессов и ссылки для получения более подробных сведений о процессах.

dbgcmd

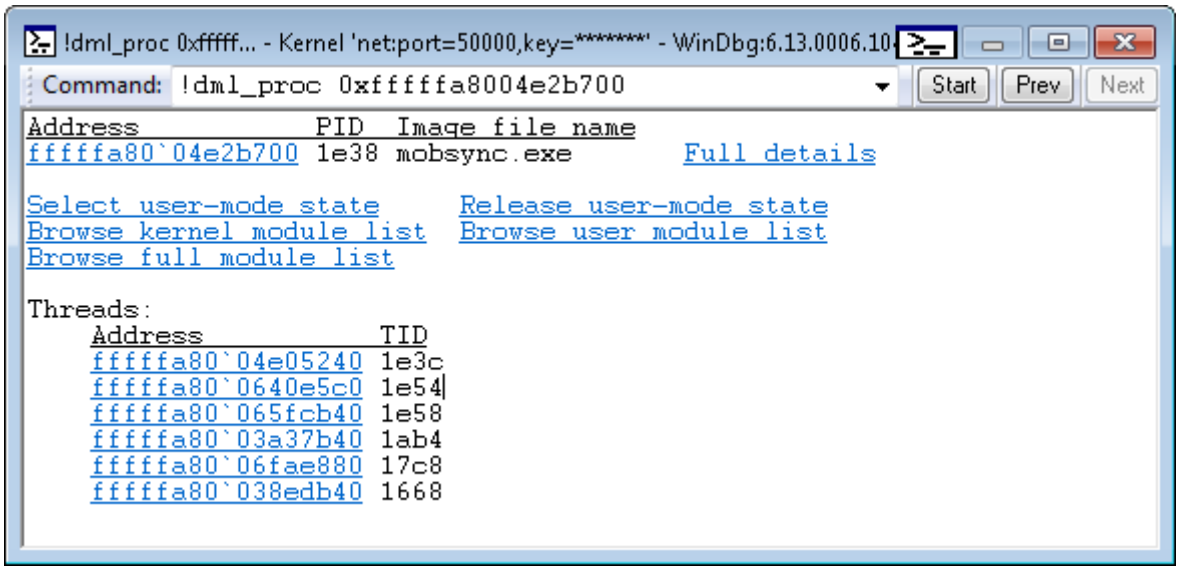
```
!dml_proc
```

## Комментарии

На следующем рисунке показана часть выходных данных, отображаемая **!dml_proc**.




В предыдущих выходных данных адреса процесса — это ссылки, которые можно щелкнуть, чтобы просмотреть более подробные сведения. Например, если щелкнуть **fffffa80'04e2b700** (адрес для mobsync.exe), вы увидите подробные сведения о процессе mobsync.exe, как показано на следующем рисунке.




Предыдущие выходные данные, описывающие отдельный процесс, содержат ссылки, которые можно щелкнуть для более подробного изучения процесса и его потоков.

## См. также

Команды языка разметки отладчика

# !dumpfa

Статья • 05.04.2024

Расширение **!dumpfa** отображает содержимое записи анализа сбоев.

```
dbgcmd
```

```
!dumpfa Address
```

## Параметры

*Адрес*
Указывает адрес отображаемой записи анализа сбоев.

## DLL-библиотеки

Ext.dll

## Замечания

**Расширение .dumpfa** полезно только для отладки **расширения !analyze**, как показано в следующем примере.

```
dbgcmd
```

```
0:000> !dumpfa 0x00a34140
DataUsed 3b0
Type =        DEBUG_FLR_MARKER_BUCKET 00010016 - Size = 9
Type =          DEBUG_FLR_MARKER_FILE 0001000d - Size = 16
Type =      DEBUG_FLR_SYSXML_LOCALEID 00004200 - Size = 4
Type =      DEBUG_FLR_SYSXML_CHECKSUM 00004201 - Size = 4
Type =         DEBUG_FLR_READ_ADDRESS 0000000e - Size = 8
Type =          DEBUG_FLR_FAULTING_IP 80000000 - Size = 8
Type =     DEBUG_FLR_MM_INTERNAL_CODE 00001004 - Size = 8
Type = DEBUG_FLR_CPU_MICROCODE_VERSION 0000301f - Size = 28
Type = DEBUG_FLR_CUSTOMER_CRASH_COUNT 0000300b - Size = 8
Type =    DEBUG_FLR_DEFAULT_BUCKET_ID 00010008 - Size = 12
Type =         DEBUG_FLR_BUGCHECK_STR 00000600 - Size = 5
Type = DEBUG_FLR_LAST_CONTROL_TRANSFER 0000000a - Size = 18
Type =           DEBUG_FLR_TRAP_FRAME c0000002 - Size = 8
Type =           DEBUG_FLR_STACK_TEXT 00010005 - Size = 1fb
Type =        DEBUG_FLR_STACK_COMMAND 00010004 - Size = 17
Type =        DEBUG_FLR_OS_BUILD_NAME 0000301e - Size = 9
Type =          DEBUG_FLR_MODULE_NAME 00010006 - Size = 8
```

```
Type =           DEBUG_FLR_IMAGE_NAME 00010001 - Size = c
Type =      DEBUG_FLR_IMAGE_TIMESTAMP 80000002 - Size = 8
```

Для отладки расширения !asd можно также использовать **расширение !asd**.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !envvar

Статья • 05.04.2024

Расширение **!envvar** отображает значение указанной переменной среды.

```
dbgcmd
```

```
!envvar Variable
```

## Параметры

*Переменная*
Указывает переменную среды, значение которой отображается. *Переменная* не учитывает регистр.

## DLL-библиотеки

Exts.dll

## Дополнительная информация

Дополнительные сведения о переменных среды см. в документации по переменным среды и пакету SDK для Microsoft Windows.

## Замечания

**Расширение !envvar** работает как в пользовательском режиме, так и в режиме ядра. Однако в режиме ядра при установке потока простоя в качестве текущего процесса указатель на блок среды обработки (PEB) имеет значение **NULL**, поэтому он завершается ошибкой. В режиме **ядра расширение !envvar** отображает переменные среды на целевом компьютере, как показано в следующем примере.

```
dbgcmd
```

```
0:000> !envvar _nt_symbol_path
        _nt_symbol_path =
srv*C:\mysyms*https://msdl.microsoft.com/download/symbols
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ошибка

Статья • 08.07.2024

Расширение **!error** декодирует и отображает сведения о значении ошибки.

```
dbgcmd
```

```
!error Value [Flags]
```

## Параметры

*Value*
Указывает один из следующих кодов ошибок:

- Win32
- Winsock
- NTSTATUS
- NetAPI

*Флаги*
Если *флаги* равны 1, код ошибки считывается как код NTSTATUS.

## DLL-библиотеки

Ext.dll

## Замечания

В следующем примере показано, как использовать **!error**.

```
dbgcmd
```

```
0:000> !error 2
Error code: (Win32) 0x2 (2) - The system cannot find the file specified.
0:000> !error 2 1
Error code: (NTSTATUS) 0x2 - STATUS_WAIT_2
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !exchain

Статья • 05.04.2024

Расширение **!exchain** отображает текущую цепочку обработчика исключений.

```
dbgcmd
```

```
!exchain [Options]
```

# Параметры

*Параметры* одного из следующих значений:

**/c**
Отображает сведения, относящиеся к отладке исключения C++ **/** при обнаружении такого исключения.

**/C**
Отображает сведения, относящиеся к отладке исключения C++**/**, даже если такое исключение не обнаружено.

**/f**
Отображает сведения, полученные путем обхода таблиц функций CRT, даже если обработчик исключений CRT не обнаружен.

# DLL-библиотеки

Ext.dll

**Расширение !exchain** доступно только для целевого компьютера на основе x86.

# Замечания

Расширение **!exchain** отображает список обработчиков исключений для текущего потока.

Список начинается с первого обработчика в цепочке (тот, который получает первую возможность обрабатывать исключение) и продолжается до конца. В следующем примере показано это расширение.

```
dbgcmd
```

```
0:000> !exchain
0012fea8: Prymes!_except_handler3+0 (00407604)
  CRT scope  0, filter: Prymes!dzExcepError+e6 (00401576)
                func:   Prymes!dzExcepError+ec (0040157c)
0012ffb0: Prymes!_except_handler3+0 (00407604)
  CRT scope  0, filter: Prymes!mainCRTStartup+f8 (004021b8)
                func:   Prymes!mainCRTStartup+113 (004021d3)
0012ffe0: KERNEL32!GetThreadContext+1c (77ea1856)
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Exr

Статья • 05.04.2024

Команда **расширения !exr** устарела. **Используйте вместо этого команду EXR (Отображаемая запись исключений**).

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !findxmldata

Статья • 08.07.2024

Расширение **!findxmldata** извлекает XML-данные из CAB-файла, содержащего файл небольшого дампа памяти в режиме ядра.

dbgcmd

```
!findxmldata [ -d DeviceName | -h HwId ]
!findxmldata -r Driver
!findxmldata -chksum [ -z CabFile ]
!findxmldata -v
```

## Параметры

**-d** *DeviceName*
Отображает все устройства, *имя которого содержит строку, указанную DeviceName* .

**-h** *HwId*
Отображает все устройства, идентификаторы оборудования которых содержат строку, *указанную HwId* . Если используется как -d**, так** и **-h** отладчик отображает только те устройства, которые удовлетворяют обоим совпадениям.

**-r** *Водитель*
Отображает сведения о драйвере, который *указывает параметр Driver* , включая все устройства, использующие этот драйвер.

**-chksum**
Отображает контрольную сумму XML-файла.

**-z** *CabFile*
Позволяет выполнять контрольную сумму для CAB-файла, указанного параметром CabFile, а не в файле Sysdata.xml по умолчанию.

**-v**
Отображает сведения о версии системы.

## DLL-библиотеки

Ext.dll

# Дополнительная информация

**Расширение !findxmldata** работает только в файле небольшого дампа памяти в режиме ядра, который хранится в CAB-файле.

Дополнительные сведения о том, как поместить файлы дампа в CAB-файлы, см. в разделе **.dumpcab (создание CAB-файла дампа).** Дополнительные сведения о отладке файла дампа в режиме ядра, включая файлы дампа, хранящиеся в CAB-файлах, см. в разделе "Анализ файла дампа в режиме ядра".

# Замечания

**Расширение !findxmldata** извлекает данные из файла Sysdata.xml, который хранится в CAB-файле, который содержит файл дампа памяти в режиме ядра.

Если вы не используете никаких параметров, расширение отображает все устройства.

В следующих примерах показано, как использовать **!findxmldata**.

dbgcmd

```
kd> !findxmldata -v
SYSTEM Info:
OSVER: 5.1.2600 2.0
OSLANGUAGE: 2052
OSNAME: Microsoft Windows XP Home Edition
kd> !findxmldata -d MIDI
Node DEVICE
 DESCRIPTION    : MPU-401 Compatible MIDI Device
        HARDWAREID     : ACPI\PNPB006
        SERVICE        : ms_mpu401
        DRIVER         : msmpu401.sys

kd> !findxmldata -r msmpu
Node DRIVER
 FILENAME       : msmpu401.sys
        FILESIZE       : 2944
        CREATIONDATE   : 05-06-2005 09:18:34
        VERSION        : 5.1.2600.0
        MANUFACTURER   : Microsoft Corporation
        PRODUCTNAME    : Microsoft« Windows« Operating System
Node DEVICE
        DESCRIPTION    : MPU-401 Compatible MIDI Device
 HARDWAREID     : ACPI\PNPB006
        SERVICE        : ms_mpu401
        DRIVER         : msmpu401.sys

kd> !findxmldata -h PCI\VEN_8086&DEV_24C3&SUBSYS_24C28086
```

```
Node DEVICE
 DESCRIPTION    : Intel(R) 82801DB/DBM SMBus Controller - 24C3
        HARDWAREID     : PCI\VEN_8086&DEV_24C3&SUBSYS_24C28086&REV_01
kd> !findxmldata -h USB\ROOT_HUB&VID8086&PID24C4&REV0001
Node DEVICE
        DESCRIPTION    : USB Root Hub
 HARDWAREID     : USB\ROOT_HUB&VID8086&PID24C4&REV0001
        SERVICE        : usbhub
        DRIVER         : usbhub.sys

kd> !findxmldata -h ACPI\PNPB006
Node DEVICE
        DESCRIPTION    : MPU-401 Compatible MIDI Device
 HARDWAREID     : ACPI\PNPB006
        SERVICE        : ms_mpu401
        DRIVER         : msmpu401.sys
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !for_each_frame

Статья • 05.04.2024

**Расширение !for_each_frame** выполняет команду отладчика один раз для каждого кадра в стеке текущего потока.

dbgcmd

```
!for_each_frame ["CommandString"]
!for_each_frame -?
```

## Параметры

*CommandString*
Указывает команды отладчика для выполнения по одному разу для каждого кадра. Если *CommandString* содержит несколько команд, их необходимо разделить точкой с запятой и заключить *CommandString* в кавычки. Если включить несколько команд, отдельные команды в *CommandString* не могут содержать кавычки. Если вы хотите ссылаться на индекс текущего кадра в *CommandString*, используйте псевдорегистр @$frame.

**-?**
Отображает текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Ext.dll

## Дополнительная информация

Дополнительные сведения о локальном контексте см. в разделе "Изменение контекстов".

## Замечания

Если аргументы не указаны, **расширение !for_each_frame** отображает список всех кадров и их индексов кадров. Чтобы получить более подробный список всех кадров, используйте **команду k (Display Stack Backtrace).**

Команда **k** проходит до 256 кадров. Для каждого перечисленного кадра этот кадр временно становится текущим локальным контекстом (аналогично **команде .frame (Set Local Context).** После установки *контекста выполняется CommandString* . После того как все кадры были использованы, локальный контекст сбрасывается на значение, которое оно было до использования **расширения !for_each_frame** .

Если включить *CommandString*, отладчик отображает кадр и его индекс перед выполнением команды для этого кадра.

Следующая команда отображает все локальные переменные для текущего стека.

dbgcmd

```
!for_each_frame !for_each_local dt @#Local
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !for_each_function

Статья • 05.04.2024

**Расширение !for_each_function** выполняет команду отладчика для каждой функции в указанном модуле, имя которого соответствует указанному шаблону.

dbgcmd

```
!for_each_function -m:ModuleName -p:Pattern -c:CommandString
!for_each_function -?
```

## Параметры

-m:*ModuleName*
Указывает имя модуля. Обычно это имя файла без расширения имени файла. В некоторых случаях имя модуля значительно отличается от имени файла.

-p:*Pattern*
Указывает соответствующий шаблон.

-c:*CommandString*
Указывает команду отладчика, выполняемую для каждой функции в указанном модуле, которая соответствует шаблону.

В CommandString *можно использовать следующие псевдонимы*.

Развернуть таблицу

| Псевдоним | Тип данных | Значение |
|---|---|---|
| @#SymbolName | строка | Имя символа. |
| @#SymbolAddress | ULONG64 | Адрес символа. |
| @#ModName | строка | Имя модуля. |
| @#FunctionName | строка | имя функции. |

-?
Отображает справку по этому расширению.

## DLL-библиотеки

# Замечания

В следующем примере показано, как перечислить все имена функций в модуле PCI, соответствующие шаблону *read*.

dbgcmd

```
1: kd> !for_each_function -m:pci -p:*read* -c:.echo @#FunctionName

PciReadDeviceConfig
PciReadDeviceSpace
PciReadSlotIdData
PciExternalReadDeviceConfig
PiRegStateReadStackCreationSettingsFromKey
VmProxyReadDevicePathsFromRegistry
PpRegStateReadCreateClassCreationSettings
ExpressRootPortReadConfigSpace
PciReadRomImage
PciDevice_ReadConfig
PciReadDeviceConfigEx
PciReadSlotConfig
```

Если псевдоним не предшествует и за ним следует пустое пространство, необходимо поместить псевдоним в **маркер интерпретатора ${}Alias.**

В следующем примере показано, как перечислить все символы во всех модулях, имена функций которых соответствуют шаблону *CreateFile*. Псевдоним @#ModuleName не предшествует пустому пробелу. Поэтому он помещается внутри **маркера интерпретатора ${}Alias.**

**Примечание. Не** путайте псевдоним @#ModuleName с псевдонимом @#ModName. Псевдоним @#ModuleName принадлежит расширению !for_each_module**, а псевдоним @#ModName принадлежит расширению !for_each_function.**

dbgcmd

```
1: kd> !for_each_module !for_each_function -m:${@#ModuleName} -
p:*CreateFile* -c:.echo @#SymbolName
nt!BiCreateFileDeviceElement
nt!NtCreateFile
...
Wdf01000!FxFileObject::_CreateFileObject
fltmgr!FltCreateFileEx2$fin$0
fltmgr!FltCreateFileEx2
...
Ntfs!TxfIoCreateFile
```

```
Ntfs!NtfsCreateFileLock
...
MpFilter!MpTxfpCreateFileEntryUnsafe
mrxsmb10!MRxSmbFinishLongNameCreateFile
...
srv!SrvIoCreateFile
```

Вы можете поместить последовательность команд в файл команд и использовать **$$>< (run Script File)** для выполнения этих команд для каждой функции, которая соответствует шаблону. Предположим, что файл с именем Commands.txt содержит следующие команды:

dbgcmd

```
.echo
.echo @#FunctionName
u @#SymbolAddress L1
```

В следующем примере команды в текстовом файле Commands.text выполняются для каждой функции в модуле PCI, которая соответствует шаблону *read*.

dbgcmd

```
1: kd> !for_each_function -m:pci -p:*read* -c:$$><Commands.txt

PciReadDeviceConfig
pci!PciReadDeviceConfig [d:\wmm1\drivers\busdrv\pci\config.c @ 349]:
fffff880`00f7b798 48895c2408      mov     qword ptr [rsp+8],rbx

PciReadDeviceSpace
pci!PciReadDeviceSpace [d:\wmm1\drivers\busdrv\pci\config.c @ 1621]:
fffff880`00f7c044 48895c2408      mov     qword ptr [rsp+8],rbx

...
```

# См. также

**!for_each_module**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !for_each_local

Статья • 05.04.2024

**Расширение !for_each_local** выполняет команду отладчика один раз для каждой локальной переменной в текущем кадре.

```
dbgcmd
```

```
!for_each_local ["CommandString"]
!for_each_local -?
```

# Параметры

*CommandString*
Указывает команды отладчика для выполнения по одному разу для каждой локальной переменной в текущем кадре стека. Если *CommandString* содержит несколько команд, их необходимо разделить точкой с запятой и заключить *CommandString* в кавычки. Если включить несколько команд, отдельные команды в *CommandString* не могут содержать кавычки.

В CommandString *или в* любом скрипте, который выполняются команды в *CommandString*, можно использовать **псевдоним @#Local**. Этот псевдоним заменяется именем локальной переменной. Эта замена возникает перед *выполнением CommandString* и перед выполнением любого другого синтаксического анализа. Этот псевдоним учитывает регистр, и необходимо добавить пробел перед ним и добавить пробел после него, даже если псевдоним заключен в скобки. При использовании синтаксиса выражений C++ необходимо ссылаться на этот псевдоним как **@@( @#Local )**.

Этот псевдоним доступен только в течение времени существования вызова **!for_each_local**. Не путайте этот псевдоним с псевдорегистрацией, псевдонимами фиксированного имени или псевдонимами с именем пользователя.

**-?**
Отображает текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Ext.dll

# Дополнительная информация

Дополнительные сведения о том, как отображать и изменять локальные переменные и описание других команд, связанных с памятью, см. в разделе "Чтение и запись памяти".

# Замечания

Если аргументы не указаны, **расширение !for_each_local** перечисляет локальные переменные. Дополнительные сведения о локальных переменных используйте **команду dv (Display Local Variables).**

Если включить подробные выходные данные отладчика, отображение включает общее количество локальных переменных при вызове расширения и каждый раз, когда *commandString* выполняется для локальной переменной, эта переменная и текст *CommandString* будут эхо.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !for_each_module

Статья • 05.04.2024

**Расширение !for_each_module** выполняет команду отладчика один раз для каждого загруженного модуля.

dbgcmd

```
!for_each_module ["CommandString"]
!for_each_module -?
```

## <Параметры

*CommandString*
Указывает команды отладчика для выполнения по одному разу для каждого модуля в списке модулей отладчика. Если *CommandString* содержит несколько команд, их необходимо разделить точкой с запятой и заключить *CommandString* в кавычки. Если включить несколько команд, отдельные команды в *CommandString* не могут содержать кавычки.

Вы можете использовать следующие псевдонимы в *CommandString* или в любом скрипте, который выполняет команды в *CommandString* .

Развернуть таблицу

| Псевдоним | Тип данных | Значение |
|---|---|---|
| @#FileVersion | строка | Версия файла модуля. |
| @#ProductVersion | строка | Версия продукта модуля. |
| @#ModuleIndex | ULONG | Номер модуля. Модули перечисляются последовательно, начиная с нуля. |
| @#ModuleName | строка | Имя модуля. Обычно это имя файла без расширения имени файла. В некоторых ситуациях имя модуля значительно отличается от имени файла. |

| Псевдоним | Тип данных | Значение |
|---|---|---|
| @#ImageName | строка | Имя исполняемого файла, включая расширение имени файла. Как правило, полный путь включен в пользовательский режим, но не в режиме ядра. |
| @#LoadedImageName | строка | Если символы Microsoft CodeView отсутствуют, этот псевдоним совпадает с именем изображения. |
| @#MappedImageName | строка | В большинстве случаев этот псевдоним имеет значение **NULL**. Если отладчик сопоставляет файл изображения (например, во время отладки minidump), этот псевдоним — это имя сопоставленного изображения. |
| @#SymbolFileName | строка | Путь и имя файла символов. Если вы не загрузили символы, этот псевдоним — это имя исполняемого файла. |
| @#ModuleNameSize | ULONG | Длина строки имени модуля, а также одна. |
| @#ImageNameSize | ULONG | Длина строки имени изображения, а также одна. |
| @#LoadedImageNameSize | ULONG | Длина строки загруженного имени изображения плюс одна. |
| @#MappedImageNameSize | ULONG | Длина строки сопоставленного имени изображения плюс одна. |
| @#SymbolFileNameSize | ULONG | Длина строки имени файла символов плюс одна. |
| @#Base | ULONG64 | Адрес начала изображения. |
| @#Size | ULONG | Размер изображения в байтах. |

| Псевдоним | Тип данных | Значение |
|---|---|---|
| @#End | ULONG64 | Адрес конца изображения. |
| @#TimeDateStamp | ULONG | Метка времени и даты изображения. Если вы хотите развернуть это время и метку даты на удобочитаемую дату, используйте **команду .форматов (показать форматы чисел).** |
| | | |
| @#Checksum | ULONG | Проверка сумм модуля. |
| @#Flags | ULONG | Флаги модуля. Список значений DEBUG_MODULE_*Xxx* см. в разделе Dbgeng.h. |
| @#SymbolType | USHORT | Тип символа. Список значений DEBUG_SYMTYPE_*Xxx* см. в разделе Dbgeng.h. |

Эти псевдонимы заменяются перед *выполнением CommandString* для каждого модуля и перед выполнением любого другого синтаксического анализа. Эти псевдонимы чувствительны к регистру. Необходимо добавить пробел перед псевдонимом и пробелом после него, даже если псевдоним заключен в скобки. При использовании синтаксиса выражений C++ необходимо ссылаться на эти псевдонимы как @@( @#*alias*).

Эти псевдонимы доступны только в течение времени существования вызова **!for_each_module**. Не путайте их с псевдорегистрацией, псевдонимами фиксированного имени или псевдонимами с именем пользователя.

-?
Отображает текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Ext.dll

# Дополнительная информация

Дополнительные сведения об определении и использовании псевдонимов в качестве сочетаний клавиш для ввода символьных строк (включая использование маркера ${ } см. в разделе "Использование псевдонимов".

# Замечания

Если аргументы не указаны, **расширение !for_each_module** отображает общие сведения о загруженных модулях. Эти сведения похожи на сведения, отображаемые в следующей команде.

dbgcmd

```
!for_each_module .echo @#ModuleIndex : @#Base @#End @#ModuleName @#ImageName
@#LoadedImageName
```

Дополнительные сведения о загруженных и выгрузочных модулях используйте **команду lm (список загруженных модулей).**

Если включить подробные выходные данные отладчика, отладчик отображает общее количество загруженных и выгрузленных модулей при вызове расширения, а отладчик отображает подробные сведения о каждом модуле (включая значения каждого доступного псевдонима) перед *выполнением CommandString* для этого модуля.

В следующих примерах показано, как использовать **расширение !for_each_module** . Следующие команды отображают глобальные флаги отладки.

dbgcmd

```
!for_each_module x ${@#ModuleName}!*Debug*Flag*
!for_each_module x ${@#ModuleName}!g*Debug*
```

Следующая команда проверка для повреждения двоичных файлов в каждом загруженном модуле с помощью расширения !chkimg:

dbgcmd

```
!for_each_module !chkimg @#ModuleName
```

Следующая команда ищет шаблон "MZ" в каждом загруженном изображении.

dbgcmd

```
!for_each_module s-a @#Base @#End "MZ"
```

В следующем примере показано использование @#FileVersion и @#ProductVersion для каждого имени модуля:

dbgcmd

```
0:000> !for_each_module .echo @#ModuleName fver = @#FileVersion pver =
@#ProductVersion
USER32 fver = 6.0.6000.16438 (vista_gdr.070214-1610) pver = 6.0.6000.16438
kernel32 fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
ntdll fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
notepad fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
WINSPOOL fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
COMCTL32 fver = 6.10 (vista_rtm.061101-2205) pver = 6.0.6000.16386
SHLWAPI fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
msvcrt fver = 7.0.6000.16386 (vista_rtm.061101-2205) pver = 7.0.6000.16386
GDI32 fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
RPCRT4 fver = 6.0.6000.16525 (vista_gdr.070716-1600) pver = 6.0.6000.16525
SHELL32 fver = 6.0.6000.16513 (vista_gdr.070626-1505) pver = 6.0.6000.16513
ole32 fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
ADVAPI32 fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
COMDLG32 fver = 6.0.6000.16386 (vista_rtm.061101-2205) pver = 6.0.6000.16386
```


# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !for_each_register

Статья • 05.04.2024

**Расширение !for_each_register** выполняет указанную команду для каждого регистра.

```
dbgcmd
```

```
!for_each_register -c:CommandString
!for_each_register -?
```

## Параметры

**-c:***CommandString*
Указывает команду, выполняемую для каждого регистра. Псевдонимы @#RegisterName и @#RegisterValue действительны во время выполнения команды.

**-?**
Отображает справку по расширению **!for_each_register** .

## DLL-библиотеки

Ext.dll

## Примеры

В этом примере перечислены имена каждого регистра.

```
dbgcmd
```

```
0:000> !for_each_register -c:.echo @#RegisterName
rax
rcx
rdx
rbx
...
```

В этом примере выполняется !address **для каждого значения регистра**.

```
dbgcmd
```

```
0:000> !for_each_register -c:!address ${@#RegisterValue}
...
Usage:                  Stack
Base Address:           00000008`a568f000
End Address:            00000008`a56a0000
Region Size:            00000000`00011000
State:                  00001000    MEM_COMMIT
Protect:                00000004    PAGE_READWRITE
Type:                   00020000    MEM_PRIVATE
Allocation Base:        00000008`a5620000
Allocation Protect:     00000004    PAGE_READWRITE
More info:              ~0k
...
```

# Замечания

Если псевдоним является аргументом расширения отладчика (например, !address), **используйте маркер интерпретатора псевдонима ${} (Псевдоним интерпретатора), чтобы псевдоним был правильно разрешен.**

Дополнительные сведения об определении и использовании псевдонимов в качестве сочетаний клавиш для ввода символьных строк (включая использование маркера ${}) см. в разделе "Использование псевдонимов".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gflag

Статья • 08.07.2024

Расширение **!gflag** задает или отображает глобальные флаги.

```
dbgcmd
```

```
!gflag [+|-] Value
!gflag {+|-} Abbreviation
!gflag -?
!gflag
```

## Параметры

*Value*
Указывает 32-разрядное шестнадцатеричное число. Если вы не используете знак плюса (+) или знак минуса (-), это число становится новым значением битового поля глобального флага. Если вы добавляете знак плюса (+) перед этим числом, число указывает один или несколько глобальных битов флагов, для которых задано значение 1. Если вы добавляете знак минуса (-) перед этим числом, число указывает один или несколько битов глобального флага, чтобы задать значение нулю.

*Сокращение*
Задает один глобальный флаг. *Аббревиатура* — это трехбуквенное сокращение для глобального флага, которое имеет значение 1 (+) или равно нулю (-).

**-?**
Отображает текст справки для этого расширения, включая список сокращенных глобальных флагов в окне команды отладчика.

## DLL-библиотеки

Exts.dll

## Дополнительная информация

Эти флаги также можно задать с помощью программы глобальных флагов (Gflags.exe).

# Замечания

Если аргументы не указаны, **расширение !gflag** отображает текущие параметры глобального флага.

В следующей таблице содержатся сокращенные значения, которые можно использовать для *параметра аббревиации* .

Развернуть таблицу

| Значение | Имя | Описание |
|---|---|---|
| 0x00000001 | "soe" | Остановите исключение. |
| 0x00000002 | Sls | Отображение оснастки загрузчика. |
| 0x00000004 | "dic" | Отладка начальной команды. |
| 0x00000008 | "shg" | Остановите, если ГРАФИЧЕСКИй интерфейс перестает отвечать (т. е. зависает). |
| 0x00000010 | "htc" | Включите проверку хвоста кучи. |
| 0x00000020 | "hfc" | Включите бесплатную проверку кучи. |
| 0x00000040 | "hpc" | Включите проверку параметров кучи. |
| 0x00000080 | "hvc" | Включите проверку кучи при вызове. |
| 0x00000100 | "ptc" | Включите проверку хвоста пула. |
| 0x00000200 | "pfc" | Включите бесплатную проверку пула. |
| 0x00000400 | "ptg" | Включение тегов пула. |
| 0x00000800 | "htg" | Включите тег кучи. |
| 0x00001000 | "ust" | Создайте базу данных трассировки стека в пользовательском режиме. |

| Значение | Имя | Описание |
| --- | --- | --- |
| 0x00002000 | "kst" | Создайте базу данных трассировки стека в режиме ядра. |
| 0x00004000 | "otl" | Сохраняйте список объектов для каждого типа. |
| 0x00008000 | "htd" | Включение тегов кучи по библиотеке DLL. |
| 0x00010000 | "idp" | Не используется. |
| 0x00020000 | "d32" | Включите отладку подсистемы Microsoft Win32. |
| 0x00040000 | Ksl | Включите загрузку символов отладчика ядра. |
| 0x00080000 | "dps" | Отключите разбиение по страницам стека ядра. |
| 0x00100000 | "scb" | Включите критически важные разрывы системы. |
| 0x00200000 | "dhc" | Отключите кучу на свободном режиме. |
| 0x00400000 | "эхо" | Включите исключение закрытия. |
| 0x00800000 | "угорь" | Включите ведение журнала исключений. |
| 0x01000000 | "eot" | Включение тегов типа дескриптора объекта. |
| 0x02000000 | "hpa" | Поместите выделения кучи в конец страниц. |
| 0x04000000 | "dwl" | Отладка WINLOGON. |
| 0x08000000 | "ddp" | Отключите выходные данные DbgPrint и **KdPrint в режиме** ядра. |
| 0x10000000 | NULL | Не используется. |
| 0x20000000 | "sue" | Остановка на необработанных |

| Значение | Имя | Описание |
| --- | --- | --- |
| | | исключениях в пользовательском режиме |
| 0x40000000 | NULL | Не используется. |
| 0x80000000 | "dpd" | Отключите проверку защищенной библиотеки DLL. |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gle

Статья • 05.04.2024

Расширение **!gle** отображает последнее значение ошибки для текущего потока.

```
dbgcmd
```

```
!gle [-all]
```

## Параметры

**-Все**
Отображает последнюю ошибку для каждого потока пользовательского режима в целевой системе. Если этот параметр не указан в пользовательском режиме, отладчик отображает последнюю ошибку для текущего потока. Если этот параметр не указан в режиме ядра, отладчик отображает последнюю ошибку для потока, указанного текущим контекстом регистрации.

## DLL-библиотеки

Exts.dll

## Дополнительная информация

Дополнительные сведения о подпрограмме **GetLastError** см. в документации по пакету SDK для Windows.

## Замечания

Расширение **!gle** отображает значение **GetLastError** и пытается декодировать это значение.

В режиме ядра расширение !gle **работает только в том случае,** если отладчик может считывать блок среды потока (TEB).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Gs

Статья • 05.04.2024

Расширение !gs **анализирует переполнение** стека /GS.

dbgcmd

```
!gs
```

## DLL-библиотеки

Ext.dll

## Замечания

Расширение !gs**помогает отладить переполнение** буфера. Запустите **!gs**при возникновении ошибки STATUS_STACK_BUFFER_OVERRUN, как показано в следующем примере.

dbgcmd

```
0:000> !gs
Corruption occurred in mshtml!CDoc::OnPaint or one of its callers
Real canary not found at 0x74866010
Canary at gsfailure frame 0x292ea4e7
Corrupted canary 0x0013e2c8: 0x00000000
Corrupted cookie value too generic, skipping init bit-flip check
a caller of mshtml!CDoc::OnPaint has corrupted the EBP from 0x0013e254 to
0x0013
e234
check callers (without canary) of mshtml!CDoc::OnPaint for 0x1 bytes of
overflow

The canary doesn't look corrupted. Not sure how we got here
EBP/ESP check skipped: No saved EBP in exception context
Function mshtml!CDoc::OnPaint:
  00000000 - 00000004 this                      CDoc*
 0013de40 - 0013e180 rd
CDoc::OnPaint::__l39::REGION_DAT
A
  0013e180 - 0013e18c Lock                      CDoc::CLock
  0013e18c - 0013e224 DI                        CFormDrawInfo
  0013e23c - 0013e240 hwndInplace               HWND__*
  0013e240 - 0013e244 prc                       tagRECT*
  0013e248 - 0013e250 ptBefore                  tagPOINT
  0013e250 - 0013e254 fViewIsReady              int
```

```
  0013e250 - 0013e254 fHtPalette                int
  0013e254 - 0013e258 fNoPaint                  int
  0013e258 - 0013e260 ptAfter                   tagPOINT
  0013e260 - 0013e264 c                         int
  0013e264 - 0013e268 hrgn                      HRGN__*
  0013e268 - 0013e2a8 ps                        tagPAINTSTRUCT
Candidate buffer : ps 0013e268 to 0013e2a7
  0013e268 ea 04 01 a7 00 00 00 00-10 01 00 00 f3 00 00 00 ................
  0013e278 ed 03 00 00 44 02 00 00-84 e5 13 00 f4 e2 13 00 ....D...........
  ...
  0013e2ac 38 20 01 03 10 e3 13 00-68 6b e6 01 d0 e6 03 00 8 ......hk......
 0013e2bc 80 fa 03 00 0d 00 00 00-10 08 19 00 00 00 00 00 ................
0:000>
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Обрабатывать

Статья • 05.04.2024

Расширение **!handle отображает сведения об дескрипторе или дескрипторе** , который принадлежит одной или всем процессам в целевой системе.

Режим пользователя

```
dbgcmd
```

```
!handle [Handle [UMFlags [TypeName]]]
!handle -?
```

Режим ядра

```
dbgcmd
```

```
!handle [Handle [KMFlags [Process [TypeName]]]]
```

# Параметры

*Дескриптор*
Задает индекс отображаемого дескриптора. Если *дескриптор равен -1 или если этот параметр опущен, отладчик отображает данные для всех дескрипторов* , связанных с текущим процессом. Если *дескриптор* равен 0, отладчик отображает данные для всех дескрипторов.

*UMFlags*
(Только в пользовательском режиме) Указывает, что должен содержать дисплей. Этот параметр может быть суммой любого из следующих битовых значений. (Значение по умолчанию — 0x1.)

Bit 0 (0x1)
Отображает сведения о типе дескриптора.

Бит 1 (0x2)
Отображает основные сведения об дескрипторе.

Бит 2 (0x4)
Отображает сведения об имени дескриптора.

Бит 3 (0x8)
Отображает сведения об дескрипторе для конкретного объекта при наличии.

*KMFlags*
(только режим ядра) Указывает, что должен содержать дисплей. Этот параметр может быть суммой любого из следующих битовых значений. (Значение по умолчанию — 0x3.)

Bit 0 (0x1)
Отображает основные сведения об дескрипторе.

Бит 1 (0x2)
Отображает сведения о объектах.

Бит 2 (0x4)
Отображает записи бесплатного дескриптора. Если этот бит не задан и *не задан дескриптор или задано значение нулю, список отображаемых дескрипторов* не включает бесплатные дескрипторы. Если *Дескриптор* задает один бесплатный дескриптор, он отображается даже в том случае, если этот бит не задан.

Бит 4 (0x10)
Отображает дескриптор из таблицы обработчика ядра вместо текущего процесса.

Бит 5 (0x20)
Интерпретирует дескриптор как идентификатор потока или идентификатор процесса и отображает сведения о соответствующем объекте ядра.

*Обработать*
(только режим ядра) Задает процесс. Вы можете использовать идентификатор процесса или шестнадцатеричный адрес объекта процесса. Этот параметр должен ссылаться на текущий запущенный процесс в целевой системе. Если этот параметр равен -1 или если он не указан, используется текущий процесс. Если этот параметр равен 0, отображается информация обо всех процессах.

*Typename*
Указывает тип дескриптора, который требуется проверить. Отображаются только дескрипторы, соответствующие этому типу. *TypeName* учитывает регистр. Допустимые типы: Event, Section, File, Port, Directory, SymbolicLink, Mutant, WindowStation, Semaphore, Key, Token, Process, Thread, Desktop, IoCompletion, Timer, Job и WaitablePort.

**-?**
(Только в пользовательском режиме) Отображает текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Kdextx86.dll Uext.dll Ntsdexts.dll |
| **Windows XP и более поздние версии** | Kdexts.dll Uext.dll Ntsdexts.dll |

# Дополнительная информация

Дополнительные сведения об дескрипторах см. **в расширении !htrace**, документации по пакету SDK Для Microsoft Windows и *внутренним службам* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Расширение !handle **можно использовать** во время динамической отладки в режиме пользователя и в режиме ядра. Это расширение также можно использовать в файлах дампа в режиме ядра. Однако вы не можете использовать это расширение в файлах дампа в пользовательском режиме, если только вы не создали их с информацией об дескрипторе. (Такие файлы дампа можно создать с помощью **>. команда дампа /mh (создать файл** дампа).)

Во время динамической отладки в режиме пользователя можно использовать **команду closehandle (Close Handle)** для закрытия одного или нескольких дескрипторов.

Ниже приведены примеры пользовательского **режима расширения !handle** . Следующая команда отображает список всех дескрипторов.

```
dbgcmd

0:000> !handle
Handle 4
  Type          Section
Handle 8
  Type          Event
Handle c
  Type          Event
Handle 10
  Type          Event
Handle 14
  Type          Directory
Handle 5c
```

```
  Type          File
6 Handles
Type            Count
Event           3
Section         1
File            1
Directory       1
```

Следующая команда содержит подробные сведения об обработке 0x8.

dbgcmd

```
0:000> !handle 8 f
Handle 8
  Type          Event
  Attributes    0
  GrantedAccess 0x100003:
         Synch
         QueryState,ModifyState
  HandleCount   2
  PointerCount  3
  Name          <none>
  Object Specific Information
    Event Type Auto Reset
    Event is Waiting
```

В следующих примерах приведены примеры **режима ядра !handle**. Следующая команда содержит список всех дескрипторов, включая бесплатные дескрипторы.

dbgcmd

```
kd> !handle 0 4
processor number 0
PROCESS 80559800  SessionId: 0  Cid: 0000    Peb: 00000000  ParentCid: 0000
    DirBase: 00039000  ObjectTable: e1000d60  TableSize: 380.
    Image: Idle

New version of handle table at e1002000 with 380 Entries in use

0000: free handle, Entry address e1002000, Next Entry fffffffe
0004: Object: 80ed5238  GrantedAccess: 001f0fff
0008: Object: 80ed46b8  GrantedAccess: 00000000
000c: Object: e1281d00  GrantedAccess: 000f003f
0010: Object: e1013658  GrantedAccess: 00000000
......
0168: Object: ffb6c748  GrantedAccess: 00000003 (Protected)
016c: Object: ff811f90  GrantedAccess: 0012008b
0170: free handle, Entry address e10022e0, Next Entry 00000458
0174: Object: 80dfd5c8  GrantedAccess: 001f01ff
......
```

Следующая команда содержит подробные сведения о дескрипторе 0x14 в таблице обработчиков ядра.

dbgcmd

```
kd> !handle 14 13
processor number 0
PROCESS 80559800  SessionId: 0  Cid: 0000    Peb: 00000000  ParentCid: 0000
    DirBase: 00039000  ObjectTable: e1000d60  TableSize: 380.
    Image: Idle

Kernel New version of handle table at e1002000 with 380 Entries in use
0014: Object: e12751d0  GrantedAccess: 0002001f
Object: e12751d0  Type: (80ec8db8) Key
    ObjectHeader: e12751b8
        HandleCount: 1  PointerCount: 1
        Directory Object: 00000000  Name:
\REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\SESSION MANAGER\EXECUTIVE
```

Следующая команда содержит сведения обо всех дескрипторах объектов Section во всех процессах.

dbgcmd

```
!handle 0 3 0 Section
...
PROCESS fffffa8004f48940
    SessionId: none  Cid: 0138    Peb: 7f6639bf000  ParentCid: 0004
    DirBase: 10cb74000  ObjectTable: fffff8a00066f700  HandleCount:  39.
    Image: smss.exe

Handle table at fffff8a00066f700 with 39 entries in use

0040: Object: fffff8a000633f00  GrantedAccess: 00000006 (Inherit) Entry:
fffff8a000670100
Object: fffff8a000633f00  Type: (fffffa80035fef20) Section
    ObjectHeader: fffff8a000633ed0 (new version)
        HandleCount: 1  PointerCount: 262144
...
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Кучи

Статья • 05.04.2024

Расширение **!heap** отображает сведения об использовании кучи, управляет точками останова в диспетчере куч, обнаруживает утечки блоков куч, ищет блоки кучи или отображает сведения о куче страницы.

Это расширение поддерживает кучу сегментов и кучу NT. Используйте !heap без параметра для перечисления всех кучи и их типа.

dbgcmd

```
!heap [HeapOptions] [ValidationOptions] [Heap]
!heap -b [{alloc|realloc|free} [Tag]] [Heap | BreakAddress]
!heap -B {alloc|realloc|free} [Heap | BreakAddress]
!heap -l
!heap -s [SummaryOptions] [StatHeapAddress]
!heap -i HeapAddress
!heap -x [-v] Address
!heap -p [PageHeapOptions]
!heap -srch [Size] Pattern
!heap -flt FilterOptions
!heap -stat [-h Handle [-grp GroupBy [MaxDisplay]]]
!heap [-p] -?
!heap -triage [Handle | Address]
```

# Параметры сегментов и кучи NT

Эти параметры работают с кучами сегментов и NT.

**-s**
Указывает, что запрашиваются сводные сведения. Если *сводкаoptions и StatHeapAddress опущены*, то для всех куч, связанных с текущим процессом, отображается сводная информация.

*SummaryOptions*
Может быть любым сочетанием следующих параметров. СводкаOptions не учитывает регистр. Тип !heap -s -? дополнительные сведения.

Развернуть таблицу

| Вариант | Действие |
| --- | --- |
| **-v** | Проверяет все блоки данных. |

| Вариант | Действие |
| --- | --- |
| -b*BucketSize* | Указывает размер контейнера. Значение по умолчанию — 1024 бита. |
| -d*DumpBlockSize* | Указывает размер контейнера. |
| -a | Дамп всех блоков кучи. |
| -c | Указывает, что содержимое каждого блока должно отображаться. |

**-triage** [Адрес дескриптора|*]
Позволяет отладчику автоматически искать сбои в кучах процесса. Если дескриптор кучи указан в качестве аргумента, эта куча проверяется; в противном случае выполняется поиск всех куч, содержащих указанный адрес, и если он найден, он проверяется. Использование **-triage** — единственный способ проверить повреждение кучи с низкой фрагментацией (LFH).

**-x** [**-v**]
Позволяет отладчику искать блок кучи, содержащий указанный адрес. При добавлении -v команда будет выполнять поиск всего пространства виртуальной памяти текущего процесса для указателей на этот блок кучи.

**-L**
Позволяет отладчику обнаруживать утечки блоков куч.

**-i***Address-h HeapAddress*
Отображает сведения о указанной *куче*.

*Адрес*
Указывает адрес для поиска.

**-?**
Отображает краткий текст справки для этого расширения в окне команды отладчика. Используйте **!heap -?** для универсальной справки и **!кучи -p -?** для справки кучи страницы.

# Параметры кучи NT

Эти параметры работают только с кучей NT.

*HeapOptions*
Может быть любым сочетанием следующих параметров. Значения *HeapOptions* чувствительны к регистру.

Развернуть таблицу

| Вариант | Действие |
| --- | --- |
| **-v** | Вызывает отладчик проверить указанную кучу.<br><br>**Обратите внимание,** что этот параметр не обнаруживает повреждение кучи с низкой фрагментацией (LFH). Вместо этого используйте **-triage** . |
| **-a** | Приводит к тому, что отображение будет содержать все сведения для указанной кучи. Размер, в данном случае, округляется до кучи детализации. (Запуск **!кучи** с параметром -a **эквивалентен запуску с** тремя параметрами **-h -f -m**, что может занять много времени.) |
| **-h** | Вызывает включение всех записей, отличных от LFH, для указанной кучи. |
| **-Hl** | Вызывает включение всех записей для указанных куч, включая записи LFH. |
| **-f** | Приводит к включению всех записей бесплатного списка для указанной кучи. |
| **-m** | Приводит к тому, что отображение включает все записи сегмента для указанной кучи. |
| **-t** | Вызывает включение сведений тега для указанной кучи. |
| **-T** | Приводит к включению записей псевдо тега для указанной кучы. |
| **-g** | Приводит к включению сведений о глобальном теге. Глобальные теги связаны с каждым выделением без тегов. |
| **-s** | Приводит к тому, что отображение содержит сводную информацию для указанной кучы. |
| **-k** | (только целевые объекты на основе x86) Приводит к включению стека обратного вывода, связанного с каждой записью. |

*ValidationOptions*
Может быть одним из следующих вариантов. ПроверкаOptions учитывает регистр.

Развернуть таблицу

| Вариант | Действие |
|---|---|
| **-D** | Отключает проверку по вызову указанной кучы. |
| **-E** | Включает проверку по вызову указанной кучи. |
| **-d** | Отключает кучу проверка для указанной кучы. |
| **-e** | Включает кучи проверка для указанной кучи. |

**-i***Heap Address***или***HeapAddress*
Отображает сведения о указанной *куче*.

*BreakAddress*
Указывает адрес блока, в котором должна быть задана или удалена точка останова.

**-b**
Приводит отладчик к созданию условной точки останова в диспетчере куч. За **параметром -b** можно следовать **распределить**, **переделать** или **освободить** память. Это указывает, будет ли точка останова активирована путем выделения, перераспределения или освобождения памяти. Если *BreakAddress* используется для указания адреса блока, то тип точки останова может быть опущен. Если *куча* используется для указания адреса кучи или индекса кучи, необходимо включить тип, а также *параметр Tag* .

*Тег*
Указывает имя тега в куче.

**-B**
Позволяет отладчику удалить условную точку останова из диспетчера кучи. Тип точки останова (**alloc**, **realloc** или **free**) должен быть указан и должен быть таким же, как и для **параметра -b** .

*StatHeapAddress*
Указывает адрес кучи. Если это значение равно 0 или опущено, отображаются все кучи, связанные с текущим процессом.

**-p**
Указывает, что запрашиваются сведения о куче страниц. Если это используется без *pageHeapOptions*, будут отображаться все кучи страниц.

*PageHeapOptions*
Может быть одним из следующих вариантов. PageHeapOptions учитывает регистр. Если параметры не указаны, будут отображаться все возможные дескрипторы кучи страницы.

Развернуть таблицу

| Вариант | Действие |
|---|---|
| **-h***Handle* | Позволяет отладчику отображать подробные сведения о куче страницы с дескриптором дескриптора. |
| **-a***Address* | Позволяет отладчику найти кучу страницы, блок которой содержит *адрес*. Полные сведения о том, как этот адрес относится к блокам кучи полностраничных кучи, например, является ли этот адрес частью кучи страницы, ее смещением внутри блока, а также выделение или освобождение блока. Трассировки стека включаются всякий раз, когда они доступны. При использовании этого параметра размер отображается в нескольких фрагментах детализации распределения кучи. |
| **-t**[**c**\|] [*трассировки]* | Позволяет отладчику отображать собранные трассировки тяжелых пользователей кучи. *Трассировки* указывают количество отображаемых трассировок; значение по умолчанию — четыре. Если существует больше трассировок, чем указанное число, отображаются самые ранние трассировки. Если **используется функция -t** или **-tc** , трассировки сортируются по количеству использования. Если **используется -ts** , трассировки сортируются по размеру. (Параметры **-tc** и **-ts** поддерживаются только в Windows XP; **параметр -t** поддерживается только в Windows XP и более ранних версиях Windows.) |
| **-fi** [*трассировки*] | Вызывает отладчик отображать последние трассировки внедрения ошибок. |

| Вариант | Действие |
| --- | --- |
| | *Трассировки указывают количество отображаемых* значений; значение по умолчанию — 4. |
| **-Все** | Приводит к отображению отладчиком подробных сведений обо всех кучах страниц. |
| **-?** | Позволяет отладчику отображать справку кучи страницы, включая схему блоков кучи. (Эти схемы также можно увидеть в следующем разделе "Примечания".) |

Прежде чем использовать любую **команду расширения !heap -p** , для целевого процесса необходимо включить кучу страницы. Дополнительные сведения см. в следующем разделе "Примечания".

**-srch**
Сканирует все кучы для заданного шаблона.

*Шаблон*
Указывает шаблон, для которого нужно выглядеть.

*Размер*
Может быть одним из следующих вариантов. Это указывает размер шаблона. Требуется значение "-".

Развернуть таблицу

| Вариант | Действие |
| --- | --- |
| **-b** | Шаблон — один РАЗМЕР BYTE. |
| **-w** | Шаблон — это один размер WORD. |
| **-d** | Шаблон — один DWORD в размере. |
| **-q** | Шаблон имеет один размер QWORD. |

Если ни один из указанных выше не указан, предполагается, что шаблон имеет тот же размер, что и указатель компьютера.

**-Flt**
Ограничивает отображение только кучами с указанным размером или диапазоном размеров.

*FilterOptions*
Может быть одним из следующих вариантов. FilterOptions учитывает регистр.

Развернуть таблицу

| Вариант | Действие |
|---|---|
| **Размер s** | Ограничивает отображение только кучами указанного размера. |
| **r***SizeMinSizeMax* | Ограничивает отображение только кучами в указанном диапазоне размеров. |

**-Стат**
Отображает статистику использования для указанной кучы.

**-h***Handle*
Приводит к отображению статистики использования только кучи при *дескрипторе* . Если *дескриптор* равен 0 или опущен, отображается статистика использования для всех кучи.

**-grp***GroupBy*
Переупорядочение отображения, указанного *GroupBy*. *Параметры GroupBy* можно найти в следующей таблице.

Развернуть таблицу

| Вариант | Действие |
|---|---|
| **Объект** | Отображает статистику использования в соответствии с размером выделения. |
| **B** | Отображает статистику использования в соответствии с числом блоков. |
| **S** | Отображает статистику использования в соответствии с общим размером каждого выделения. |

*MaxDisplay*
Ограничивает выходные данные только *maxDisplay* число строк.

# DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows XP и более поздние версии** | Ext.dll Exts.dll |

# Дополнительная информация

Дополнительные сведения о кучах см. в следующих ресурсах:

Книга: *Внутренние* компоненты Microsoft Windows Марк Руссинович и Дэвид Соломон.

Пример 11. Включение проверки кучи страниц

Пример 12. Использование проверки кучи страниц для поиска ошибки

Сведения об использовании средства ведения журнала обработки памяти кучи см . в примере 11. Запуск сеанса частной трассировки

# Замечания

Эта команда расширения может использоваться для выполнения различных задач.

Стандартная **команда !heap используется для отображения сведений о куче** для текущего процесса. (Это должно использоваться только для процессов пользовательского режима. **Команда расширения пула !pool** должна использоваться для системных процессов.)

Команды **!heap -b** и **!heap -B** используются для создания и удаления условных точек останова в диспетчере кучи.

Команда **!heap -l** обнаруживает утечки блоков кучи. Он использует алгоритм сборщика мусора для обнаружения всех занятых блоков из куч, на которые нет ссылки в адресном пространстве процесса. Для огромных приложений может потребоваться несколько минут. Эта команда доступна только в Windows XP и более поздних версиях Windows.

Команда **!heap -x** ищет блок кучи, содержащий заданный адрес. **Если используется параметр -v**, эта команда будет дополнительно выполнять поиск всего пространства виртуальной памяти текущего процесса для указателей на этот блок кучи. Эта команда доступна только в Windows XP и более поздних версиях Windows.

Команда **!heap -p** отображает различные формы сведений о куче страницы. Перед использованием **!heap -p** необходимо включить кучу страниц для целевого процесса. Это делается с помощью программы глобальных флагов (gflags.exe). Для этого запустите служебную программу, введите имя целевого приложения в **текстовом поле "Имя** файла изображения", выберите "Параметры **файла изображения" и** "Включить кучу **страницы" и** нажмите кнопку "**Применить**". Кроме того, можно запустить служебную программу global Flags из окна командной строки, введя **gflags /i***xxx.exe***+hpa**, где *xxx.exe* — имя целевого приложения.

**Команды !heap -p -t[c|s]** не поддерживаются за пределами Windows XP. Используйте средство UMDH, предоставленное пакетом отладчика, чтобы получить аналогичные результаты.

Команда **!heap -srch** отображает эти записи кучи, содержащие определенный указанный шаблон.

Команда **!heap -flt** ограничивает отображение только выделения кучи указанного размера.

Команда **!heap -stat** отображает статистику использования кучи.

Ниже приведен пример стандартной **команды !heap** :

dbgcmd

```
0:000> !ntsdexts.heap -a
Index   Address  Name      Debugging options enabled
  1:   00250000
    Segment at 00250000 to 00350000 (00056000 bytes committed)
    Flags:                50000062
    ForceFlags:           40000060
    Granularity:          8 bytes
    Segment Reserve:      00100000
    Segment Commit:       00004000
    DeCommit Block Thres:00000400
    DeCommit Total Thres:00002000
    Total Free Size:      000003be
    Max. Allocation Size:7ffddfff
    Lock Variable at:     00250b54
    Next TagIndex:        0012
    Maximum TagIndex:     07ff
    Tag Entries:          00350000
    PsuedoTag Entries:    00250548
    Virtual Alloc List:   00250050
    UCR FreeList:         002504d8
    128-bit bitmap of free lists
    FreeList Usage:       00000014 00000000 00000000 00000000
              Free    Free
```

```
              List    List
#       Head      Blink      Flink
    FreeList[ 00 ] at 002500b8: 002a4378 . 002a4378
                                0x02 - HEAP_ENTRY_EXTRA_PRESENT
                                0x04 - HEAP_ENTRY_FILL_PATTERN
        Entry     Prev    Cur   0x10 - HEAP_ENTRY_LAST_ENTRY

Address   Size    Size  flags
002a4370: 00098 . 01c90 [14] - free
    FreeList[ 02 ] at 002500c8: 0025cb30 . 002527b8
002527b0: 00058 . 00010 [04] - free
0025cb28: 00088 . 00010 [04] - free
    FreeList[ 04 ] at 002500d8: 00269a08 . 0026e530
0026e528: 00038 . 00020 [04] - free
0026a4d0: 00038 . 00020 [06] - free
0026f9b8: 00038 . 00020 [04] - free
0025cda0: 00030 . 00020 [06] - free
00272660: 00038 . 00020 [04] - free
0026ab60: 00038 . 00020 [06] - free
00269f20: 00038 . 00020 [06] - free
00299818: 00038 . 00020 [04] - free
0026c028: 00038 . 00020 [06] - free
00269a00: 00038 . 00020 [46] - free

    Segment00 at 00250b90:
Flags:           00000000
Base:            00250000
First Entry:     00250bc8
Last Entry:      00350000
Total Pages:     00000080
Total UnCommit:  00000055
Largest UnCommit:000aa000
UnCommitted Ranges: (1)
    002a6000: 000aa000

    Heap entries for Segment00 in Heap 250000
                         0x01 - HEAP_ENTRY_BUSY
                         0x02 - HEAP_ENTRY_EXTRA_PRESENT
                         0x04 - HEAP_ENTRY_FILL_PATTERN
                         0x08 - HEAP_ENTRY_VIRTUAL_ALLOC
                         0x10 - HEAP_ENTRY_LAST_ENTRY
                         0x20 - HEAP_ENTRY_SETTABLE_FLAG1
                         0x40 - HEAP_ENTRY_SETTABLE_FLAG2
Entry     Prev    Cur    0x80 - HEAP_ENTRY_SETTABLE_FLAG3

Address   Size    Size  flags       (Bytes used)    (Tag name)
00250000: 00000 . 00b90 [01] - busy (b90)
00250b90: 00b90 . 00038 [01] - busy (38)
00250bc8: 00038 . 00040 [07] - busy (24), tail fill (NTDLL!LDR Database)
00250c08: 00040 . 00060 [07] - busy (48), tail fill (NTDLL!LDR Database)
00250c68: 00060 . 00028 [07] - busy (10), tail fill (NTDLL!LDR Database)
00250c90: 00028 . 00060 [07] - busy (48), tail fill (NTDLL!LDR Database)
00250cf0: 00060 . 00050 [07] - busy (38), tail fill (Objects=  80)
00250d40: 00050 . 00048 [07] - busy (2e), tail fill (NTDLL!LDR Database)
00250d88: 00048 . 00c10 [07] - busy (bf4), tail fill (Objects>1024)
```

```
00251998: 00c10 . 00030 [07] - busy (12), tail fill (NTDLL!LDR Database)
...
002525c0: 00030 . 00060 [07] - busy (48), tail fill (NTDLL!LDR Database)
00252620: 00060 . 00050 [07] - busy (38), tail fill (NTDLL!LDR Database)
00252670: 00050 . 00040 [07] - busy (22), tail fill (NTDLL!CSRSS Client)
002526b0: 00040 . 00040 [07] - busy (24), tail fill (Objects=  64)
002526f0: 00040 . 00040 [07] - busy (24), tail fill (Objects=  64)
00252730: 00040 . 00028 [07] - busy (10), tail fill (Objects=  40)
00252758: 00028 . 00058 [07] - busy (3c), tail fill (Objects=  88)
002527b0: 00058 . 00010 [04] free fill
002527c0: 00010 . 00058 [07] - busy (3c), tail fill (NTDLL!LDR Database)
00252818: 00058 . 002d0 [07] - busy (2b8), tail fill (Objects= 720)
00252ae8: 002d0 . 00330 [07] - busy (314), tail fill (Objects= 816)
00252e18: 00330 . 00330 [07] - busy (314), tail fill (Objects= 816)
00253148: 00330 . 002a8 [07] - busy (28c), tail fill (NTDLL!LocalAtom)
002533f0: 002a8 . 00030 [07] - busy (18), tail fill (NTDLL!LocalAtom)
00253420: 00030 . 00030 [07] - busy (18), tail fill (NTDLL!LocalAtom)
00253450: 00030 . 00098 [07] - busy (7c), tail fill (BASEDLL!LMEM)
002534e8: 00098 . 00060 [07] - busy (44), tail fill (BASEDLL!TMP)
00253548: 00060 . 00020 [07] - busy (1), tail fill (Objects=  32)
00253568: 00020 . 00028 [07] - busy (10), tail fill (Objects=  40)
00253590: 00028 . 00030 [07] - busy (16), tail fill (Objects=  48)
...
0025ccb8: 00038 . 00060 [07] - busy (48), tail fill (NTDLL!LDR Database)
0025cd18: 00060 . 00058 [07] - busy (3c), tail fill (NTDLL!LDR Database)
0025cd70: 00058 . 00030 [07] - busy (18), tail fill (NTDLL!LDR Database)
0025cda0: 00030 . 00020 [06] free fill (NTDLL!Temporary)
0025cdc0: 00020 . 00258 [07] - busy (23c), tail fill (Objects= 600)
0025d018: 00258 . 01018 [07] - busy (1000), tail fill (Objects>1024)
0025e030: 01018 . 00060 [07] - busy (48), tail fill (NTDLL!LDR Database)
...
002a4190: 00028 . 00118 [07] - busy (100), tail fill (BASEDLL!GMEM)
002a42a8: 00118 . 00030 [07] - busy (18), tail fill (Objects=  48)
002a42d8: 00030 . 00098 [07] - busy (7c), tail fill (Objects= 152)
002a4370: 00098 . 01c90 [14] free fill
002a6000:      000aa000      - uncommitted bytes.
```

Ниже приведен пример **команды !heap -l** :

dbgcmd

```
1:0:011> !heap -l
1:Heap 00170000
Heap 00280000
Heap 00520000
Heap 00b50000
Heap 00c60000
Heap 01420000
Heap 01550000
Heap 016d0000
Heap 019b0000
Heap 01b40000
Scanning VM ...
```

```
## Entry     User      Heap      Segment       Size  PrevSize  Flags

001b2958  001b2960  00170000  00000000        40        18  busy extra
001b9cb0  001b9cb8  00170000  00000000        80       300  busy extra
001ba208  001ba210  00170000  00000000        80        78  busy extra
001cbc90  001cbc98  00170000  00000000        e0        48  busy extra
001cbd70  001cbd78  00170000  00000000        d8        e0  busy extra
001cbe90  001cbe98  00170000  00000000        68        48  busy extra
001cbef8  001cbf00  00170000  00000000        58        68  busy extra
001cc078  001cc080  00170000  00000000        f8       128  busy extra
001cc360  001cc368  00170000  00000000        80        50  busy extra
001cc3e0  001cc3e8  00170000  00000000        58        80  busy extra
001fe550  001fe558  00170000  00000000       150       278  busy extra
001fe6e8  001fe6f0  00170000  00000000        48        48  busy extra
002057a8  002057b0  00170000  00000000        58        58  busy extra
00205800  00205808  00170000  00000000        48        58  busy extra
002058b8  002058c0  00170000  00000000        58        70  busy extra
00205910  00205918  00170000  00000000        48        58  busy extra
00205958  00205960  00170000  00000000        90        48  busy extra
00246970  00246978  00170000  00000000        60        88  busy extra
00251168  00251170  00170000  00000000        78        d0  busy extra
user_flag
00527730  00527738  00520000  00000000        40        40  busy extra
00527920  00527928  00520000  00000000        40        80  busy extra
21 leaks detected.
```

Таблица в этом примере содержит все 21 утечки.

Ниже приведен пример **команды !heap -x** :

dbgcmd

```
0:011> !heap 002057b8 -x
## Entry     User      Heap      Segment       Size  PrevSize  Flags

002057a8  002057b0  00170000  00170640        58        58  busy extra
```

Ниже приведен пример **команды !heap -x -v** :

dbgcmd

```
1:0:011> !heap 002057b8 -x -v
## 1:Entry     User      Heap      Segment       Size  PrevSize  Flags

002057a8  002057b0  00170000  00170640        58        58  busy extra

Search VM for address range 002057a8 - 002057ff : 00205990 (002057d0),
```

В этом примере есть указатель на этот блок кучи по адресу 0x00205990.

Ниже приведен пример **команды !heap -flt s** :

dbgcmd

```
0:001>!heap -flt s 0x50
```

При этом будут отображаться все выделения размера 0x50.

Ниже приведен пример **команды !heap -flt r** :

dbgcmd

```
0:001>!heap -flt r 0x50 0x80
```

Будет отображаться каждое выделение, размер которого составляет от 0x50 до 0x7F.

Ниже приведен пример **команды !heap -srch** .

dbgcmd

```
0:001> !heap -srch 77176934
    _HEAP @ 00090000
   in HEAP_ENTRY: Size : Prev Flags - UserPtr UserSize - state
        00099A48: 0018 : 0005 [01] - 00099A50 (000000B8) - (busy)
          ole32!CALLFRAME_CACHE<INTERFACE_HELPER_CLSID>::`vftable'
    _HEAP @ 00090000
   in HEAP_ENTRY: Size : Prev Flags - UserPtr UserSize - state
        00099B58: 0018 : 0005 [01] - 00099B60 (000000B8) - (busy)
          ole32!CALLFRAME_CACHE<INTERFACE_HELPER_CLSID>::`vftable'
```

На следующих схемах показано расположение блоков кучи.

Блок кучи светлой страницы: выделено:

dbgcmd

```
 +-----+---------------+---+
 |     |               |   |
 +-----+---------------+---+
    ^         ^          ^
    |         |          8 suffix bytes (filled with 0xA0)
    |         User allocation (filled with E0 if zeroing not requested)
    Block header (starts with 0xABCDAAAA and ends with 0xDCBAAAAA)
```

Блок кучи светлой страницы — освобожден:

dbgcmd

```
+-----+--------------+---+
|     |              |   |
+-----+--------------+---+
   ^         ^          ^
   |         |          8 suffix bytes (filled with 0xA0)
   |         User allocation (filled with F0 bytes)
   Block header (starts with 0xABCDAAA9 and ends with 0xDCBAAA9)
```

Блокировка кучи полной страницы— выделена:

dbgcmd

```
+-----+---------+---+-------
|     |         |   |  ... N/A page
+-----+---------+---+-------
   ^       ^      ^
   |       |      0-7 suffix bytes (filled with 0xD0)
   |       User allocation (if zeroing not requested, filled
           with C0)
   Block header (starts with 0xABCDBBBB and ends with 0xDCBABBBB)
```

Полный блок кучи страницы — освобожден:

dbgcmd

```
+-----+---------+---+-------
|     |         |   |  ... N/A page
+-----+---------+---+-------
   ^       ^      ^
   |       |      0-7 suffix bytes (filled with 0xD0)
   |       User allocation (filled with F0 bytes)
   Block header (starts with 0xABCDBBA and ends with 0xDCBABBBA)
```

Чтобы просмотреть трассировку стека выделения или освобождения блока кучи или блока полной кучи страницы, используйте dt DPH_BLOCK_INFORMATION **с адресом заголовка, а затем dds с полем StackTrace блока.**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Справка

Статья • 05.04.2024

Расширение **!help** отображает текст справки, описывающий команды расширения, экспортированные из библиотеки DLL расширения.

Не путайте эту команду расширения с **помощью команды ? (Справка по командам)** или **команды .help (Meta-Command Help).**

dbgcmd

```
![ExtensionDLL.]help [-v] [CommandName]
```

## Параметры

*ExtensionDLL*
Отображает справку по указанной библиотеке DLL расширения. Введите имя библиотеки DLL расширения без расширения .dll имени файла. Если DLL-файл не находится в пути поиска расширения (как показано с помощью **цепочки (расширения отладчика списка)),** добавьте путь к DLL-файлу. Например, чтобы отобразить справку по uext.dll, введите **!uext.help** или !*Path***winext\uext.help**.

Если опустить *ExtensionDLL*, отладчик отобразит текст справки для первой библиотеки DLL расширения в списке загруженных БИБЛИОТЕК DLL.

**-v**
Отображает наиболее подробный текст справки. Эта функция не поддерживается во всех библиотеках DLL.

*CommandName*
Отображает только текст справки для указанной команды. Эта функция не поддерживается во всех библиотеках DLL или для всех команд.

## DLL-библиотеки

Это расширение поддерживается большинством библиотек DLL расширений.

## Замечания

Некоторые отдельные команды также отображают текст справки, если используется **параметр /?** или **-?** с именем команды.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !homedir

Статья • 05.04.2024

Расширение **!homedir** задает каталог по умолчанию, используемый сервером символов и исходным сервером.

dbgcmd

```
!homedir Directory
!homedir
```

## Параметры

*Каталог*
Указывает новый каталог, используемый в качестве домашнего каталога.

## DLL-библиотеки

Dbghelp.dll

## Замечания

**Если расширение !homedir** используется без аргумента, отображается текущий домашний каталог.

Кэш исходного сервера находится в подкаталоге src домашнего каталога. Нижестоящему хранилищу для сервера символов по умолчанию используется подкаталог домашнего каталога, если не указано другое расположение.

При запуске WinDbg домашний каталог — это каталог, в котором были установлены средства отладки для Windows. **Расширение !homedir** можно использовать для изменения этого значения.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Hstring

Статья • 05.04.2024

Расширение **!hstring** отображает поля **HSTRING**. Последний элемент в отображении — это сама строка.

```
dbgcmd
```

```
!hstring Address
```

## Параметры

*Адрес*
Адрес **HSTRING**.

## Замечания

Тип **данных HSTRING** поддерживает строки с внедренными символами NULL. **Однако расширение !hstring** отображает строку только до первого символа NULL. Чтобы просмотреть всю строку, включая внедренные символы NULL, используйте **расширение !hstring2** .

## См. также

команды отладчика среда выполнения Windows

**!hstring2**

**!winrterr**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !hstring2

Статья • 05.04.2024

Расширение **!hstring2** отображает весь **HSTRING** , включая все внедренные символы NULL в самой строке.

dbgcmd

```
!hstring2 Address
```

## Параметры

*Адрес*
Адрес **HSTRING**.

## См. также

команды отладчика среда выполнения Windows

**!Hstring**

**!winrterr**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !htrace

Статья • 05.04.2024

Расширение **!htrace** отображает сведения трассировки стека для одного или нескольких дескрипторов.

Синтаксис пользовательского режима

dbgcmd

```
!htrace [Handle [Max_Traces]]
!htrace -enable [Max_Traces]
!htrace -snapshot
!htrace -diff
!htrace -disable
!htrace -?
```

Синтаксис режима ядра

dbgcmd

```
!htrace [Handle [Process [Max_Traces]]]
!htrace -?
```

# Параметры

*Дескриптор указывает дескриптор* , трассировка стека которого будет отображаться. Если *дескриптор* равен 0 или опущен, будут отображаться трассировки стека для всех дескрипторов процесса.

*Процесс* (только в режиме ядра) Указывает процесс, дескриптор которого будет отображаться. Если *процесс* равен 0 или опущен, используется текущий процесс. В пользовательском режиме текущий процесс всегда используется.

Max_Traces Указывает максимальное количество отображаемых трассировок стека. В пользовательском режиме, если этот параметр опущен, будут отображаться все трассировки стека для целевого процесса.

**-enable** (только в пользовательском режиме) включает трассировку и принимает первый моментальный снимок данных дескриптора, используемых в качестве начального состояния параметром **-diff** .

**-snapshot (только в пользовательском режиме) принимает моментальный снимок** текущей информации дескриптора для использования в качестве начального состояния параметром **-diff** .

**-diff** (только в пользовательском режиме) сравнивает текущие данные дескриптора с последним моментальным снимком данных дескриптора, которые были приняты. Отображает все дескрипторы, которые по-прежнему открыты.

**-disable (только в пользовательском режиме) отключает** трассировку.

**-?**

Отображает краткий текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Windows XP и более поздние версии

Kdexts.dll Ntsdexts.dll

# Дополнительная информация

Чтобы отобразить дополнительные сведения о конкретном дескрипторе , **используйте расширение !handle** .

Сведения об дескрипторах *см. в разделе* "Внутренние компоненты Microsoft Windows" Марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Перед **использованием !htrace** необходимо включить трассировку дескриптора. Одним из способов включения трассировки дескриптора является ввод **команды !htrace -enable** . Если трассировка дескриптора включена, данные трассировки стека сохраняются каждый раз, когда процесс открывает дескриптор, закрывает дескриптор или ссылается на недопустимый дескриптор. Это данные трассировки стека, отображаемые **!htrace** .

**Обратите внимание**, что можно также включить трассировку дескриптора, активировав **средство проверки приложений для целевого процесса и выбрав параметр "Дескриптор".**

Некоторые трассировки, сообщаемые **!htrace** , могут находиться в другом контексте процесса. В этом случае возвращаемые адреса могут не разрешаться должным образом в текущем контексте процесса или разрешать неправильные символы.

В следующем примере отображаются сведения обо всех дескрипторах в процессе 0x81400300.

dbgcmd

```
kd> !htrace 0 81400300
Process 0x81400300
ObjectTable 0xE10CCF60
##

Handle 0x7CC - CLOSE:
0x8018FCB9: ntoskrnl!ExDestroyHandle+0x103
0x801E1D12: ntoskrnl!ObpCloseHandleTableEntry+0xE4
0x801E1DD9: ntoskrnl!ObpCloseHandle+0x85
0x801E1EDD: ntoskrnl!NtClose+0x19
0x010012C1: badhandle!mainCRTStartup+0xE3
## 0x77DE0B2F: KERNEL32!BaseProcessStart+0x3D

Handle 0x7CC - OPEN:
0x8018F44A: ntoskrnl!ExCreateHandle+0x94
0x801E3390: ntoskrnl!ObpCreateUnnamedHandle+0x10C
0x801E7317: ntoskrnl!ObInsertObject+0xC3
0x77DE23B2: KERNEL32!CreateSemaphoreA+0x66
0x010011C5: badhandle!main+0x45
0x010012C1: badhandle!mainCRTStartup+0xE3
## 0x77DE0B2F: KERNEL32!BaseProcessStart+0x3D

Handle 0x7DC - BAD REFERENCE:
0x8018F709: ntoskrnl!ExMapHandleToPointerEx+0xEA
0x801E10F2: ntoskrnl!ObReferenceObjectByHandle+0x12C
0x801902BE: ntoskrnl!NtSetEvent+0x6C
0x80154965: ntoskrnl!_KiSystemService+0xC4
0x010012C1: badhandle!mainCRTStartup+0xE3
## 0x77DE0B2F: KERNEL32!BaseProcessStart+0x3D

Handle 0x7DC - CLOSE:
0x8018FCB9: ntoskrnl!ExDestroyHandle+0x103
0x801E1D12: ntoskrnl!ObpCloseHandleTableEntry+0xE4
0x801E1DD9: ntoskrnl!ObpCloseHandle+0x85
0x801E1EDD: ntoskrnl!NtClose+0x19
0x010012C1: badhandle!mainCRTStartup+0xE3
## 0x77DE0B2F: KERNEL32!BaseProcessStart+0x3D

Handle 0x7DC - OPEN:
0x8018F44A: ntoskrnl!ExCreateHandle+0x94
0x801E3390: ntoskrnl!ObpCreateUnnamedHandle+0x10C
0x801E7317: ntoskrnl!ObInsertObject+0xC3
0x77DE265C: KERNEL32!CreateEventA+0x66
0x010011A0: badhandle!main+0x20
```

```
0x010012C1: badhandle!mainCRTStartup+0xE3
0x77DE0B2F: KERNEL32!BaseProcessStart+0x3D
##

Parsed 0x6 stack traces.
Dumped 0x5 stack traces.
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !imggp

Статья • 05.04.2024

Расширение **!imggp** отображает значение записи каталога глобального указателя (GP) для 64-разрядного образа.

dbgcmd

```
!imggp Address
```

## Параметры

*Адрес*
Указывает базовый адрес изображения.

## DLL-библиотеки

Ext.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !imgreloc

Статья • 05.04.2024

Расширение **!imgreloc** отображает адреса каждого загруженного модуля и указывает их бывшие адреса до их перемещения.

```
dbgcmd
```

```
!imgreloc Address
```

## Параметры

*Адрес*
Указывает базовый адрес изображения.

## DLL-библиотеки

Ext.dll

## Замечания

Рассмотрим пример:

```
dbgcmd
```

```
0:000> !imgreloc 00400000
00400000 Prymes - at preferred address
010e0000 appvcore - RELOCATED from 00400000
5b2f0000 verifier - at preferred address
5d160000 ShimEng - at preferred address
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !kuser

Статья • 05.04.2024

Расширение **!kuser** отображает общую страницу пользовательского режима (KUSER_SHARED_DATA).

dbgcmd

```
!kuser
```

## DLL-библиотеки

Exts.dll

## Замечания

Страница KUSER_SHARED_DATA предоставляет ресурс и другие сведения о пользователе, вошедшем в систему.

Ниже приведен пример. Обратите внимание, что в этом примере число галок отображается как в необработанной форме, так и в более удобной для пользователя форме, которая находится в скобках. Удобный для пользователя дисплей доступен только в Windows XP и более поздних версиях.

dbgcmd

```
kd> !kuser
_KUSER_SHARED_DATA at 7ffe0000
TickCount:    fa00000 * 00482006 (0:20:30:56.093)
TimeZone Id: 2
ImageNumber Range: [14c .. 14c]
Crypto Exponent: 0
SystemRoot: 'F:\WINDOWS'
```


## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Список

Статья • 05.04.2024

**Расширение !list** выполняет указанные команды отладчика несколько раз для каждого элемента в связанном списке.

dbgcmd

```
!list -t [Module!]Type.Field -x "Commands" [-a "Arguments"] [Options]
StartAddress
!list " -t [Module!]Type.Field -x \"Commands\" [-a \"Arguments\"] [Options]
StartAddress "
!list -h
```

# Параметры

*Модуль*
Необязательный параметр, указывающий модуль, определяющий эту структуру. Если существует вероятность того, что *тип* может соответствовать допустимому символу в другом модуле, следует включить *модуль* , чтобы устранить неоднозначность.

*Тип*
Задает имя структуры данных.

*Поле*
Указывает поле, содержащее ссылку списка. Это может быть последовательность полей, разделенных точками (другими словами, **Type.Field.Subfield.Subsubfield** и т. д.).

**-x** "*Команды*"
Указывает выполняемые команды. Это может быть любое сочетание команд отладчика. Он должен быть заключен в кавычки. Если указаны несколько команд, разделите их точкой с запятой, заключите всю коллекцию **аргументов !list** в кавычки и используйте escape-символ (\ ) перед каждой кавычками внутри этих внешних кавычки. Если *команды опущены*, значение по умолчанию — **dp (display Memory)**.

**-a** "*Аргументы*"
Указывает аргументы для передачи *в параметр Commands* . Это должно быть заключено в кавычки. *Аргументы могут быть любой допустимой строкой аргументов* , которая обычно разрешена выполнять эту команду, за исключением

того, что *аргументы не могут содержать кавычки* . Если псевдорегистрация **$extret включена в *команды*, параметр -a "*Arguments*" может быть опущен**.

*Параметры* могут быть любым количеством следующих параметров:

**-e**
Эхо выполняет команду для каждого элемента.

**-m***Max*
Указывает максимальное количество элементов для выполнения команды.

*StartAddress*
Задает адрес первой структуры данных. Это адрес в верхней части структуры, а не обязательно адрес поля ссылки.

**-h**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Ext.dll

# Замечания

**Расширение !list** будет проходить через связанный список и выдавать указанную команду один раз для каждого элемента списка.

Псевдорегистрирующий **$extret** имеет значение адреса записи списка для каждого элемента списка. Для каждого элемента выполняются команды *командной строки*. Эта командная строка может ссылаться на этот псевдорегистрирующий регистр с помощью синтаксиса **$extret** . Если это не отображается в командной строке, значение адреса записи списка добавляется к концу командной строки перед выполнением. Если необходимо указать, где это значение должно отображаться в команде, необходимо явно указать этот псевдорегистрирующий регистр.

Эта последовательность команд будет выполняться до тех пор, пока список не завершится в указателе NULL, или завершится циклом обратно на первый элемент. Если список циклит обратно на более поздний элемент, эта команда не остановится. Однако эту команду можно остановить в любое время с помощью **CTRL+C** в KD и CDB или **Debug | Разрыв** или CTRL+BREAK в WinDbg.

Каждый раз при выполнении команды адрес текущей структуры будет использоваться в качестве *адреса* по умолчанию, если используется команда имеет необязательные параметры адреса.

Ниже приведены два примера использования этой команды в пользовательском режиме. Обратите внимание, что использование режима ядра также возможно, но соответствует другому синтаксису.

В качестве простого примера предположим, что у вас есть структура, имя типа которой — **MYTYPE**, и которая содержит ссылки в его **ссылках .links. Flink** и **.links. Поля мигания** . У вас есть связанный список, начинающийся со структуры по 0x6BC000. Следующая команда расширения будет проходить через список, и для каждого элемента будет выполняться **команда dd** L2. Так как адрес не указан в команде **dd** , он будет принимать адрес головы списка в качестве нужного адреса. Это приводит к отображению первых двух DWORD в каждой структуре.

dbgcmd

```
0:000> !list -t MYTYPE.links.Flink -x "dd" -a "L2" 0x6bc00
```

В более сложном примере рассмотрим вариант использования **$extret**. Он следует списку типов _LIST_ENTRY в **RtlCriticalSectionList**. Для каждого элемента отображается первые четыре DWORDS, а затем отображается структура _RTL_CRITICAL_SECTION_DEBUG, расположенная в смещение 8 байтов до **элемента Flink** записи списка.

dbgcmd

```
0:000> !list "-t ntdll!_LIST_ENTRY.Flink -e -x \"dd @$extret l4; dt
ntdll!_RTL_CRITICAL_SECTION_DEBUG @$extret-0x8\"
ntdll!RtlCriticalSectionList"
dd @$extret l4; dt ntdll!_RTL_CRITICAL_SECTION_DEBUG @$extret-0x8
7c97c0c8  7c97c428 7c97c868 01010000 00000080
   +0x000 Type             : 1
   +0x002 CreatorBackTraceIndex : 0
   +0x004 CriticalSection  : (null)
   +0x008 ProcessLocksList : _LIST_ENTRY [ 0x7c97c428 - 0x7c97c868 ]
   +0x010 EntryCount       : 0x1010000
   +0x014 ContentionCount  : 0x80
   +0x018 Spare            : [2] 0x7c97c100

dd @$extret l4; dt ntdll!_RTL_CRITICAL_SECTION_DEBUG @$extret-0x8
7c97c428  7c97c448 7c97c0c8 00000000 00000000
   +0x000 Type             : 0
   +0x002 CreatorBackTraceIndex : 0
   +0x004 CriticalSection  : 0x7c97c0a0
   +0x008 ProcessLocksList : _LIST_ENTRY [ 0x7c97c448 - 0x7c97c0c8 ]
```

```
    +0x010 EntryCount       : 0
    +0x014 ContentionCount  : 0
    +0x018 Spare            : [2] 0
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !lmi

Статья • 05.04.2024

Расширение !lmi **отображает подробные** сведения о модуле.

```
dbgcmd
```

```
!lmi Module
```

## Параметры

*Модуль*
Задает загруженный модуль по имени или по базовому адресу.

## DLL-библиотеки

Dbghelp.dll

## Замечания

Адреса модулей можно определить с помощью **команды lm (список загруженных модулей).**

Расширение **!lmi** анализирует заголовки модуля и отображает отформатированную сводку сведений о нем. Если заголовки модуля выводятся, отображается сообщение об ошибке. Чтобы просмотреть более подробное отображение сведений о заголовке, используйте **команду расширения !dh** .

Эта команда показывает ряд полей, каждый из которых имеет другое название. Некоторые из этих названий имеют конкретные значения:

- В **поле "Имя** изображения" отображается имя исполняемого файла, включая расширение. Как правило, полный путь включен в пользовательский режим, но не в режиме ядра.
- В поле "Модуль" отображается *имя* модуля. Обычно это просто имя файла без расширения. В некоторых случаях имя модуля значительно отличается от имени файла.

- В **поле "Тип** символа" отображаются сведения о попытках отладчика загрузить символы этого модуля. Описание различных значений состояния см . в разделе "Аббревиация состояния символов". Если символы загружены, имя файла символов следует этому.

- Первый адрес в модуле отображается как **базовый адрес**. Размер модуля отображается как **размер**. Таким образом, если **базовый адрес** равен "faab4000" и **размер** равен "2000", модуль расширяется от 0xFAAB4000 до 0xFAAB5FFF включительно.

Рассмотрим пример:

dbgcmd

```
0:000> lm
start    end        module name
00400000 0042d000   Prymes     C (pdb symbols)              Prymes.pdb
77e80000 77f35000   KERNEL32     (export symbols)
C:\WINNT\system32\KERNEL32.dll
77f80000 77ffb000   ntdll        (export symbols)           ntdll.dll

0:000> !lmi 00400000
Loaded Module Info: [00400000]
         Module: Prymes
   Base Address: 00400000
     Image Name: Prymes.exe
   Machine Type: 332 (I386)
     Time Stamp: 3c76c346 Fri Feb 22 14:16:38 2002
           Size: 2d000
       CheckSum: 0
Characteristics: 230e stripped
Debug Data Dirs: Type Size     VA  Pointer
                 MISC  110,     0,   77a00 [Data not mapped]
    Symbol Type: EXPORT   - PDB not found
    Load Report: export symbols
```

Описание аббревиаций, показанных в **строке характеристик** этого примера, см . в разделе "Аббревиация состояния символов".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Mui

Статья • 05.04.2024

Расширение **!mui** отображает сведения о кэше многоязычный пользовательский интерфейс (MUI).

dbgcmd

```
!mui -c
!mui -s
!mui -r ModuleAddress
!mui -i
!mui -f
!mui -t
!mui -u
!mui -d ModuleAddress
!mui -e ModuleAddress
!mui -?
```

# Параметры

**-c**
Приводит к тому, что выходные данные включают идентификатор языка (идентификатор), указатель на модуль, указатель на данные конфигурации ресурсов и указатель на связанную библиотеку DLL MUI для каждого модуля.

**-s**
(Только режим ядра) Приводит к включению полного пути к файлам для модуля и связанной библиотеки DLL MUI для каждого модуля.

**-r** **** *ModuleAddress*
Приводит к отображению данных конфигурации ресурсов модуля в *ModuleAddress* . К ним относятся тип файла, значение проверка sum и типы ресурсов.

**-i**
Приводит к тому, что выходные данные включают установленные и лицензированные языки MUI и связанные с ними сведения.

**-f**
Приводит к тому, что выходные данные включают в себя резервный список объединенных языков загрузчика.

**-t**
Приводит к тому, что выходные данные включают язык предпочтений потока.

**-u**
Приводит к тому, что выходные данные включают текущий параметр языка пользовательского интерфейса пользователя потока потока.

**-d** **** *ModuleAddress*
Приводит к тому, что выходные данные содержат содержащиеся ресурсы для модуля в *ModuleAddress*.

**-e** **** *ModuleAddress*
Приводит к тому, что выходные данные включают содержащиеся типы ресурсов для модуля в *ModuleAddress*.

**-?**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows XP** | Рекомендации недоступны |
| **Windows Vista и более поздние версии** | Exts.dll |

# Дополнительная информация

Сведения о формате данных конфигурации mul и ресурсов см. в документации по пакету SDK для Microsoft Windows.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !net_send

Статья • 26.03.2024

Команда расширения !net_send устарела и больше не поддерживается.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !obja

Статья • 05.04.2024

Расширение **!obja** отображает атрибуты объекта в диспетчере объектов.

```
dbgcmd
```

```
!obja Address
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес заголовка объекта, который требуется проверить.

## DLL-библиотеки

Ext.dll

## Дополнительная информация

Дополнительные сведения об объектах и диспетчере объектов см. в документации по пакету SDK для Microsoft Windows, документации по комплекту драйверов Windows (WDK) и *внутренним* службам Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Перечислены атрибуты, относящиеся к указанному объекту. Допустимые атрибуты:

```
C++
```

```
#define OBJ_INHERIT             0x00000002L
#define OBJ_PERMANENT           0x00000010L
#define OBJ_EXCLUSIVE           0x00000020L
#define OBJ_CASE_INSENSITIVE    0x00000040L
#define OBJ_OPENIF              0x00000080L
#define OBJ_OPENLINK            0x00000100L
#define OBJ_VALID_ATTRIBUTES    0x000001F2L
```

Рассмотрим пример:

dbgcmd

```
kd> !obja 80967768
Obja +80967768 at 80967768:
        OBJ_INHERIT
        OBJ_PERMANENT
        OBJ_EXCLUSIVE
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Владелец

Статья • 05.04.2024

Расширение **!owner** отображает владельца модуля или функции.

```
dbgcmd
```

```
!owner [Module[!Symbol]]
```

## Параметры

*Модуль*
Указывает модуль, владелец которого необходим. Звездочка (*) в конце *модуля* представляет любое количество дополнительных символов.

*Символ*
Указывает символ в *модуле* , владелец которого необходим. Звездочка (*) в конце символа представляет любое количество дополнительных символов. Если *символ* опущен, отображается владелец всего модуля.

## DLL-библиотеки

Ext.dll

## Замечания

Если параметры не используются и произошел сбой, **!owner** отобразит имя владельца модуля или функции сбоя.

При передаче имени модуля или функции в **расширение !owner** отладчик отображает слово **"Отслеживание"** , за которым следует имя владельца указанного модуля или функции.

Чтобы это расширение отображалось полезно, необходимо сначала создать файл triage.ini, содержащий имена владельцев модулей и функций.

Дополнительные сведения о файле triage.ini и примере **расширения !owner** см. в разделе "Указание владельцев модулей и функций".

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Peb

Статья • 05.04.2024

Расширение **!peb** отображает форматируемое представление сведений в блоке среды процесса (PEB).

dbgcmd

```
!peb [PEB-Address]
```

## Параметры

*PEB-Address*
Шестнадцатеричный адрес процесса, peB которого требуется проверить. (Это не адрес PEB, производный от блока процесса ядра для процесса.) Если *PEB-Address* опущен в пользовательском режиме, используется PEB для текущего процесса. Если он опущен в режиме ядра, отображается PEB, соответствующий текущему контексту процесса.

## DLL-библиотеки

Exts.dll

## Дополнительная информация

Сведения о блоках *среды обработки см. в разделе* "Внутренние компоненты Microsoft Windows" МаркОм Руссиновичем и Дэвидом Соломоном.

## Замечания

PEB — это часть структур управления процессами Microsoft Windows в пользовательском режиме.

**Если расширение !peb** без аргумента дает ошибку в режиме ядра, следует использовать **расширение !process** для определения адреса PEB для требуемого процесса. Убедитесь, что контекст процесса задан для требуемого процесса, а затем используйте PEB-адрес в качестве аргумента для **!peb**.

Точные выходные данные зависят от версии Windows и от того, выполняется ли отладка в режиме ядра или пользовательском режиме. Следующий пример взят из отладчика ядра, подключенного к целевому объекту Windows Server 2003:

dbgcmd

```
kd> !peb
PEB at 7ffdf000
    InheritedAddressSpace:    No
    ReadImageFileExecOptions: No
    BeingDebugged:            No
    ImageBaseAddress:         4ad00000
    Ldr                       77fbe900
    Ldr.Initialized:          Yes
    Ldr.InInitializationOrderModuleList: 00241ef8 . 00242360
    Ldr.InLoadOrderModuleList:           00241e90 . 00242350
    Ldr.InMemoryOrderModuleList:         00241e98 . 00242358
            Base TimeStamp                     Module
        4ad00000 3d34633c Jul 16 11:17:32 2002 D:\WINDOWS\system32\cmd.exe
        77f40000 3d346214 Jul 16 11:12:36 2002 D:\WINDOWS\system32\ntdll.dll
        77e50000 3d3484ef Jul 16 13:41:19 2002
D:\WINDOWS\system32\kernel32.dll
....
    SubSystemData:     00000000
    ProcessHeap:       00140000
    ProcessParameters: 00020000
    WindowTitle: "'D:\Documents and
Settings\Administrator\Desktop\Debuggers.lnk'"
    ImageFile:     'D:\WINDOWS\system32\cmd.exe'
    CommandLine:   '"D:\WINDOWS\system32\cmd.exe" '
    DllPath:       'D:\WINDOWS\system32;D:\WINDOWS\system32;....
    Environment:  00010000
        ALLUSERSPROFILE=D:\Documents and Settings\All Users
        APPDATA=D:\Documents and Settings\UserTwo\Application Data
        CLIENTNAME=Console
....
        windir=D:\WINDOWS
```

**Аналогичное расширение !teb** отображает блок среды потока.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Rebase

Статья • 05.04.2024

Расширение **!rebase** выполняет поиск в файле rebase.log для указанного адреса или символа.

dbgcmd

```
!rebase [-r] Address [Path]
!rebase Symbol [Path]
!rebase -stack [Path]
!rebase -?
```

## Параметры

**-r**
Пытается загрузить любой модуль, найденный в rebase.log.

*Адрес*
Задает адрес в стандартном шестнадцатеричном формате. Расширение будет искать библиотеки DLL рядом с этим адресом.

*Путь*
Указывает путь к файлу rebase.log. Если *путь* не указан, расширение пытается угадать путь к rebase.log или, завершив ошибку, пытается считывать файл rebase.log из текущего рабочего каталога.

*Символ*
Указывает имя символа или изображения. Расширение будет искать библиотеки DLL, содержащие эту подстроку.

**-Стека**
Отображает все модули в текущем стеке.

**-?**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Ext.dll

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rtlavl

Статья • 05.04.2024

Расширение **!rtlavl** отображает записи структуры RTL_AVL_TABLE.

```
dbgcmd
```

```
!rtlavl Address [Module!Type]
!rtlavl -?
```

## Параметры

*Адрес*
Указывает адрес отображаемого RTL_AVL_TABLE.

*Модуль*
Указывает модуль, в котором определена структура данных.

*Тип*
Задает имя структуры данных.

**-?**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Ext.dll

## Дополнительная информация

**Используйте расширение !gentable** для отображения таблиц AVL.

## Замечания

*Включая модуль!Параметр типа* приводит к интерпретации каждой записи в таблице с заданным типом.

Отображение может быть прервано в любое время, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD или CDB).

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Sd

Статья • 05.04.2024

Расширение **!sd** отображает дескриптор безопасности по указанному адресу.

Синтаксис

```
dbgcmd
```

```
!sd Address [Flags]
```

## Параметры

*Адрес*
Задает шестнадцатеричный адрес структуры SECURITY_DESCRIPTOR.

*Flags*
Если задано значение 1, отображается понятное имя. Сюда входят тип идентификатора безопасности (SID), а также домен и имя пользователя для идентификатора безопасности.

## DLL-библиотеки

Exts.dll

## Дополнительная информация

Пример приложения и примера этой команды см. в разделе "Определение ACL объекта". Дополнительные сведения о дескрипторах безопасности см. в документации по пакету SDK для Microsoft Windows, документации по комплекту драйверов Windows (WDK) и *внутренним* службам Microsoft Windows марком Руссиновичем и Дэвидом Соломоном. Также см. статью !sid **и !acl.**

## Замечания

Рассмотрим пример:

```
dbgcmd
```

```
kd> !sd e1a96a80 1
->Revision: 0x1
->Sbz1    : 0x0
->Control : 0x8004
            SE_DACL_PRESENT
            SE_SELF_RELATIVE
->Owner   : S-1-5-21-518066528-515770016-299552555-2981724 (User:
MYDOMAIN\myuser)
->Group   : S-1-5-21-518066528-515770016-299552555-513 (Group:
MYDOMAIN\Domain Users)
->Dacl    :
->Dacl    : ->AclRevision: 0x2
->Dacl    : ->Sbz1       : 0x0
->Dacl    : ->AclSize    : 0x40
->Dacl    : ->AceCount   : 0x2
->Dacl    : ->Sbz2       : 0x0
->Dacl    : ->Ace[0]: ->AceType: ACCESS_ALLOWED_ACE_TYPE
->Dacl    : ->Ace[0]: ->AceFlags: 0x0
->Dacl    : ->Ace[0]: ->AceSize: 0x24
->Dacl    : ->Ace[0]: ->Mask : 0x001f0003
->Dacl    : ->Ace[0]: ->SID: S-1-5-21-518066528-515770016-299552555-2981724
(User: MYDOMAIN\myuser)

->Dacl    : ->Ace[1]: ->AceType: ACCESS_ALLOWED_ACE_TYPE
->Dacl    : ->Ace[1]: ->AceFlags: 0x0
->Dacl    : ->Ace[1]: ->AceSize: 0x14
->Dacl    : ->Ace[1]: ->Mask : 0x001f0003
->Dacl    : ->Ace[1]: ->SID: S-1-5-18 (Well Known Group: NT
AUTHORITY\SYSTEM)

->Sacl    :  is NULL
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Sid

Статья • 05.04.2024

Расширение **!sid** отображает идентификатор безопасности (SID) по указанному адресу.

Синтаксис

```
dbgcmd
```

```
!sid Address [Flags]
```

## Параметры

*Адрес*
Указывает адрес структуры SID.

*Flags*
Если для этого параметра задано значение 1, отображается тип безопасности, домен и имя пользователя для идентификатора безопасности.

Если задано значение 1, отображается понятное имя. К ним относятся тип БЕЗОПАСНОСТИ, а также домен и имя пользователя для идентификатора безопасности.

## DLL-библиотеки

Exts.dll

## Дополнительная информация

Сведения о идентификаторах SID см. в документации по пакету SDK Для Microsoft Windows, документации по комплекту драйверов Windows (WDK) или *внутренним* устройствам Microsoft Windows марком Руссиновичем и Дэвидом Соломоном. Кроме того, см. статью !sd **и !acl**.

## Замечания

Ниже приведены два примера: один без понятного имени и один с:

dbgcmd

```
kd> !sid 0xe1bf35b8
SID is: S-1-5-21-518066528-515770016-299552555-513

kd> !sid 0xe1bf35b8 1
SID is: S-1-5-21-518066528-515770016-299552555-513 (Group: MYGROUP\Domain
Users)
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !slist

Статья • 05.04.2024

Расширение **!slist** отображает singly связанный список (SList).

```
dbgcmd
```

```
!slist Address [ Symbol [Offset] ]
!slist -?
```

## Параметры

*Адрес*
Задает адрес SLIST_HEADER.

*Символ*
Указывает тип данных, используемый для отображения. Если *задан символ* , отладчик предполагает, что каждый узел SList является экземпляром этого типа данных при отображении.

*Offset*
Указывает смещение байтов указателя SList в структуре.

**-?**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Exts.dll

## Замечания

Если вы знаете характер связанных структур, *параметры символа* и *смещения* очень полезны. Чтобы увидеть разницу, ниже приведены два примера. первый окупит *параметры символа* и *смещения* , а второй включает их.

```
dbgcmd
```

```
0:000> !slist ListHead
SLIST HEADER:
```

```
   +0x000 Alignment          : a000a002643e8
   +0x000 Next               : 2643e8
   +0x004 Depth              : a
   +0x006 Sequence           : a

SLIST CONTENTS:
002643e8  002642c0 0000000a 6e676953 72757461
002642c0  00264198 00000009 6e676953 72757461
00264198  00264070 00000008 6e676953 72757461
00264070  00263f48 00000007 6e676953 72757461
00263f48  00261420 00000006 6e676953 72757461
00261420  002612f8 00000005 6e676953 72757461
002612f8  002611d0 00000004 6e676953 72757461
002611d0  002610a8 00000003 6e676953 72757461
002610a8  00260f80 00000002 6e676953 72757461
00260f80  00000000 00000001 6e676953 72757461
```

dbgcmd

```
0:000> !slist ListHead _PROGRAM_ITEM 0
SLIST HEADER:
 +0x000 Alignment          : a000a002643e8
   +0x000 Next               : 2643e8
 +0x004 Depth              : a
   +0x006 Sequence           : a

SLIST CONTENTS:
002643e8
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 0xa
   +0x008 Description      : [260]  "Signature is: 10"
002642c0
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 9
   +0x008 Description      : [260]  "Signature is: 9"
00264198
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 8
   +0x008 Description      : [260]  "Signature is: 8"
00264070
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 7
   +0x008 Description      : [260]  "Signature is: 7"
00263f48
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 6
   +0x008 Description      : [260]  "Signature is: 6"
00261420
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 5
   +0x008 Description      : [260]  "Signature is: 5"
002612f8
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
```

```
   +0x004 Signature        : 4
   +0x008 Description      : [260]  "Signature is: 4"
002611d0
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 3
   +0x008 Description      : [260]  "Signature is: 3"
002610a8
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 2
   +0x008 Description      : [260]  "Signature is: 2"
00260f80
   +0x000 ItemEntry        : _SINGLE_LIST_ENTRY
   +0x004 Signature        : 1
   +0x008 Description      : [260]  "Signature is: 1"
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !std_map

Статья • 01.04.2024

Команда расширения !std_map устарела и больше не поддерживается.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !stl

Статья • 22.10.2024

Команда расширения !stl устарела и больше не поддерживается. Используйте модель данных отладчика,
Команда dx (выражение объектной модели отладчика) вместо этого.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ул

Статья • 05.04.2024

**Расширение !str** устарело и больше не поддерживается. Используйте модель данных отладчика,
Команда dx (выражение объектной модели отладчика) вместо этого.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !sym

Статья • 08.07.2024

**Расширение !sym** управляет шумной загрузкой символов и запросами символов.

```
dbgcmd
```

```
!sym
!sym noisy
!sym quiet
!sym prompts
!sym prompts off
```

# Параметры

**шумный**
Активирует загрузку шумных символов.

**тихий**
Деактивирует загрузку шумных символов.

**Запросы**
Позволяет отображать диалоговые окна проверки подлинности, когда SymSrv получает запрос проверки подлинности.

**запросы отключены**
Подавляет все диалоговые окна проверки подлинности, когда SymSrv получает запрос проверки подлинности. Это может привести к тому, что SymSrv не может получить доступ к символам через Интернет.

# DLL-библиотеки

Dbghelp.dll

# Замечания

**Если расширение !sym** используется без аргументов, отображается текущее состояние загрузки символов и запроса символов.

**!sym шумные** и **!sym тихие** расширения управляют шумной загрузкой символов. Дополнительные сведения и другие методы отображения и изменения этого

параметра см. в SYMOPT_DEBUG.

**Запросы !sym и !sym запрашивают** расширения, определяют, отображаются ли диалоги проверки подлинности при обнаружении запроса проверки подлинности SymSrv. Эти команды должны следовать **повторной загрузке (модуль перезагрузки),** чтобы они вошли в силу. Запросы проверки подлинности могут отправляться прокси-серверами, брандмауэрами Интернета, средствами чтения смарт-карт и безопасными веб-сайтами. Дополнительные сведения и другие методы изменения этого параметра см. в разделе "Брандмауэры" и "Прокси-серверы".

**Обратите внимание, что** загрузка символов Noisy не должна быть путана с шумной загрузкой источника, которая управляется **командой Srcnoisy (шумная загрузка источника).**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !symsrv

Статья • 01.04.2024

Команда расширения !symsrv устарела и больше не поддерживается.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !teb

Статья • 05.04.2024

Расширение **!teb** отображает форматируемое представление сведений в блоке среды потока (TEB).

```
dbgcmd

!teb [TEB-Address]
```

## \Parameters

*TEB-Address*
Шестнадцатеричный адрес потока, чей TEB вы хотите проверить. (Это не адрес TEB, производный от блока потока ядра для потока.) Если *TEB-Address* опущен в пользовательском режиме, используется TEB для текущего потока. Если он опущен в режиме ядра, отображается TEB, соответствующий текущему контексту регистра.

## DLL-библиотеки

Exts.dll

## Дополнительная информация

Сведения о блоках *среды потоков см. в разделе* "Внутренние компоненты Microsoft Windows" марком Руссиновичем и Дэвидом Соломоном.

## Замечания

TEB — это часть структур управления потоками Microsoft Windows в пользовательском режиме.

**Если расширение !teb** без аргумента дает ошибку в режиме ядра, следует использовать **расширение !process** для определения адреса TEB для требуемого потока. Убедитесь, что контекст регистрации задан в нужном потоке, а затем используйте адрес TEB в качестве аргумента для **!teb**.

Ниже приведен пример выходных данных этой команды в пользовательском режиме:

dbgcmd

```
0:001> ~
   0  id: 324.458   Suspend: 1 Teb 7ffde000 Unfrozen
.  1  id: 324.48c   Suspend: 1 Teb 7ffdd000 Unfrozen

0:001> !teb
TEB at 7FFDD000
    ExceptionList:    76ffdc
    Stack Base:       770000
    Stack Limit:      76f000
    SubSystemTib:     0
 FiberData:        1e00
    ArbitraryUser:    0
    Self:             7ffdd000
    EnvironmentPtr:   0
 ClientId:         324.48c
    Real ClientId:    324.48c
    RpcHandle:        0
    Tls Storage:      0
    PEB Address:      7ffdf000
    LastErrorValue:   0
    LastStatusValue:  0
    Count Owned Locks:0
    HardErrorsMode:   0
```

**Аналогичное расширение !peb** отображает блок среды процесса.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Tls

Статья • 05.04.2024

Расширение **!tls** отображает слот локального хранилища потока (TLS).

```
dbgcmd
```

```
!tls Slot [TEB]
```

## Параметры

*Слот*
Указывает слот TLS. Это может быть любое значение от 0 до 1088 (десятичное). Если *слот* равен -1, отображаются все слоты.

*TEB*
Указывает блок среды потока (TEB). Если это значение равно 0 или опущено, используется текущий поток.

## DLL-библиотеки

Exts.dll

## Замечания

Рассмотрим пример:

```
dbgcmd
```

```
0:000> !tls -1
TLS slots on thread: c08.f54
0x0000 : 00000000
0x0001 : 003967b8
0:000> !tls 0
c08.f54: 00000000
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Маркер

Статья • 05.04.2024

Расширение **!token** отображает форматное представление объекта маркера безопасности.

Синтаксис режима ядра:

```
dbgcmd

!token [-n] [Address]
!token -?
```

Синтаксис пользовательского режима:

```
dbgcmd

!token [-n] [Handle]
!token -?
```

## Параметры

*Адрес*
(только режим ядра) Указывает адрес отображаемого маркера. Если это значение равно 0 или опущено, отображается маркер для активного потока.

*Дескриптор*
(Только в пользовательском режиме) Указывает дескриптор отображаемого маркера. Если это значение равно 0 или опущено, отображается маркер, связанный с целевым потоком.

**-n**
(Только в пользовательском режиме) Приводит к тому, что отображение будет содержать понятное имя. Сюда входят тип идентификатора безопасности (SID), а также домен и имя пользователя для идентификатора безопасности. Этот параметр нельзя использовать при отладке LSASS.

**-?**
Отображает текст справки для этого расширения.

## DLL-библиотеки

Exts.dll

Команда **!token** доступна в режиме ядра и динамической отладке пользовательского режима. Его нельзя использовать в файлах дампа в пользовательском режиме.

# Дополнительная информация

Сведения о структуре токена в режиме ядра см. в разделе "*Внутренние* компоненты Microsoft Windows" Марком Руссиновичем и Дэвидом Соломоном. Сведения о структуре TOKENа в пользовательском режиме см. в документации по пакету SDK для Microsoft Windows.

# Замечания

Структура TOKEN — это тип объекта безопасности, представляющий прошедший проверку подлинности пользовательский процесс. Каждый процесс имеет назначенный маркер, который становится маркером по умолчанию для каждого потока этого процесса. Однако отдельный поток можно назначить маркер, который переопределяет этот параметр по умолчанию.

Адрес маркера можно получить из выходных **данных !process**. Чтобы отобразить список отдельных полей структуры TOKEN, используйте **команду dt nt!_TOKEN** .

Рассмотрим пример:

dbgcmd

```
kd> !process 81464da8 1
PROCESS 81464da8  SessionId: 0  Cid: 03bc    Peb: 7ffdf000  ParentCid: 0124
    DirBase: 0dec2000  ObjectTable: e1a31198  TableSize: 275.
    Image: MSMSGS.EXE
    VadRoot 81468cc0 Vads 170 Clone 0 Private 455. Modified 413. Locked 0.
    DeviceMap e1958438
    Token                             e1bed030
    ElapsedTime                       0:44:15.0142
    UserTime                          0:00:00.0290
    KernelTime                        0:00:00.0300
    QuotaPoolUsage[PagedPool]         49552
    QuotaPoolUsage[NonPagedPool]      10872
    Working Set Sizes (now,min,max)  (781, 50, 345) (3124KB, 200KB, 1380KB)
    PeakWorkingSetSize                1550
    VirtualSize                       57 Mb
    PeakVirtualSize                   57 Mb
    PageFaultCount                    2481
    MemoryPriority                    BACKGROUND
```

```
    BasePriority                      8
    CommitCharge                      2497
kd> !exts.token -n e1bed030
_TOKEN e1bed030
TS Session ID: 0
User: S-1-5-21-518066528-515770016-299552555-2981724 (User: MYDOMAIN\myuser)
Groups:
 00 S-1-5-21-518066528-515770016-299552555-513 (Group: MYDOMAIN\Domain
Users)
    Attributes - Mandatory Default Enabled
 01 S-1-1-0 (Well Known Group: localhost\Everyone)
    Attributes - Mandatory Default Enabled
 02 S-1-5-32-544 (Alias: BUILTIN\Administrators)
    Attributes - Mandatory Default Enabled Owner
 03 S-1-5-32-545 (Alias: BUILTIN\Users)
    Attributes - Mandatory Default Enabled
 04 S-1-5-21-518066528-515770016-299552555-2999049 (Group:
MYDOMAIN\AllUsers)
    Attributes - Mandatory Default Enabled
 05 S-1-5-21-518066528-515770016-299552555-2931095 (Group:
MYDOMAIN\SomeGroup1)
    Attributes - Mandatory Default Enabled
 06 S-1-5-21-518066528-515770016-299552555-2931096 (Group:
MYDOMAIN\SomeGroup2)
    Attributes - Mandatory Default Enabled
 07 S-1-5-21-518066528-515770016-299552555-3014318 (Group:
MYDOMAIN\SomeGroup3)
    Attributes - Mandatory Default Enabled
 08 S-1-5-21-518066528-515770016-299552555-3053352 (Group: MYDOMAIN\Another
Group)
    Attributes - Mandatory Default Enabled
 09 S-1-5-21-518066528-515770016-299552555-2966661 (Group:
MYDOMAIN\TestGroup)
    Attributes - Mandatory Default Enabled
 10 S-1-5-21-2117033040-537160606-1609722162-17637 (Group:
MYOTHERDOMAIN\someusers)
    Attributes - Mandatory Default Enabled
 11 S-1-5-21-518066528-515770016-299552555-3018354 (Group:
MYDOMAIN\TestGroup2)
    Attributes - Mandatory Default Enabled
 12 S-1-5-21-518066528-515770016-299552555-3026602 (Group:
MYDOMAIN\SomeGroup4)
    Attributes - Mandatory Default Enabled
 13 S-1-5-21-518066528-515770016-299552555-2926570 (Group:
MYDOMAIN\YetAnotherGroup)
    Attributes - Mandatory Default Enabled
 14 S-1-5-21-661411660-2927047998-133698966-513 (Group: MYDOMAIN\Domain
Users)
    Attributes - Mandatory Default Enabled
 15 S-1-5-21-518066528-515770016-299552555-2986081 (Alias:
MYDOMAIN\an_alias)
    Attributes - Mandatory Default Enabled GroupResource
 16 S-1-5-21-518066528-515770016-299552555-3037986 (Alias:
MYDOMAIN\AReallyLongGroupName1)
    Attributes - Mandatory Default Enabled GroupResource
```

```
 17 S-1-5-21-518066528-515770016-299552555-3038991 (Alias:
MYDOMAIN\AReallyLongGroupName2)
    Attributes - Mandatory Default Enabled GroupResource
 18 S-1-5-21-518066528-515770016-299552555-3037999 (Alias:
MYDOMAIN\AReallyLongGroupName3)
    Attributes - Mandatory Default Enabled GroupResource
 19 S-1-5-21-518066528-515770016-299552555-3038983 (Alias:
MYDOMAIN\AReallyReallyLongGroupName)
    Attributes - Mandatory Default Enabled GroupResource
 20 S-1-5-5-0-71188 (no name mapped)
    Attributes - Mandatory Default Enabled LogonId
 21 S-1-2-0 (Well Known Group: localhost\LOCAL)
    Attributes - Mandatory Default Enabled
 22 S-1-5-4 (Well Known Group: NT AUTHORITY\INTERACTIVE)
    Attributes - Mandatory Default Enabled
 23 S-1-5-11 (Well Known Group: NT AUTHORITY\Authenticated Users)
    Attributes - Mandatory Default Enabled
Primary Group: S-1-5-21-518066528-515770016-299552555-513 (Group:
MYDOMAIN\Domain Users)
Privs:
 00 0x000000017 SeChangeNotifyPrivilege           Attributes - Enabled
Default
 01 0x000000008 SeSecurityPrivilege               Attributes -
 02 0x000000011 SeBackupPrivilege                 Attributes -
 03 0x000000012 SeRestorePrivilege                Attributes -
 04 0x00000000c SeSystemtimePrivilege             Attributes -
 05 0x000000013 SeShutdownPrivilege               Attributes -
 06 0x000000018 SeRemoteShutdownPrivilege         Attributes -
 07 0x000000009 SeTakeOwnershipPrivilege          Attributes -
 08 0x000000014 SeDebugPrivilege                  Attributes -
 09 0x000000016 SeSystemEnvironmentPrivilege      Attributes -
 10 0x00000000b SeSystemProfilePrivilege          Attributes -
 11 0x00000000d SeProfileSingleProcessPrivilege   Attributes -
 12 0x00000000e SeIncreaseBasePriorityPrivilege   Attributes -
 13 0x00000000a SeLoadDriverPrivilege             Attributes - Enabled
 14 0x00000000f SeCreatePagefilePrivilege         Attributes -
 15 0x000000005 SeIncreaseQuotaPrivilege          Attributes -
 16 0x000000019 SeUndockPrivilege                 Attributes - Enabled
 17 0x00000001c SeManageVolumePrivilege           Attributes -
Authentication ID:         (0,11691)
Impersonation Level:       Anonymous
TokenType:                 Primary
Source: User32             TokenFlags: 0x9 ( Token in use )
Token ID: 18296            ParentToken ID: 0
Modified ID:               (0, 18298)
RestrictedSidCount: 0      RestrictedSids: 00000000
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !tp

Статья • 08.07.2024

Расширение **!tp** отображает сведения о пуле потоков.

dbgcmd

```
!tp pool Address [Flags]
!tp tqueue Address [Flags]
!tp ItemType Address [Flags]
!tp ThreadType [Address]
!tp stats Address [Flags]
!tp wfac Address
!tp wqueue Address Priority Node
!tp -?
```

# Параметры

**бассейн** **** *Адрес*
Приводит к отображению всего пула потоков в *адресе* . Если *адрес* равен 0, будут отображаться все пулы потоков.

**tqueue** **** *Адрес*
Приводит к отображению активной очереди таймера на *адресе* .

*Адрес ItemType*
Приводит к отображению указанного элемента пула потоков. *Адрес* указывает адрес элемента. *ItemType* указывает тип элемента. Это может включать любые из следующих возможностей:

**obj**
Отобразится универсальный элемент пула (например, элемент ввода-вывода).

**таймер**
Отобразится элемент таймера.

**ждать**
Появится элемент ожидания.

**работа**
Откроется рабочий элемент.

*ThreadType* [*Адрес*]
Приводит к отображению потоков указанного типа. Если *адрес* включен и

ненулево, отображается только поток по этому адресу. Если *адрес* равен 0, отображаются все потоки, *соответствующие ThreadType* . Если *адрес* опущен, отображаются только потоки, соответствующие *ThreadType* , связанному с текущим потоком. *ThreadType* указывает тип отображаемого потока. Это может включать любой из следующих возможностей:

**официант**
Будет отображаться поток ожидания пула потоков.

**рабочая роль**
Будет отображаться рабочий поток пула потоков.

**статистика** [*адрес*]
Приводит к отображению статистики отладки текущего потока. *Адрес* может быть опущен, но если он указан, он должен иметь значение -1 (отрицательное значение), чтобы представить текущий поток.

**wfac** **** *Адрес*
(Только Windows 7 и более поздних версий) Приводит к отображению рабочей фабрики по *адресу* . Указанный *адрес* должен быть допустимым ненулевом адресом.

**wqueue** **** *Адрес*
(Только Windows 7 и более поздних версий) Вызывает отображение рабочей очереди и узла NUMA, соответствующего следующему: указанному приоритету, указанному узлу NUMA и пулу по указанному адресу, которому принадлежит узел NUMA. *Адрес* указывает адрес пула. **Если используется параметр wqueue**, за ним следует *адрес*, *приоритет* и *узел*.

*Приоритет*
(Только Windows 7 и более поздних версий) Указывает уровни приоритета отображаемых рабочих очередей. *Приоритет* может быть любым из следующих значений:

**0**
Отображаются рабочие очереди с высоким приоритетом.

**1**
Отображаются рабочие очереди с обычным приоритетом.

**2**
Отображаются рабочие очереди с низким приоритетом.

**-1**
Отображаются все рабочие очереди.

*Узел*
(Только Windows 7 и более поздних версий) Указывает узел NUMA, принадлежащий пулу, указанному адресом. Если *узел* равен -1 (отрицательный), отображаются все узлы NUMA.

*Флаги*
Указывает, что должен содержать дисплей. Это может быть сумма любого из следующих битовых значений (значение по умолчанию — 0x0):

Bit 0 (0x1)
Вызывает отображение однострочного вывода. Это битовое значение не влияет на выходные данные при *отображении ItemType* .

Бит 1 (0x2)
Приводит к включению сведений о членах.

Бит 2 (0x4)
Этот флаг имеет значение только в том случае, если используется параметр пула. В Windows XP, Windows Server 2003, Windows Vista и Windows Server 2008 этот флаг приводит к включению рабочей очереди пула. В Windows 7 и более поздних версиях этот флаг приводит к включению всех рабочих очередей пула, которые находятся в обычном приоритете, и всех узлов NUMA.

**-?**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Exts.dll

# Дополнительная информация

Дополнительные сведения о пуле потоков см. в документации по пакету SDK для Microsoft Windows.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Сортировки

Статья • 01.04.2024

Команда **расширения !triage** устарела. Вместо этого используйте **!analyze** .

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ustr

Статья • 05.04.2024

Расширение **!ustr** устарело и больше не поддерживается. Используйте модель данных отладчика,
Команда dx (выражение объектной модели отладчика) вместо этого.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Версия

Статья • 05.04.2024

Расширение **!version отображает сведения о версии** библиотеки DLL расширения.

Эта команда расширения не должна путаться с командой **версии (показать версию отладчика).**

dbgcmd

```
![ExtensionDLL.]version
```

## Параметры

*ExtensionDLL*
Указывает библиотеку DLL расширения, номер версии которой должен отображаться.

## DLL-библиотеки

Это расширение доступно в большинстве библиотек DLL расширений.

## Замечания

Если версия DLL расширения не соответствует версии отладчика, будут отображаться сообщения об ошибках.

Эта команда расширения не будет работать в Windows XP и более поздних версиях Windows. Чтобы отобразить сведения о версии, используйте **команду "Показать версию отладчика".**

Первоначальное назначение этого расширения заключается в том, чтобы версия DLL соответствовала целевой версии, так как несоответствие приведет к неточным результатам для многих расширений. Новые библиотеки DLL больше не ограничиваются работой только с одной версией Windows, поэтому это расширение устарело.

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !winrterr

Статья • 05.04.2024

Расширение **!winrterr** задает режим отчетов отладчика для среда выполнения Windows ошибок.

dbgcmd

```
!winrterr Mode
!winrterr
```

## Параметры

*Режим*
В следующей таблице описываются возможные значения режима.

Развернуть таблицу

| значение | Описание |
|---|---|
| отчет | При возникновении ошибки среда выполнения Windows в отладчике отображаются ошибки и связанный текст, но выполняется перенастройка выполнения. Это режим по умолчанию. |
| break | При возникновении ошибки среда выполнения Windows в отладчике отображается ошибка и связанный текст, а выполнение останавливается. |
| quiet | При возникновении ошибки среда выполнения Windows ничего не отображается в отладчике, а выполнение продолжается. |

Если *режим* опущен, **функция !winrterr** отображает текущий режим отчетности. Если отладчик сломался в результате ошибки среда выполнения Windows, отображается также сообщение об ошибке и связанном тексте.

## См. также

команды отладчика среда выполнения Windows

**!Hstring**

**!hstring2**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Kernel-Mode Extensions

18.07.2025

В этом разделе ссылки описываются команды расширения, которые в основном используются во время отладки в режиме ядра.

Отладчик автоматически загружает правильную версию этих команд расширения. Если вы вручную не загружали другую версию, вам не нужно отслеживать используемые версии DLL. Сведения об описании порядка поиска модуля по умолчанию см. в командах расширения отладчика . Сведения о том, как загружать модули расширения, см. в статье о загрузке dll расширений отладчика .

На каждой странице ссылки на команду расширения перечислены библиотеки DLL, предоставляющие следующую команду. Используйте следующие правила, чтобы определить правильный каталог, из которого необходимо загрузить библиотеку DLL этого расширения:

- Если целевой компьютер работает под управлением Windows XP или более поздней версии Windows, используйте winxp\Kdexts.dll.

Кроме того, расширения в режиме ядра, которые не относятся к одной операционной системе, можно найти в winext\kext.dll.

# !ahcache

Статья • 08.07.2024

Расширение **!ahcache** отображает кэш совместимости приложений.

```
dbgcmd
```

```
!ahcache [Flags]
```

## Параметры

*Флаги*
Указывает сведения, которые необходимо включить в дисплей. Это может быть любое сочетание следующих битов (значение по умолчанию равно нулю):

Bit 0 (0x1)
Отображает список RTL_GENERIC_TABLE вместо списка LRU.

Бит 4 (0x10)
Подробное отображение: включает все сведения о записи, а не только имена.

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !alignmentfaults

Статья • 07.02.2024

Расширение **!alignmentfaults** отображает все текущие ошибки выравнивания типов по расположению и изображению, отсортированные по частоте.

dbgcmd

```
!alignmentfaults
```

# DLL

Kdexts.dll

## Дополнительные сведения

Сведения об ошибках выравнивания см. в документации по Microsoft Windows SDK.

# Комментарии

Эта возможность доступна только в более ранних версиях Windows до Windows 10 версии 1803, в которых были предоставлены проверенные сборки.

# !analyzebug проверка

Статья • 05.04.2024

Команда **расширения !analyzebug проверка** устарела. Вместо этого используйте **!analyze** .

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Apc

Статья • 05.04.2024

Форматы **расширений !apc** и отображение содержимого одного или нескольких асинхронных вызовов процедур (APCs).

dbgcmd

```
!apc
!apc proc Process
!apc thre Thread
!apc KAPC
```

## Параметры

*Обработать*
Указывает адрес процесса, api-интерфейсы которого должны отображаться.

*Поток*
Указывает адрес потока, чьи API-интерфейсы должны отображаться.

*КАПК*
Указывает адрес отображаемого APC ядра.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения об API см. в документации по комплекту драйверов Windows (WDK) и внутренних компонентах Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Без параметров **!apc** отображает все API.

Рассмотрим пример:

Консоль

```
kd> !apc
*** Enumerating APCs in all processes
Process e0000000858ba8b0 System
Process e0000165fff86040 smss.exe
Process e0000165fff8c040 csrss.exe
Process e0000165fff4e1d0 winlogon.exe
Process e0000165fff101d0 services.exe
Process e0000165fffa81d0 lsass.exe
Process e0000165fff201d0 svchost.exe
Process e0000165fff8e040 svchost.exe
Process e0000165fff3e040 svchost.exe
Process e0000165fff6e040 svchost.exe
Process e0000165fff24040 spoolsv.exe
Process e000000085666640 wmiprvse.exe
Process e00000008501e520 wmiprvse.exe
Process e0000000856db480 explorer.exe
Process e0000165fff206a0 ctfmon.exe
Process e0000000850009d0 ctfmon.exe
Process e0000165fff51600 conime.exe
Process e000000085496340 taskmgr.exe
Process e000000085489c30 userinit.exe
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !apicerr

Статья • 05.04.2024

Расширение **!apicerr** отображает локальный журнал ошибок расширенного программируемого контроллера прерываний (APIC).

dbgcmd

```
!apicerr [Format]
```

## Параметры

*Формат*
Указывает порядок отображения содержимого журнала ошибок. Это может быть любое из следующих значений:

0x0
Отображает журнал ошибок в соответствии с порядком вхождения.

0x1
Отображает журнал ошибок в соответствии с обработчиком.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x86 или на основе x64.

## Дополнительная информация

Сведения об API-интерфейсах *см. в разделе* "Внутренние компоненты Microsoft Windows" марком Руссиновичем и Дэвидом Соломоном.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !arbinst

Статья • 05.04.2024

Расширение **!arbinst** отображает сведения о указанном arbiter.

dbgcmd

```
    !arbinst Address [Flags]
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес отображаемого arbiter.

*Flags*
Указывает, сколько сведений отображается для каждого arbiter. В настоящее время единственным флагом является 0x100. Если этот флаг задан, отображаются псевдонимы.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. также расширение !arbiter.

## Замечания

Для указанного **arbiter** **!arbinst** отображает каждый выделенный диапазон системных ресурсов, некоторые необязательные флаги, PDO, присоединенные к этому диапазону (другими словами, владелец диапазона), и имя службы этого владельца (если известно).

Рассмотрим пример:

Консоль

```
kd> !arbinst e0000106002ee8e8
Port Arbiter "PCI I/O Port (b=02)" at e0000106002ee8e8
```

```
  Allocated ranges:
    0000000000000000 - 0000000000001fff       00000000 <Not on bus>
    0000000000002000 - 00000000000020ff     P e0000000858bea20  (ql1280)
    0000000000003000 - ffffffffffffffff       00000000 <Not on bus>
  Possible allocation:
    < none >
kd> !arbinst e0000106002ec458
Memory Arbiter "PCI Memory (b=02)" at e0000106002ec458
  Allocated ranges:
    0000000000000000 - 00000000ebffffff       00000000 <Not on bus>
    00000000effdef00 - 00000000effdefff   B   e0000000858be560
    00000000effdf000 - 00000000effdffff       e0000000858bea20  (ql1280)
    00000000f0000000 - ffffffffffffffff       00000000 <Not on bus>
  Possible allocation:
    < none >
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Арбитр

Статья • 05.04.2024

Расширение **!arbiter** отображает текущие системные arbiters и диапазоны арбитража.

```
dbgcmd

    !arbiter [Flags]
```

## Параметры

*Flags*
Указывает, какие классы арбитеров отображаются. Если опущено, отображаются все arbiters. Эти биты можно объединять свободно.

Bit 0 (0x1)
Отображение arbiters ввода-вывода.

Бит 1 (0x2)
Отображение arbiters памяти.

Бит 2 (0x4)
Отображение arbiters IRQ.

Бит 3 (0x8)
Отображение arbiters DMA.

Бит 4 (0x10)
Отображение arbiters номера шины.

Бит 8 (0x100)
Не отображайте псевдонимы.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения.

# Замечания

Для каждого arbiter **!arbiter** отображает каждый выделенный диапазон системных ресурсов, некоторые необязательные флаги, PDO, присоединенные к этому диапазону (другими словами, владелец диапазона) и имя службы этого владельца (если известно).

Флаги имеют следующие значения:

Развернуть таблицу

| Флаг | Значение |
|---|---|
| S | Диапазон общий |
| O | Диапазон в конфликте |
| Б | Диапазон выделен загрузочным |
| D | Диапазон является драйвером эксклюзивным |
| a | Псевдоним диапазона |
| P | Положительное декодирование диапазона |

Рассмотрим пример:

Консоль

```
kd> !arbiter 4

DEVNODE 80e203b8 (HTREE\ROOT\0)
  Interrupt Arbiter "" at 80167140
    Allocated ranges:
      0000000000000000 - 0000000000000000   B   80e1d3d8
      0000000000000001 - 0000000000000001   B   80e1d3d8
 .....
      00000000000001a2 - 00000000000001a2
        00000000000001a2 - 00000000000001a2  CB   80e1d3d8
        00000000000001a2 - 00000000000001a2  CB   80e52538  (Serial)
      00000000000001a3 - 00000000000001a3       80e52778  (i8042prt)
      00000000000001b3 - 00000000000001b3       80e1b618  (i8042prt)
 Possible allocation:
      < none >
```

В этом примере в следующей строке показан диапазон ресурсов (который состоит из 0x1A3 только), PDO 0x80E52778 и службы i8042prt.sys. В этой строке нет флагов.

Теперь вы можете использовать **!devobj** с этим адресом PDO, чтобы найти адреса расширения устройства и узла устройства:

Консоль

```
kd> !devobj 80e52778
Device object (80e52778) is for:
 00000034 \Driver\PnpManager DriverObject 80e20610
Current Irp 00000000 RefCount 1 Type 00000004 Flags 00001040
DevExt 80e52830 DevObjExt 80e52838 DevNode 80e52628
ExtensionFlags (0000000000)
AttachedDevice (Upper) 80d78b28 \Driver\i8042prt
Device queue is not busy.
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Съел

Статья • 05.04.2024

Расширение **!ate** отображает запись альтернативной таблицы страницы (ATE) для указанного адреса.

```
dbgcmd

    !ate Address
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Адрес*
Указывает ATE для отображения.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о таблицах страниц и каталогах *страниц см. в разделе* "Внутренние компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

## Замечания

Это расширение доступно только на компьютерах на основе Itanium.

Флаги состояния для ATE показаны в следующей таблице. Отображение **!ate** указывает эти биты с прописными буквами или дефисами и добавляет дополнительные сведения, а также.

Развернуть таблицу

| Отображение при установке | Отображение при очистке | Значение |
|---|---|---|
| V | - | Фиксация. |
| G | - | Доступ к ней недоступен. |
| E | - | Execute. |
| Ср. | R | Доступная для записи или доступная только для чтения. |
| L | - | Заблокирован. ATE заблокирован, поэтому все ошибки на странице, содержащей ATE, будут извлечены до тех пор, пока не будет выполнена текущая ошибка. Это может произойти в многопроцессорных системах. |
| Z | - | Заполните ноль. |
| N | - | Нет доступа. |
| O | - | Скопируйте запись. |
| I | - | Непрямый PTE. Этот ATE косвенно ссылается на другую физическую страницу. Страница, содержащая ATE, может иметь два конфликтующих атрибута ATE. |
| P | | Зарезервировано. |

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Bcb

Статья • 05.04.2024

Расширение **!bcb** отображает указанный блок управления буфером.

```
dbgcmd
```

```
!bcb Address
```

## Параметры

*Адрес*
Указывает адрес блока управления буфером.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Kdextx86.dll |
| **Windows XP и более поздние версии** | Недоступно (см. раздел "Примечания") |

## Дополнительная информация

Сведения об управлении кэшем см. в документации по microsoft Windows SDK и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

Для получения сведений о других расширениях управления кэшем используйте **расширение !cchelp** .

## Замечания

Это расширение доступно только для Windows 2000. В Windows XP или более поздней версии используйте **команду dt nt!_BCB Address** для отображения блока управления буфером напрямую.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !заблокировано

Статья • 05.04.2024

Расширение **!blockeddrv** отображает список заблокированных драйверов на целевом компьютере.

dbgcmd

```
    !blockeddrv
```

## DLL-библиотеки

Kdexts.dll

## Замечания

Рассмотрим пример:

dbgcmd

```
kd> !blockeddrv
Driver:      Status    GUID
afd.sys      0:        {00000008-0206-0001-0000-000030C964E1}
agp440.sys   0:        {0000005C-175A-E12D-5000-010020885580}
atapi.sys    0:        {0000005C-B04A-E12E-5600-000020885580}
audstub.sys  0:        {0000005C-B04A-E12E-5600-000020885580}
Beep.SYS     0:        {0000005C-B04A-E12E-5600-000020885580}
Cdfs.SYS     0:        {00000008-0206-0001-0000-000008F036E1}
.....
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bpid

Статья • 05.04.2024

**Расширение !bpid** запрашивает, что процесс на целевом компьютере прерывается в отладчик или запрашивает, что отладчик пользовательского режима будет присоединен к процессу на целевом компьютере.

dbgcmd

```
!bpid [Options] PID
```

# Параметры

*Параметр*
Управляет дополнительными действиями этой команды.

Допустимые значения параметра отображаются в следующей таблице.

Развернуть таблицу

| | |
|---|---|
| **-a** | Присоединяет новый отладчик пользовательского режима к процессу, указанному *PID*. Отладчик пользовательского режима выполняется на целевом компьютере. |
| **-s** | Добавляет точку останова, которая возникает в процессе WinLogon непосредственно перед разрывом в процессе пользовательского режима, заданном *PID*. Это позволяет пользователю проверить запрос перед попыткой действия. |
| **-w** | Сохраняет запрос в памяти на целевом компьютере. Целевая система может повторить запрос, но обычно это не обязательно. |

*PID*
Указывает идентификатор процесса требуемого процесса на целевом компьютере. Если вы используете это для управления отладчиком пользовательского режима на целевом компьютере, *PID* должен быть идентификатором процесса целевого приложения, а не отладчика в пользовательском режиме. (Так как идентификаторы

процессов обычно перечислены в десятичном формате, может потребоваться префикс с **0n** или преобразовать его в шестнадцатеричный формат.)

# DLL-библиотеки

Kdexts.dll

Эта команда расширения поддерживается на целевых компьютерах на основе x86 на основе x64.

# Замечания

Эта команда особенно полезна при перенаправлении входных и выходных данных отладчика пользовательского режима в отладчик ядра. Это приводит к разрыву целевого приложения в пользовательском режиме отладчика, который, в свою очередь, запрашивает входные данные от отладчика ядра. Дополнительные сведения см. в разделе "Управление отладчиком пользовательского режима" из отладчика ядра.

Если эта команда используется в другой ситуации, процесс пользовательского режима вызывает **DbgBreakPoint**. Обычно это будет прерываться непосредственно в отладчик ядра.

Параметр **-s** вызывает разрыв в WinLogon непосредственно перед прерыванием указанного процесса. Это полезно, если вы хотите выполнить действия отладки в контексте процесса WinLogon. Затем **команду g (Go)** можно использовать для перехода к второму перерыву.

Обратите внимание, что это расширение может завершиться ошибкой.

- Отсутствие ресурсов. **Расширение !bpid** внедряет поток в целевой процесс, поэтому система должна иметь достаточно ресурсов для создания потока. **Использование параметра -a** требует еще больше системных ресурсов, так как **!bpid -a** должно запускать полный экземпляр отладчика на целевом компьютере.

- Блокировка загрузчика уже проводится. Как !bpid, так **и !bpid** - требуется поток для запуска в целевом процессе, чтобы сделать его разрывом в отладчике. Если другой поток удерживает блокировку загрузчика, то поток !bpid **не сможет запуститься,** и разрыв в отладчике может не произойти. Таким образом, если !bpid **завершается ошибкой, если** для целевого

процесса доступно достаточно памяти пользовательского режима, возможно, что блокировка загрузчика проводится.

- Отсутствие разрешений. Для операции расширения !bpid требуется разрешение, достаточное для создания удаленного потока и подключения отладчика к заданному процессу.
- Нет доступа к ntsd.exe. Если ntsd.exe не найден в обычно известном пути, !bpid не сможет задать соответствующий идентификатор идентификатора. Обратите внимание, что ntsd.exe по умолчанию не включены в Windows Vista.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Btb

Статья • 05.04.2024

Расширение **!btb** отображает процессор на основе Itanium, буфер трассировки ветви (В ТБ) и регистры трассировки для текущего процессора.

dbgcmd

```
!btb
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## DLL-библиотеки

**Windows XP и более поздние версии**

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе Itanium.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Bth

Статья • 05.04.2024

Расширение **!bth** отображает журнал трассировок ветви на основе Itanium для указанного процессора.

dbgcmd

```
!bth [Processor]
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Процессор*
Указывает процессор. Если *обработчик* опущен, отображается журнал трассировки ветви для всех процессоров.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе Itanium.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bugdump

Статья • 08.07.2024

Форматы **расширений !bugdump** и отображают сведения, содержащиеся в буферах обратного вызова ошибки.

```
dbgsyntax

    !bugdump [Component]
```

# Параметры

*Компонент*
Указывает компонент, для которого необходимо проверить данные обратного вызова. Если опущено, отображаются все данные обратного вызова проверки ошибок.

## DLL-библиотеки

Kdexts.dll

# Замечания

Это расширение можно использовать только после того, как произошла проверка ошибок или при отладке файла аварийного дампа в режиме ядра.

Параметр *Component* соответствует окончательному параметру, используемому в KeRegisterBugCheckCallback.

Буферы, в которых хранятся данные обратного вызова, недоступны в небольшом дампе памяти. Эти буферы присутствуют в дампах памяти ядра и дампах полной памяти. Однако в Windows XP с пакетом обновления 1 (SP1) Windows Server 2003 и более поздних версиях Windows файл дампа создается до вызова подпрограмм BugCheckCallback **драйверов**, поэтому эти буферы не будут содержать данные, записанные этими подпрограммами.

Если выполняется динамическая отладка аварийной системы, все данные обратного вызова будут присутствовать.

# Дополнительная информация

Дополнительные сведения см. в статье "Чтение данных обратного вызова для проверки ошибок".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bushnd

Статья • 05.04.2024

Расширение **!bushnd** отображает структуру HAL BUS_HANDLER.

```
dbgsyntax

    !bushnd [Address]
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес структуры HAL BUS_HANDLER. Если опущено, **!bushnd** отображает список автобусов и базовый адрес обработчика.

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ca

Статья • 05.04.2024

Расширение **!ca** отображает сведения о области управления.

```
dbgsyntax
```

```
!ca [Address | 0 | -1] [Flags]
```

## Параметры

*Адрес*
Адрес области управления. При указании 0 для этого параметра отображаются сведения обо всех областях управления. При указании -1 для этого параметра отображаются сведения о неиспользуемом списке сегментов.

*Flags*
Флаги, указывающие, какие сведения отображаются. Этот параметр является битовой или одним или несколькими из следующих флагов.

Развернуть таблицу

| Флаг | Description |
|---|---|
| 0x1 | Отображение сведений о сегменте. |
| 0x2 | Отображение сведений подраздела. |
| 0x4 | Отображение списка сопоставленных представлений. (Windows 7 и более поздние версии) |
| 0x8 | Отображение компактных (однострочных) выходных данных. |
| 0x10 | Отображение областей управления с поддержкой файлов. |
| 0x20 | Отображение областей управления, поддерживаемых файлом страницы. |
| 0x40 | Отображение областей управления изображением. |

Если ни один из последних трех флагов не указан, отображаются все три типа области управления.

## DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о областях управления см. в разделах "Внутренние элементы Microsoft Windows" Марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Чтобы получить список областей управления всех сопоставленных файлов, используйте **расширение !memusage** .

Рассмотрим пример:

dbgcmd

```
kd> !memusage
 loading PFN database
loading (99% complete)
             Zeroed:     16 (    64 kb)
               Free:      0 (     0 kb)
            Standby:   2642 ( 10568 kb)
           Modified:    720 (  2880 kb)
    ModifiedNoWrite:      0 (     0 kb)
       Active/Valid:  13005 ( 52020 kb)
         Transition:      0 (     0 kb)
            Unknown:      0 (     0 kb)
              TOTAL:  16383 ( 65532 kb)
  Building kernel map
  Finished building kernel map

  Usage Summary (in Kb):
Control Valid Standby Dirty Shared Locked PageTables  name
ff8636e8    56    376     0     0     0     0  mapped_file( browseui.dll )
ff8cf388    24      0     0     0     0     0  mapped_file( AVH32DLL.DLL )
ff8d62c8    12      0     0     0     0     0  mapped_file( PSAPI.DLL )
ff8dd468   156     28     0     0     0     0  mapped_file( INOJOBSV.EXE )
fe424808   136     88     0    52     0     0  mapped_file( oleaut32.dll )
fe4228a8   152     44     0   116     0     0  mapped_file( MSVCRT.DLL )
ff8ec848     4      0     0     0     0     0    No Name for File
ff859de8     0     32     0     0     0     0  mapped_file( timedate.cpl )
. . . . .
```

```
kd> !ca ff8636e8

ControlArea @ff8636e8
  Segment:    e1b74548    Flink              0   Blink:               0
 Section Ref        0    Pfn Ref           6c   Mapped Views:        1
  User Ref           1    Subsections        5   Flush Count:         0
  File Object ff86df88    ModWriteCount      0   System Views:        0
  WaitForDel         0    Paged Usage      380   NonPaged Usage       e0
  Flags (10000a0) Image File HadUserReference

   File: \WINNT\System32\browseui.dll

Segment @ e1b74548:
   Base address        0  Total Ptes        c8  NonExtendPtes:       c8
   Image commit        1  ControlArea ff8636e8  SizeOfSegment: c8000
   Image Base          0  Committed          0  PTE Template:   31b8438
 Based Addr   76e10000  ProtoPtes   e1b74580  Image Info:    e1b748a4

Subsection 1. @ ff863720
   ControlArea: ff8636e8  Starting Sector 0 Number Of Sectors 2
   Base Pte     e1b74580  Ptes In subsect        1 Unused Ptes          0
   Flags              15  Sector Offset          0 Protection           1
    ReadOnly CopyOnWrite

Subsection 2. @ ff863740
   ControlArea: ff8636e8  Starting Sector 2 Number Of Sectors 3d0
   Base Pte     e1b74584  Ptes In subsect       7a Unused Ptes          0
   Flags              35  Sector Offset          0 Protection           3
    ReadOnly CopyOnWrite

Subsection 3. @ ff863760
   ControlArea: ff8636e8  Starting Sector 3D2 Number Of Sectors 7
   Base Pte     e1b7476c  Ptes In subsect        1 Unused Ptes          0
   Flags              55  Sector Offset          0 Protection           5
    ReadOnly CopyOnWrite

Subsection 4. @ ff863780
   ControlArea: ff8636e8  Starting Sector 3D9 Number Of Sectors 21f
   Base Pte     e1b74770  Ptes In subsect       44 Unused Ptes          0
   Flags              15  Sector Offset          0 Protection           1
    ReadOnly CopyOnWrite

Subsection 5. @ ff8637a0
   ControlArea: ff8636e8  Starting Sector 5F8 Number Of Sectors 3a
   Base Pte     e1b74880  Ptes In subsect        8 Unused Ptes          0
   Flags              15  Sector Offset          0 Protection           1
    ReadOnly CopyOnWrite
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Обратного вызова

Статья • 05.04.2024

Расширение **!callback отображает данные обратного** вызова, связанные с ловушкой для указанного потока.

dbgsyntax

```
!callback Address [Number]
```

# Параметры

*Адрес*
Указывает шестнадцатеричный адрес потока. Если это значение равно -1 или опущено, используется текущий поток.

*Число*
Указывает число требуемого кадра обратного вызова. Этот кадр отмечается в дисплее.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x86.

# Дополнительная информация

Сведения о системных ловушках см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Если система не испытала системную ловушку, это расширение не будет создавать полезные данные.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !calldata

Статья • 08.07.2024

Расширение **!calldata** отображает сведения о производительности в виде статистики вызовов процедур из именованной таблицы.

dbgsyntax

```
!calldata Table
```

## Параметры

*Стол*
Имя таблицы, которая собирает данные вызова.

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !can_write_kdump

Статья • 05.04.2024

**Расширение !can_write_kdump** проверяет, достаточно ли дискового пространства на целевом компьютере для записи файла дампа ядра указанного типа.

```
dbgsyntax

!can_write_kdump [-dn] [Options]
```

# Параметры

**-Dn**
Указывает, что файловая система на целевом компьютере является файловой системой NTFS. Если этот параметр опущен, то не удается определить объем свободного места на диске, и появится предупреждение. Однако объем необходимого пространства по-прежнему будет отображаться.

*Допустимы* следующие параметры:

**-t**
Указывает, что расширение должно определить, достаточно ли места для мини-модуля.

**-s**
Указывает, что расширение должно определить, достаточно ли места для сводного дампа ядра. Это значение по умолчанию.

**-f**
Указывает, что расширение должно определить, достаточно ли места для полного дампа ядра.

# DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Kext.dll |
| **Windows XP и более поздние версии** | Kext.dll |

# Замечания

Если параметр *не* указан, расширение определит, есть ли достаточно места для сводного дампа ядра.

В следующем примере файловая система не указана:

dbgcmd

```
kd> !can_write_kdump
Checking kernel summary dump...
WARNING: Can't predict how many pages will be used, assuming worst-case.
Physical memory: 285560 KB
Page file size: 1572864 KB
NO: Page file too small
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !cbreg

Статья • 05.04.2024

Расширение **!cbreg** отображает регистры сокета CardBus и регистры архитектуры карточки (ExCA).

```
dbgsyntax

    !cbreg [%%]Address
```

## Параметры

%%
Указывает, что *адрес* является физическим адресом, а не виртуальным адресом.

*Адрес*
Указывает адрес отображаемого регистра.

## DLL-библиотеки

Kext.dll

**Расширение !cbreg** доступно только для целевого компьютера на основе x86.

## Дополнительная информация

**Расширение !exca** можно использовать для отображения регистров PCIC ExCA по номеру сокета.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !cchelp

Статья • 05.04.2024

Расширение **!cchelp** отображает краткий текст справки в командном окне отладчика для некоторых расширений управления кэшем.

```
dbgsyntax
```

```
!cchelp
```

## DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения об управлении кэшем см. в документации по microsoft Windows SDK и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

Расширение **!cchelp** отображает справку для **расширений управления кэшем !bcb**, !defwrites**, !finddata** и **!scm** cache management. Другие расширения управления кэшем включают **!openmaps** и **!pcm**.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !chklowmem

Статья • 26.10.2023

Расширение **!chklowmem** определяет, заполнены ли страницы физической памяти менее 4 ГБ требуемым шаблоном заполнения на компьютере, который был загружен с помощью параметров **/pae** и **/nolowmem** .

dbgsyntax

```
!chklowmem
```

## DLL

Kdexts.dll

# Комментарии

Это расширение полезно при проверке правильности работы драйверов в режиме ядра с физической памятью, превышающей границу 4 ГБ. Как правило, драйверы не усекаются путем усечения физического адреса до 32 бит, а затем в письменной форме ниже границы 4 ГБ. Расширение **!chklowmem** обнаруживает все операции записи ниже границы в 4 ГБ.

# !cmreslist

Статья • 05.04.2024

Расширение **!cmreslist** отображает структуру CM_RESOURCE_LIST для указанного объекта устройства.

dbgsyntax

```
!cmreslist Address
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес структуры CM_RESOURCE_LIST.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Сведения о структуре CM_RESOURCE_LIST см. в документации по комплекту драйверов Windows (WDK).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Cpuinfo

Статья • 05.04.2024

Расширение !cpuinfo **отображает подробные** сведения о ЦП целевого компьютера.

Синтаксис

dbgsyntax

```
!cpuinfo [Processor]
```

## Параметры

*Процессор*
Указывает процессор для отображения. Если это опущено, отображаются все процессоры.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения об отладке многопроцессорных компьютеров см. в разделе "Синтаксис многопроцессора".

## Замечания

**При отладке локального ядра можно использовать команду расширения !cpuinfo**.

Ниже приведен пример, созданный процессором на основе x86:

dbgcmd

```
kd> !cpuinfo
CP F/M/S Manufacturer  MHz Update Signature Features
 0 6,1,9 GenuineIntel  198 000000d200000000 000000ff
```

Столбец **CP** указывает номер процессора. Столбец **производителя указывает изготовителя** процессора. Столбец **MHz** или **Speed** указывает скорость процессора в МГц, если он доступен.

Для процессора на основе x86 или процессора на основе **x64 столбец F** отображает номер семейства процессоров, **столбец M** отображает номер модели процессора, а **столбец S** отображает размер шага.

Другие столбцы также будут отображаться в зависимости от конкретной архитектуры компьютера.

Дополнительные сведения о интерпретации определенных значений для каждой записи, а также о дополнительных столбцах см. в руководстве по обработчику.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !db, !dc, !dd, !dp, !dq, !du, !dw

Статья • 05.04.2024

**Расширения !db**, !dc, !dd**, !dp**, **!dq**, **!du** и **!dw** отображают данные по указанному физическому адресу на целевом компьютере.

Эти команды расширения не следует путать с командой **d* (отображаемая память)** или с командой **расширения !ntsdexts.dp**.

dbgcmd

```
!db [Caching] [-m] [PhysicalAddress] [L Size]
!dc [Caching] [-m] [PhysicalAddress] [L Size]
!dd [Caching] [-m] [PhysicalAddress] [L Size]
!dp [Caching] [-m] [PhysicalAddress] [L Size]
!dq [Caching] [-m] [PhysicalAddress] [L Size]
!du [Caching] [-m] [PhysicalAddress] [L Size]
!dw [Caching] [-m] [PhysicalAddress] [L Size]
```

# Параметры

*Кэширование*
Может быть любым из следующих значений. Значение *кэширования* должно быть окружено квадратными скобками:

**[c]**
Это расширение может считываться из кэшированного памяти.

**[uc]**
Это расширение приводит к чтению из некичированного памяти.

**[wc]**
Это расширение приводит к чтению из объединенной памяти записи.

**-m**
Приводит к тому, что память будет считываться по одной единице за раз. Например, **!db -m** считывает память в 8-разрядных блоках и **!dw -m** считывает память в 16-разрядных блоках. Если оборудование не поддерживает 32-разрядное чтение физической памяти, возможно, потребуется использовать **параметр -m** . Этот параметр не влияет на длину или внешний вид дисплея. Это влияет только на доступ к памяти.

*PhysicalAddress*
Указывает первый физический адрес, отображаемый в шестнадцатеричном формате. Если это опущено при первом использовании этой команды, адрес по умолчанию равен нулю. Если это опущено при последующем использовании, экран начнется, где заканчивается последнее отображение.

**Размер L** ****
Указывает количество блоков памяти для отображения. Размер блока определяется точным расширением.

# Среда

Развернуть таблицу

| | |
|---|---|
| **Режим** | Режим ядра |

# DLL-библиотеки

Kext.dll

# Дополнительная информация

Чтобы записать в физическую **память, используйте расширения !e*** . Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

# Замечания

Эти расширения отображают физическую память, но их форматы отображения и длина по умолчанию отличаются:

- Расширение **!db** отображает шестнадцатеричные байты и эквиваленты символов ASCII. Длина по умолчанию составляет 128 байт.
- Расширение **!dc** отображает значения DWORD и эквиваленты символов ASCII. Длина по умолчанию составляет 32 DWORD (128 байт).
- Расширение **!dd** отображает значения DWORD. Длина по умолчанию составляет 32 DWORD (128 байт).

- Расширение **!dp** отображает значения ULONG_PTR. Это 32-разрядные или 64-разрядные слова в зависимости от размера инструкции. Длина по умолчанию составляет 128 байт.
- Расширение **!dq** отображает значения ULONG64_PTR. Это 32-разрядные слова. Длина по умолчанию составляет 128 байт.
- Расширение **!du** отображает символы ЮНИКОДа. Длина по умолчанию составляет 16 символов (32 байта) или до тех пор, пока не будет обнаружен символ NULL.
- Расширение **!dw** отображает значения WORD. Длина по умолчанию составляет 64 DWORD (128 байт).

Следовательно, использование двух этих расширений, которые отличаются с одинаковым значением *size* , скорее всего, приведет к разнице в общем объеме памяти, отображаемой. Например, при использовании команды **!db L 32** отображается 32 байта (как шестнадцатеричные байты), в то время как команда **!dd L 32** приводит к отображению 128 байтов (как значения DWORD).

Ниже приведен пример, в котором требуется флаг атрибута кэширования:

dbgcmd

```
kd> !dc e9000
physical memory read at e9000 failed
If you know the caching attributes used for the memory,
try specifying [c], [uc] or [wc], as in !dd [c] <params>.
WARNING: Incorrect use of these flags will cause unpredictable
processor corruption. This may immediately (or at any time in
the future until reboot) result in a system hang, incorrect data
being displayed or other strange crashes and corruption.

kd> !dc [c] e9000
#   e9000 000ea002 000ea002 000ea002 000ea002 ................
#   e9010 000ea002 000ea002 000ea002 000ea002 ................
```


## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !dbgprint

Статья • 05.04.2024

Расширение **!dbgprint** отображает строку, которая ранее была отправлена в **буфер DbgPrint** .

dbgcmd

```
!dbgprint
```

## DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о **DbgPrint, KdPrint, DbgPrintEx и KdPrintEx** см. в разделе "Отправка выходных данных в отладчик".

# Замечания

Подпрограммы **в режиме ядра dbgPrint, KdPrint, DbgPrintEx и KdPrintEx** отправляют форматированную строку в буфер на целевом компьютере. Строка автоматически отображается в окне команды отладчика на хост-компьютере, если такая печать не отключена.

Как правило, сообщения, отправленные в этот буфер, отображаются автоматически в окне команды отладчика. Однако это отображение можно отключить с помощью служебной программы глобальных флагов (gflags.exe). Кроме того, этот дисплей не отображается автоматически во время локальной отладки ядра. Дополнительные сведения см. в разделе "Буфер DbgPrint" в разделе "Чтение и фильтрация сообщений отладки".

**Расширение !dbgprint** приводит к отображению содержимого этого буфера (независимо от того, отключена ли автоматическая печать). В нем не отображаются сообщения, отфильтрованные на основе их компонента и уровня важности. (Дополнительные сведения об этой фильтрации см. в разделе Чтение и фильтрация сообщений отладки.)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !dblink

Статья • 05.04.2024

Расширение **!dblink** отображает связанный список в обратном направлении.

dbgcmd

```
!dblink Address [Count] [Bias]
```

## Параметры

*Адрес*
Задает адрес структуры LIST_ENTRY. Отображение начнется с этого узла.

*Численность*
Указывает максимальное количество отображаемых записей списка. Если это опущено, значение по умолчанию равно 32.

*Предвзятости*
Указывает маску битов, которые следует игнорировать в каждом указателе. Каждый **адрес blink** имеет anDed with (NOT *Bias*) перед переходом к следующему расположению. Значение по умолчанию равно нулю (иными словами, не игнорируйте биты).

### DLL-библиотеки

Kdexts.dll

## Замечания

**Расширение !dblink проходит по полям Blink** структуры LIST_ENTRY и отображает до четырех ULONG на каждом адресе. Чтобы перейти в другое направление, используйте **!dflink**.

Команда dl (отображаемый связанный список) **является более универсальной, чем !dblink и !dflink**.

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Dcr

Статья • 05.04.2024

Расширение **!dcr** отображает регистр элемента управления по умолчанию (DCR) по указанному адресу.

```
dbgcmd
```

```
!dcr Expression [DisplayLevel]
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Выражение*
Указывает шестнадцатеричный адрес отображаемого DCR. Выражение **@dcr** также можно использовать для этого параметра. В этом случае отображаются сведения о текущем DCR процессора.

*DisplayLevel*
Может быть одним из следующих вариантов:

**0**
Приводит к отображению только значений каждого поля DCR. Это значение по умолчанию.

**1**
Приводит к тому, что отображение содержит более подробные сведения о каждом из полей DCR, которые не зарезервированы или игнорируются.

**2**
Приводит к тому, что отображение содержит более подробные сведения обо всех полях DCR, включая те, которые игнорируются или зарезервированы.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе Itanium.

# Замечания

DCR задает параметры по умолчанию для значений регистра состояния процессора при прерывании. DCR также указывает некоторые дополнительные глобальные элементы управления, а также возможность отложения спекулятивной нагрузки.

Вот несколько примеров.

dbgcmd

```
kd> !dcr @dcr
dcr:pp be lc dm dp dk dx dr da dd
1 0 1 1 1 1 1 1 1 1

kd> !dcr @dcr 2

  pp : 1 : Privileged Performance Monitor Default
  be : 0 : Big-Endian Default
  lc : 1 : IA-32 Lock check Enable
  rv : 0 : reserved1
  dm : 1 : Defer TLB Miss faults only
  dp : 1 : Defer Page Not Present faults only
  dk : 1 : Defer Key Miss faults only
  dx : 1 : Defer Key Permission faults only
  dr : 1 : Defer Access Rights faults only
  da : 1 : Defer Access Bit faults only
  dd : 0 : Defer Debug faults only
  rv : 0 : reserved2
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Dcs

Статья • 05.04.2024

Расширение **!dcs** устарело. Чтобы отобразить пространство конфигурации PCI, используйте функцию *устройства шины !pci 100*.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Взаимоблокировки

Статья • 05.04.2024

Расширение **!взаимоблокировки** отображает сведения о взаимоблокировках, собранных параметром **обнаружения** взаимоблокировок средства проверки драйверов.

dbgcmd

```
!deadlock
!deadlock 1
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о средстве проверки драйверов см. в документации по комплекту драйверов Windows (WDK).

## Замечания

Это расширение предоставляет только полезную информацию, если параметр обнаружения взаимоблокировки **средства проверки драйверов обнаружил нарушение иерархии блокировки и выдал ошибку проверка 0xC4 (DRIVER_VERIFIER_DETECTED_VIOLATION).**

Без каких-либо аргументов **расширение !взаимоблокировки** приводит к отображению базовой топологии иерархии блокировки. Если проблема не является простой циклической взаимоблокировкой, эта команда описывает ситуацию, которая произошла.

Расширение **!взаимоблокировки 1** приводит к отображению трассировок стека. Отображаемые стеки будут активными во время приобретения блокировок.

Рассмотрим пример:

dbgcmd

```
0:kd> !deadlock

Deadlock detected (2 resources in 2 threads):

Thread 0: A B
Thread 1: B A

Where:
Thread 0 = 8d3ba030
Thread 1 = 8d15c030
Lock A =   bba2af30 Type 'Spinlock'
Lock B =   dummy!GlobalLock Type 'Spinlock'
```

Это указывает, какие потоки и какие блокировки участвуют. Однако она предназначена для сводки и может оказаться недостаточной информацией для адекватной отладки ситуации.

Используйте **!взаимоблокировку 1** , чтобы распечатать содержимое стека вызовов во время получения каждой блокировки, участвующих в взаимоблокировке. Так как это трассировки стека во время выполнения, они будут более полными, если используется проверка сборка. Проверенные сборки были доступны в более ранних версиях Windows до Windows 10 версии 1803. В свободной сборке они могут быть усечены после одной строки.

dbgcmd

```
0:kd> !deadlock 1

Deadlock detected (2 resources in 2 threads):

Thread 0 (8D14F750) took locks in the following order:

    Lock A -- b7906f30 (Spinlock)
    Stack:   dummy!DummyActivateVcComplete+0x63
             dummy!dummyOpenVcChannels+0x2E1
             dummy!DummyAllocateRecvBufferComplete+0x436
             dummy!DummyAllocateComplete+0x55
             NDIS!ndisMQueuedAllocateSharedHandler+0xC9
             NDIS!ndisWorkerThread+0xEE

    Lock B -- dummy!GlobalLock (Spinlock)
    Stack:   dummy!dummyQueueRecvBuffers+0x2D
             dummy!DummyActivateVcComplete+0x90
             dummy!dummyOpenVcChannels+0x2E1
             dummy!DummyAllocateRecvBufferComplete+0x436
             dummy!DummyAllocateComplete+0x55

Thread 1 (8D903030) took locks in the following order:

    Lock B -- dummy!GlobalLock (Spinlock)
```

```
    Stack:   dummy!dummyRxInterruptOnCompletion+0x25D
             dummy!DummyHandleInterrupt+0x32F
             NDIS!ndisMDpcX+0x3C
             ntkrnlpa!KiRetireDpcList+0x5D

    Lock A -- b7906f30 (Spinlock)
    Stack:   << Current stack >>
```

С этой информацией у вас почти все необходимое, кроме текущего стека:

dbgcmd

```
0: kd> k
ChildEBP RetAddr
f78aae6c 80664c58 ntkrnlpa!DbgBreakPoint
f78aae74 8066523f ntkrnlpa!ViDeadlockReportIssue+0x2f
f78aae9c 806665df ntkrnlpa!ViDeadlockAnalyze+0x253
f78aaee8 8065d944 ntkrnlpa!VfDeadlockAcquireResource+0x20b
f78aaf08 bfd6df46 ntkrnlpa!VerifierKeAcquireSpinLockAtDpcLevel+0x44
f78aafa4 b1bf2d2d dummy!dummyRxInterruptOnCompletion+0x2b5
f78aafc4 bfde9d8c dummy!DummyHandleInterrupt+0x32f
f78aafd8 804b393b NDIS!ndisMDpcX+0x3c
f78aaff4 804b922b ntkrnlpa!KiRetireDpcList+0x5d
```

Из этого можно увидеть, какие замки были вовлечены и где они были приобретены. Это должно быть достаточно информации для отладки взаимоблокировки. Если исходный код доступен, можно использовать отладчик, чтобы увидеть, где возникла проблема:

dbgcmd

```
0: kd> .lines
Line number information will be loaded

0: kd> u dummy!DummyActivateVcComplete+0x63 l1
dummy!DummyActivateVcComplete+63 [d:\nt\drivers\dummy\vc.c @ 2711]:
b1bfe6c9 837d0c00         cmp     dword ptr [ebp+0xc],0x0

0: kd> u dummy!dummyQueueRecvBuffers+0x2D l1
dummy!dummyQueueRecvBuffers+2d [d:\nt\drivers\dummy\receive.c @ 2894]:
b1bf4e39 807d0c01         cmp     byte ptr [ebp+0xc],0x1

0: kd> u dummy!dummyRxInterruptOnCompletion+0x25D l1
dummy!dummyRxInterruptOnCompletion+25d [d:\nt\drivers\dummy\receive.c @
1424]:
b1bf5d05 85f6             test    esi,esi

0: kd> u dummy!dummyRxInterruptOnCompletion+0x2b5 l1
dummy!dummyRxInterruptOnCompletion+2b5 [d:\nt\drivers\dummy\receive.c @
```

```
1441]:
b1bf5d5d 8b4648          mov     eax,[esi+0x48]
```

Теперь вы знаете имя исходного файла и номер строки, в которой произошло приобретение. В этом случае исходные файлы показывают, что потоки ведут себя следующим образом:

- Поток 1: **DummyActivateVcComplete** взял фиктивную **минипорт-блокировку**. Затем она называется **dummyQueueRecvBuffers**, которая взяла фиктивную **глобальную блокировку**.
- Поток 2: **dummyRxInterruptOnCompletion** взял глобальную блокировку. Затем, несколько строк позже, он взял минипорт блокировки.

На этом этапе взаимоблокировка становится совершенно ясной.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !defwrites

Статья • 05.04.2024

Расширение **!defwrites** отображает значения переменных ядра, используемых диспетчером кэша.

dbgcmd

```
!defwrites
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о регулировании записи см. в статье "*Внутренние* компоненты Microsoft Windows" Марком Руссиновичем и Дэвидом Соломоном.

Для получения сведений о других расширениях управления кэшем используйте **расширение !cchelp** .

## Замечания

Когда число отложенных записей ("грязное страниц") становится слишком большим, запись страниц будет регулироваться. Это расширение позволяет узнать, достигла ли эта система.

Рассмотрим пример:

dbgcmd

```
kd> !defwrites
*** Cache Write Throttle Analysis ***

        CcTotalDirtyPages:                   0 (       0 Kb)
        CcDirtyPageThreshold:             1538 (    6152 Kb)
        MmAvailablePages:                 2598 (   10392 Kb)
        MmThrottleTop:                     250 (    1000 Kb)
        MmThrottleBottom:                   30 (     120 Kb)
        MmModifiedPageListHead.Total:      699 (    2796 Kb)
```

```
Write throttles not engaged
```

В этом случае нет грязное страниц. Если **CcTotalDirtyPages** достигает 1538 (значение **CcDirtyPageThreshold**), запись будет отложена до тех пор, пока число страниц грязное будет сокращено.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !devext

Статья • 05.04.2024

Расширение **!devext** отображает сведения о расширении устройства для конкретной шины для устройств на различных автобусах.

dbgcmd

```
!devext Address TypeCode
```

# Параметры

## *Адрес*

Указывает шестнадцатеричный адрес отображаемого расширения устройства.

## *TypeCode*

Указывает тип объекта, которому принадлежит расширение устройства для отображения. Коды типов не учитывает регистр. Допустимые коды типов:

Развернуть таблицу

| Код типа | Object |
|---|---|
| ISAPNP | Расширение устройства ISA PnP |
| PCMCIA | Расширение устройства PCMCIA |
| СПРЯТАЛ | Расширение устройства HID |

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Дополнительные сведения о расширениях устройств см. в документации по комплекту драйверов Windows (WDK).

# Замечания

**Расширения !usbhub**, **!hidfdo и !hidpdo** устарели; их функциональные возможности были интегрированы в **!devext**.

Для этих типов объектов, которые больше не поддерживаются **!devext**, используйте **команду отладчика dt (display Type).**

Ниже приведен пример расширения устройства ISA PnP:

dbgcmd

```
kd> !devext e0000165fff32190 ISAPNP
ISA PnP FDO @ 0x00000000, DevExt @ 0xe0000165fff32190, Bus # 196639
Flags (0x854e2530)  DF_ACTIVATED, DF_QUERY_STOPPED,
                    DF_STOPPED, DF_RESTARTED_NOMOVE,
                    DF_BUS
                    Unknown flags 0x054e2000

NumberCSNs           - -536870912
ReadDataPort         - 0x0000000d (mapped)
AddressPort          - 0x00000000 (not mapped)
CommandPort          - 0x00000000 (not mapped)
DeviceList           - 0xe000000085007b50
CardList             - 0x00000000
PhysicalBusDevice    - 0x00000000
AttachedDevice       - 0x00000000
SystemPowerState     - Unspecified
DevicePowerState     - Unspecified
```

Ниже приведен пример для устройства PCI:

dbgcmd

```
kd> !devext e0000000858c31b0 PCI
PDO Extension, Bus 0x0, Device 0, Function 0.
  DevObj 0xe0000000858c3060 PCI Parent Bus FDO DevExt 0xe0000000858c4960
  Device State = PciNotStarted
  Vendor ID 8086 (INTEL)  Device ID 123D
  Class Base/Sub 08/00  (Base System Device/Interrupt Controller)
  Programming Interface: 20, Revision: 01, IntPin: 00, Line Raw/Adj 00/00
  Enables ((cmd & 7) = 106): BM   Capabilities Pointer = <none>
  CurrentState:          System Working,  Device D0
  WakeLevel:             System Unspecified,  Device Unspecified
  Requirements: <none>
```

---

# Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !devhandles

Статья • 05.04.2024

Расширение **!devhandles** отображает открытые дескрипторы для указанного устройства.

dbgcmd

```
!devhandles Address
```

## Параметры

*Адрес*
Указывает адрес устройства, для которого будут отображаться открытые дескрипторы.

## DLL-библиотеки

Kdexts.dll

## Замечания

Для отображения полных сведений об дескрипторе это расширение требует частных символов.

Адрес объекта устройства можно получить с помощью расширений !drvobj **или** **!devnode**.

Ниже приведен усеченный пример:

dbgcmd

```
lkd> !devhandles 0x841153d8

Checking handle table for process 0x840d3940
Handle table at 95fea000 with 578 Entries in use

Checking handle table for process 0x86951d90
Handle table at 8a8ef000 with 28 Entries in use

...

Checking handle table for process 0x87e63650
```

```
Handle table at 947bc000 with 308 Entries in use

Checking handle table for process 0x87e6f4f0
00000000: Unable to read handle table
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !devnode

Статья • 05.04.2024

Расширение **!devnode** отображает сведения о узле в дереве устройств.

dbgcmd

```
!devnode Address [Flags] [Service]
!devnode 1
!devnode 2
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес расширения устройства, узел которого должен отображаться. Если это равно нулю, отображается корень основного дерева устройств.

*Flags*
Указывает уровень отображаемых выходных данных. Это может быть любое сочетание следующих битов:

Bit 0 (0x1)
Вызывает включение всех дочерних элементов узла устройства.

Бит 1 (0x2)
Приводит к включению используемых ресурсов (CM_RESOURCE_LIST). К ним относятся конфигурация загрузки, сообщаемая IRP_MN_QUERY_RESOURCES, а также ресурсы, выделенные устройству в параметре *AllocatedResources* IRP_MN_START_DEVICE.

Бит 2 (0x4)
Приводит к тому, что отображение включает необходимые ресурсы (IO_RESOURCE_REQUIREMENTS_LIST), как сообщается IRP_MN_FILTER_RESOURCE_REQUIREMENTS.

Бит 3 (0x8)
Приводит к добавлению списка переведенных ресурсов, выделенных устройству, в *параметре AllocatedResourcesTranslated* IRP_MN_START_DEVICE.

Бит 4 (0x10)
Указывает, что должны отображаться только узлы устройств, которые не запущены.

Бит 5 (0x20)
Указывает, что должны отображаться только узлы устройства с проблемами. (Это узлы, содержащие биты флага DNF_HAS_PROBLEM или DNF_HAS_PRIVATE_PROBLEM.)

*Служба*
Указывает имя службы. Если это включено, будут отображаться только те узлы устройств, управляемые этой службой. (If *Флаги* включают битовые 0x1, узлы устройств, управляемые этой службой, и все их дочерние элементы будут отображаться.)

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Сведения о деревьях устройств см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Команда **!devnode 1** перечисляет все ожидающие удаления объектов устройства.

Команда **!devnode 2** перечисляет все ожидающие извлечения объектов устройства.

Вы можете использовать **!devnode 0 1** для просмотра всего дерева устройств.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !devobj

Статья • 05.04.2024

Расширение **!devobj** отображает подробные сведения о структуре DEVICE_OBJECT.

```
dbgcmd
```

```
!devobj DeviceObject
```

## Параметры

*DeviceObject*
Указывает объект устройства. Это может быть шестнадцатеричный адрес этой структуры или имя устройства.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Примеры и приложения этой команды расширения см. в самонастраивающийся отладке. Дополнительные сведения об объектах устройств см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Если *DeviceObject* указывает имя устройства, но не предоставляет префикс, предполагается префикс \Device\. Обратите внимание, что эта команда проверка, чтобы узнать, является ли *DeviceObject* допустимым адресом или именем устройства перед использованием средства оценки выражений.

Отображаемые сведения включают имя устройства объекта, сведения о текущем IRP устройства и список адресов ожидающих irPs в очереди устройства. Он также содержит сведения об объектах устройств, слои которых наложены на вершине этого объекта (в списке "ПрисоединеннаяDevice") и те, которые слои под этим объектом (перечислены как "AttachedTo").

Адрес объекта устройства можно получить с помощью расширений !drvobj **или** **!devnode**.

Ниже приведен один пример.

dbgcmd

```
kd> !devnode
Dumping IopRootDeviceNode (= 0x80e203b8)
DevNode 0x80e203b8 for PDO 0x80e204f8
 Parent 0000000000   Sibling 0000000000   Child 0x80e56dc8
  InstancePath is "HTREE\ROOT\0"
  State = DeviceNodeStarted (0x308)
  Previous State = DeviceNodeEnumerateCompletion (0x30d)
  StateHistory[04] = DeviceNodeEnumerateCompletion (0x30d)
  StateHistory[03] = DeviceNodeStarted (0x308)
  StateHistory[02] = DeviceNodeEnumerateCompletion (0x30d)
  StateHistory[01] = DeviceNodeStarted (0x308)
  StateHistory[00] = DeviceNodeUninitialized (0x301)
  StateHistory[19] = Unknown State (0x0)
  .....
  StateHistory[05] = Unknown State (0x0)
  Flags (0x00000131)  DNF_MADEUP, DNF_ENUMERATED,
                      DNF_IDS_QUERIED, DNF_NO_RESOURCE_REQUIRED
  DisableableDepends = 11 (from children)

kd> !devobj 80e204f8
Device object (80e204f8) is for:
  \Driver\PnpManager DriverObject 80e20610
Current Irp 00000000 RefCount 0 Type 00000004 Flags 00001000
DevExt 80e205b0 DevObjExt 80e205b8 DevNode 80e203b8
ExtensionFlags (0000000000)
Device queue is not busy.
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !devstack

Статья • 08.07.2024

Расширение **!devstack** отображает форматное представление стека устройств, связанного с объектом устройства.

dbgcmd

```
!devstack DeviceObject
```

## Параметры

*DeviceObject*
Указывает объект устройства. Это может быть шестнадцатеричный адрес DEVICE_OBJECT структуры или имени устройства.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о стеках устройств см. в документации по комплекту драйверов Windows (WDK).

## Замечания

Если *DeviceObject* указывает имя устройства, но не предоставляет префикс, предполагается префикс \Device\. Обратите внимание, что эта команда проверяет, является ли *DeviceObject* допустимым адресом или именем устройства перед использованием вычислителя выражений.

Рассмотрим пример:

dbgcmd

```
kd> !devstack e000000085007b50
 !DevObj   !DrvObj            !DevExt   ObjectName
  e0000165fff32040  \Driver\kmixer     e0000165fff32190
> e000000085007b50  \Driver\swenum     e000000085007ca0  KSENUM#00000005
!DevNode e0000165fff2e010 :
```

```
  DeviceInst is "SW\{b7eafdc0-a680-11d0-96d8-00aa0051e51d}\{9B365890-165F-
11D0-A195-0020AFD156E4}"
 ServiceName is "kmixer"
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !dflink

Статья • 08.07.2024

Расширение **!dflink** отображает связанный список в направлении вперед.

```
dbgcmd
```

```
!dflink Address [Count] [Bias]
```

## Параметры

*Адрес*
Задает адрес структуры LIST_ENTRY. Отображение начнется с этого узла.

*Считать*
Указывает максимальное количество отображаемых записей списка. Если это опущено, значение по умолчанию равно 32.

*Предвзятость*
Указывает маску битов, которые следует игнорировать в каждом указателе. Каждый **адрес Flink** является ANDed with (NOT *Bias*) перед его переходом к следующему расположению. Значение по умолчанию равно нулю (иными словами, не игнорируйте биты).

### DLL-библиотеки

Kdexts.dll

## Замечания

Расширение **!dflink проходит по полям Flink** структуры LIST_ENTRY и отображает до четырех ULONG на каждом адресе. Чтобы перейти в другое направление, используйте **!dblink**.

Команда dl (отображаемый связанный список) **является более универсальной, чем !dblink и !dflink**.

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !diskspace

Статья • 08.07.2024

Расширение **!diskspace** отображает объем свободного места на жестком диске целевого компьютера.

dbgcmd

```
!diskspace Drive[:]
```

## Параметры

*Гнать*
Указывает букву диска диска. Двоеточие (:) после *диска* является необязательным.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Kext.dll |
| **Windows XP и более поздние версии** | Kext.dll |

## Замечания

Рассмотрим пример:

dbgcmd

```
kd> !diskspace c:
Checking Free Space for c: ..........
   Cluster Size 0 KB
 Total Clusters 4192901 KB
  Free Clusters 1350795 KB
    Total Space 1 GB (2096450 KB)
     Free Space 659.567871 MB (675397 KB)

kd> !diskspace f:
Checking Free Space for f:
f: is a CDROM drive. This function is not supported!
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Dma

Статья • 05.04.2024

Расширение **!dma** отображает сведения о подсистеме прямого доступа к памяти (DMA) и **параметре** проверки DMA средства проверки драйверов.

dbgcmd

```
!dma
!dma Adapter [Flags]
```

# Параметры

*Adapter* (Адаптер)
Указывает шестнадцатеричный адрес отображаемого адаптера DMA. Если это равно нулю, будут отображаться все адаптеры DMA.

*Flags*
Указывает сведения, которые необходимо включить в дисплей. Это может быть любое сочетание следующих битов. Значение по умолчанию — 0x1.

Bit 0 (0x1)
Приводит к включению сведений об универсальном адаптере.

Бит 1 (0x2)
Приводит к включению сведений о регистре карты. (Только если проверка DMA активна.)

Бит 2 (0x4)
Приводит к включению общих сведений о буфере. (Только если проверка DMA активна.)

Бит 3 (0x8)
Приводит к включению сведений о списке точечной и сборной. (Только если проверка DMA активна.)

Бит 4 (0x10)
Приводит к включению описания устройства для аппаратного устройства. (Только если проверка DMA активна.)

Бит 5 (0x20)
Приводит к включению сведений о блоке контекста ожидания.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о средстве проверки драйверов см. в документации по комплекту драйверов Windows (WDK). Дополнительные сведения о DMA см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем Дэвидом Соломоном.

# Замечания

Недопустимые аргументы (например, **!dma 1**) создают краткий текст справки.

**Если расширение !dma** используется без параметров, он отображает краткий список всех адаптеров DMA и их адресов. Это можно использовать для получения адреса адаптера для использования в более длинных версиях этой команды.

Ниже приведен пример использования этого расширения при активной проверке DMA средства проверки драйвера:

dbgcmd

```
0:kd> !dma

Dumping all DMA adapters...

Adapter: 82faebd0     Owner: SCSIPORT!ScsiPortGetUncachedExtension
Adapter: 82f88930     Owner: SCSIPORT!ScsiPortGetUncachedExtension
Adapter: 82f06cd0     Owner: NDIS!NdisMAllocateMapRegisters
Master adapter: 80076800
```

Из этих выходных данных вы увидите, что в системе есть три адаптера DMA. SCSIPORT владеет двумя и NDIS владеет третьим. Чтобы подробно изучить адаптер NDIS, используйте расширение !dma **со своим адресом**:

dbgcmd

```
0:kd> !dma  82f06cd0
Adapter: 82f06cd0     Owner: NDIS!NdisMAllocateMapRegisters (0x9fe24351)
 MasterAdapter:       00000000
   Adapter base Va      00000000
   Map register base:   00000000
 WCB:                 82f2b604
```

```
    Map registers: 00000000 mapped, 00000000 allocated, 00000002 max

    Dma verifier additional information:
    DeviceObject: 82f98690
    Map registers:        00000840 allocated, 00000000 freed
    Scatter-gather lists: 00000000 allocated, 00000000 freed
    Common buffers:       00000004 allocated, 00000000 freed
    Adapter channels:     00000420 allocated, 00000420 freed
    Bytes mapped since last flush: 000000f2
```

Первый блок данных — это конкретные сведения, которые разработчик HAL может использовать для отладки проблемы. Для ваших целей данные ниже "Dma verifier дополнительных сведений" — это то, что интересно. В этом примере вы увидите, что NDIS выделила 0x840 регистры карт. Это огромное число, особенно потому, что NDIS указал, что планируется использовать не более двух регистров карт. Этот адаптер, по-видимому, не использует списки точечной и сборной и не помещает все каналы адаптера. Дополнительные сведения см. в регистрах карты:

dbgcmd

```
0:kd> !dma  82f06cd0 2
Adapter: 82f06cd0     Owner: NDIS!NdisMAllocateMapRegisters
...

  Map register file 82f06c58 (0/2 mapped)
     Double buffer mdl: 82f2c188
     Map registers:
        82f06c80: Not mapped
        82f06c8c: Not mapped

  Map register file 82f06228 (1/2 mapped)
     Double buffer mdl: 82f1b678
     Map registers:
        82f06250: 00bc bytes mapped to f83c003c
        82f0625c: Not mapped

  Map register file 82fa5ad8 (1/2 mapped)
     Double buffer mdl: 82f1b048
     Map registers:
        82fa5b00: 0036 bytes mapped to 82d17102
        82fa5b0c: Not mapped
...
```

В этом примере вы увидите, что некоторые регистры карт сопоставлены. Каждый *файл* регистра карты — это выделение регистров карты драйвером. Иными словами, он представляет один вызов **в AllocateAdapterChannel**. NDIS собирает большое количество этих файлов регистра карты, а некоторые драйверы создают их одновременно и удаляют их после завершения.

Файлы регистра карты также являются адресами, возвращаемыми устройству под именем MapRegisterBase. Обратите внимание, что средство проверки DMA перехватывает только первые 64 регистров карты для каждого драйвера. Остальные игнорируются по причинам пространства (каждый регистр карты представляет три физических страницы).

В этом примере используются два файла регистра карты. Это означает, что драйвер сопоставляет буфер, чтобы он был видимым для оборудования. В первом случае 0xBC байты сопоставляются с системным виртуальным адресом 0xF83C003C.

Анализ общих буферов показывает:

dbgcmd

```
0:kd> !dma  82f06cd0 4
Adapter: 82f06cd0     Owner: NDIS!NdisMAllocateMapRegisters
...
   Common buffer allocated by NDIS!NdisMAllocateSharedMemory:
      Length:           1000
      Virtual address:  82e77000
      Physical address: 2a77000

   Common buffer allocated by NDIS!NdisMAllocateSharedMemory:
      Length:           12010
      Virtual address:  82e817f8
      Physical address: 2a817f8

   Common buffer allocated by NDIS!NdisMAllocateSharedMemory:
      Length:           4300
      Virtual address:  82e95680
      Physical address: 2a95680

   Common buffer allocated by NDIS!NdisMAllocateSharedMemory:
      Length:           4800
      Virtual address:  82e9d400
      Physical address: 2a9d400
```

Это довольно просто; Существует четыре общих буфера различной длины. Все физические и виртуальные адреса предоставляются.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Dpa

Статья • 05.04.2024

Расширение **!dpa** отображает сведения о выделении пула.

dbgcmd

```
!dpa Options
!dpa -?
```

# Параметры

*Параметры* должны быть именно одним из следующих параметров:

**-c**
Отображает текущую статистику выделения пула.

**-v**
Отображает все текущие выделения пула.

**-Vs**
Вызывает отображение трассировки стека.

**-f**
Отображает невыполненые выделения пула.

**-r**
Отображает выделения свободного пула.

**-p Ptr** ****
Отображает все выделения пула, содержащие указатель *Ptr*.

**-?**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Kdexts.dll

# Замечания

Инструментирование пула должно быть включено в Win32k.sys, чтобы это расширение работало.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !dpcs

Статья • 05.04.2024

Расширение **!dpcs** отображает очереди отложенных вызовов процедур (DPC) для указанного процессора.

dbgcmd

```
!dpcs [Processor]
```

## Параметры

*Процессор*
Указывает процессор. Если *обработчик* опущен, отображаются очереди DPC для всех процессоров.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Рекомендации недоступны |
| **Windows XP** | Рекомендации недоступны |
| **Windows Server 2003 и более поздних версий** | Kdexts.dll |

## Дополнительная информация

Дополнительные сведения о ЦП см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.


## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !driveinfo

Статья • 31.05.2024

Расширение **!driveinfo** отображает сведения о томе для указанного диска.

```
dbgcmd
```

```
!driveinfo Drive[:]
!driveinfo
```

## Параметры

*Диске*
Указывает диск. Двоеточие в конце имени диска является необязательным.

*Нет параметров*
Отображает краткий текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Kdexts.dll

## Замечания

Сведения о диске, отображаемые этим расширением, получаются путем запроса базовой файловой системы; Например:

```
dbgcmd
```

```
kd> !driveinfo c:
Drive c:, DriveObject e136cd88
    Directory Object: e1001408  Name: C:
        Target String is '\Device\HarddiskVolume1'
        Drive Letter Index is 3 (C:)
 Volume DevObj: 82254a68
    Vpb: 822549e0  DeviceObject: 82270718
 FileSystem: \FileSystem\Ntfs
    Volume has 0x229236 (free) / 0x2ee1a7 (total) clusters of size 0x1000
    8850.21 of 12001.7 MB free
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Драйверы

Статья • 05.04.2024

> **Примечание**
>
> В Windows XP и более поздних версиях Windows **расширение !drivers** устарело. Чтобы отобразить сведения о загруженных драйверах и других модулях, используйте **команду lm** .

Команда lm t n отображает сведения в формате, очень похожем на старое **расширение !drivers** . Однако эта команда не будет отображать использование памяти драйверов в качестве **расширения !drivers** . Он будет отображать только начальные и конечные адреса драйверов, имена изображений и метки времени. **Расширения !vm** и **!memusage** можно использовать для отображения статистики использования памяти.

dbgcmd

```
!drivers [Flags]
```

# Параметры

*Flags*
Может быть любое сочетание следующих значений. (Значение по умолчанию — 0x0.)

Bit 0 (0x1)
Приводит к включению сведений об резидентной и резервной памяти.

Бит 1 (0x2)
Если этот бит установлен, а бит 2 (0x4) не задан, то отображение будет содержать сведения об резидентной, резервной и заблокированной памяти, а также адрес записи загрузчика. Если установлен бит 2, это приводит к тому, что отображение будет гораздо длиннее и более подробный список образа драйвера. Сведения о заголовках включены, как и сведения о разделе.

Бит 2 (0x4)
Приводит к тому, что отображение будет гораздо длиннее и более подробным списком образа драйвера. Сведения о каждом разделе включены. Если задан бит 1 (0x2), это также будет содержать сведения о заголовке.

# DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Kdextx86.dll |
| **Windows XP и более поздние версии** | Рекомендации недоступны |

# Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Сведения о драйверах и использовании памяти см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Описание отображения этой команды приведено в следующей таблице:

Развернуть таблицу

| **Столбец** | **Значение** |
|---|---|
| База | Начальный адрес кода драйвера устройства в шестнадцатеричном формате. Если адрес памяти, используемый кодом, который приводит к остановке между базовым адресом драйвера и базовым адресом для следующего драйвера в списке, этот драйвер часто вызывает ошибку. Например, база для Ncrc810.sys 0x80654000. Любой адрес между этим и 0x8065a000 принадлежит этому драйверу. |
| Размер кода | Размер кода драйвера в килобайтах в шестнадцатеричном и десятичном. |
| Размер данных | Объем пространства в килобайтах, выделенный драйверу для данных в шестнадцатеричном и десятичном формате. |
| Заблокировано | (Только если используется флаг 0x2) Объем памяти, заблокированной драйвером. |

| Столбец | Значение |
| --- | --- |
| Резидент | (Только если используется флаг 0x1 или 0x2) Объем памяти драйвера, который фактически находится в физической памяти. |
| Режим ожидания | (Только если используется флаг 0x1 или 0x2) Объем памяти драйвера, которая находится в режиме ожидания. |
| Запись загрузчика | (Только если используется флаг 0x2) Адрес записи загрузчика. |
| Driver Name | Имя файла драйвера. |
| Время создания | Дата ссылки драйвера. Не путайте это с датой файла драйвера, которую можно задать внешними средствами. Дата ссылки задается компилятором при компиляции драйвера или исполняемого файла. Оно должно быть близко к дате файла, но оно не всегда совпадает. |

Ниже приведен усеченный пример этой команды:

dbgcmd

```
kd> !drivers
Loaded System Driver Summary
Base     Code Size      Data Size      Driver Name  Creation Time
80080000 f76c0 (989 kb) 1f100 (124 kb) ntoskrnl.exe Fri May 26 15:13:00
80400000 d980  ( 54 kb) 4040  ( 16 kb) hal.dll      Tue May 16 16:50:34
80654000 3f00  ( 15 kb) 1060   ( 4 kb) ncrc810.sys  Fri May 05 20:07:04
8065a000 a460  ( 41 kb) 1e80   ( 7 kb) SCSIPORT.SYS Fri May 05 20:08:05
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !drvobj

Статья • 05.04.2024

Расширение **!drvobj** отображает подробные сведения о DRIVER_OBJECT.

```
dbgcmd
```

```
!drvobj DriverObject [Flags]
```

## Параметры

*DriverObject*
Указывает объект драйвера. Это может быть шестнадцатеричный адрес структуры DRIVER_OBJECT или имени драйвера.

*Flags*
Может быть любое сочетание следующих битов. (Значение по умолчанию — 0x01.)

Bit 0 (0x1)
Приводит к включению объектов устройств, принадлежащих драйверу.

Бит 1 (0x2)
Вызывает включение точек входа для подпрограмм отправки драйвера.

Бит 2 (0x4)
Содержит подробные сведения об объектах устройства, принадлежащих драйверу (требуется бит 0 (0x1)).

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Примеры и приложения этой команды расширения см. в самонастраивающийся отладке. Дополнительные сведения об объектах драйверов см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Если *DriverObject* указывает имя устройства, но не предоставляет префикса, предполагается префикс \Driver\. Обратите внимание, что эта команда проверка, чтобы узнать, является ли *DriverObject* допустимым адресом или именем устройства перед использованием средства оценки выражений.

Если *DriverObject* является адресом, он должен быть адресом DRIVER_OBJECT структуры. Это можно получить, проверив аргументы, переданные в подпрограмму DriverEntry **драйвера**.

Эта команда расширения отобразит список всех объектов устройств, созданных указанным драйвером. Он также будет отображать все быстрые подпрограммы ввода-вывода, зарегистрированные в этом объекте драйвера.

Ниже приведен пример для драйвера мини-порта Sym биография s Logic 810 SCSI:

dbgcmd

```
kd> bp DriverEntry          //  breakpoint at DriverEntry

kd> g
symc810!DriverEntry+0x40:
80006a20: b07e0050 stl     t2,50(sp)

kd> r a0  //address of DevObj (the first parameter)
a0=809d5550

kd> !drvobj 809d5550   //  display the driver object
Driver object is for:
\Driver\symc810
Device Object list:
809d50d0
```

Вы также можете использовать **!devobj 809d50d0** для получения сведений об объекте устройства.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !dskheap

Статья • 05.04.2024

Расширение **!dskheap** отображает сведения о куче рабочего стола для указанного сеанса.

dbgcmd

```
!dskheap [-v] [-s SessionID]
```

## Параметры

**-v**
Приводит к тому, что отображение включает более подробные выходные данные.

**-s** **** *SessionID*
Задает сеанс. Если этот параметр опущен, отображается информация о кучи рабочего стола для сеанса 0.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о рабочих столах или кучи настольных компьютеров см. в документации по пакету SDK для Microsoft Windows и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Сведения о кучи рабочего стола для сеанса организованы по станции окон.

Вот несколько примеров.

dbgcmd

```
kd> !dskheap -s 3
##   Winstation\Desktop            Heap Size(KB)   Used Rate(%)
```

```
  WinSta0\Screen-saver              3072                 0%
  WinSta0\Default                   3072                 0%
  WinSta0\Disconnect                  64                 4%
##   WinSta0\Winlogon                  128                    5%

                Total Desktop: (    6336 KB -   4 desktops)
#                 Session ID:  3

kd> !dskheap
##   Winstation\Desktop           Heap Size(KB)   Used Rate(%)

  WinSta0\Default                   3072                 0%
  WinSta0\Disconnect                  64                 4%
  WinSta0\Winlogon                   128                 9%
  Service-0x0-3e7$\Default           512                 4%
  Service-0x0-3e5$\Default           512                 0%
  Service-0x0-3e4$\Default           512                 1%
##   SAWinSta\SADesktop                512                    0%

                Total Desktop: (    5312 KB -   7 desktops)
#                 Session ID:  0
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !eb, !ed

Статья • 08.07.2024

Расширения **!eb** и **!ed** записывают последовательность значений в указанный физический адрес.

Эти команды расширения не следует путать с командой **e* (ввод значений).**

dbgcmd

```
!eb [Flag] PhysicalAddress Data [ ... ]
!ed [Flag] PhysicalAddress Data [ ... ]
```

# Параметры

*Флаг*
Может быть любым из следующих значений. *Значение флага* должно быть окружено квадратными скобками:

**[c]**
Записывает в кэшированную память.

**[uc]**
Записывает в некичированную память.

**[wc]**
Записывает в объединенную память записи.

*PhysicalAddress*
Указывает первый физический адрес на целевом компьютере, на котором будут записываться данные в шестнадцатеричном виде.

*Данные*
Указывает одно или несколько значений, которые записываются последовательно в физическую память. Введите эти значения в шестнадцатеричном формате. Для расширения !EB **каждое** значение должно быть 1 байт (две шестнадцатеричные цифры). Для расширения !ed **каждое** значение должно быть одним DWORD (восемь шестнадцатеричных цифр). Можно включить любое количество значений *данных* в одной строке. Чтобы разделить несколько значений, используйте запятые или пробелы.

# DLL-библиотеки

Kext.dll

# Дополнительная информация

Для чтения физической **памяти используйте расширения !d*** . Обзор операций с памятью и описание других команд, связанных с памятью, см. в статье "Чтение и запись памяти".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ecb, !ecd, !ecw

Статья • 08.07.2024

Расширения **!ecb**, **!ecd** и **!ecw** записываются в пространство конфигурации PCI.

```
dbgcmd
```

```
!ec Bus.Device.Function Offset Data
```

## Параметры

*Автобус*
Указывает шину. *Шина* может варьироваться от 0 до 0xFF.

*Устройство*
Указывает номер устройства слота для устройства.

*Функция*
Указывает номер функции слота для устройства.

*Смещение*
Указывает адрес, по которому требуется написать.

*Данные*
Указывает значение для записи. **Для расширения *!ecb*** *данные* должны быть 1 байтами (две шестнадцатеричные цифры). **Для расширения *!ecw*** *данные* должны быть одним словом WORD (четыре шестнадцатеричные цифры). **Для расширения *!ecd*** *данные* должны быть одним DWORD (восемь шестнадцатеричных цифр).

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Kext.dll |
| **Windows XP и более поздние версии** | Kext.dll |

Эти команды расширения можно использовать только с целевым компьютером на основе x86.

# Дополнительная информация

Дополнительные примеры см. в разделе самонастраивающийся Отладка для приложений этой команды расширения. Сведения о шинах PCI см. в документации по комплекту драйверов Windows (WDK).

# Замечания

Эти команды расширения нельзя использовать для записи последовательности значений *данных* . Это можно сделать только путем многократного использования этого расширения.

Чтобы отобразить пространство конфигурации PCI, используйте функцию *устройства шины !pci 100*.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ecs

Статья • 26.10.2023

Расширение **!ecs** устарело. Чтобы изменить пространство конфигурации PCI, используйте **!ecb**, **!ecd** или **!ecw**.

# !errlog

Статья • 05.04.2024

Расширение **!errlog** отображает содержимое всех ожидающих записей в журнале ошибок системы ввода-вывода.

dbgcmd

```
!errlog
```

## DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о **IoWriteErrorLogEntry** см. в документации по комплекту драйверов Windows (WDK).

# Замечания

Эта команда отображает сведения о любых ожидающих событиях в журнале ошибок системы ввода-вывода. Это события, очередные вызовы **функции IoWriteErrorLogEntry**, записываемые в журнал событий системы для последующего просмотра **Просмотр событий**.

Отображаются только записи, которые были помещены в очередь IoWriteErrorLogEntry, но не зафиксированы в журнале ошибок.

Эта команда может использоваться в качестве средства диагностики после сбоя системы, так как она показывает ожидающие сведения об ошибке, которые не удалось зафиксировать в журнал ошибок до остановки системы.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !errpkt

Статья • 05.04.2024

Расширение **!errpkt** отображает содержимое пакета ошибок оборудования Windows (WHEA).

```
dbgcmd
```

```
!errpkt Address
```

## Параметры

*Адрес*
Указывает адрес пакета ошибок оборудования.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Рекомендации недоступны |
| **Windows XP** | Рекомендации недоступны |
| **Windows Server 2003** | Рекомендации недоступны |
| **Windows Vista и более поздние версии** | Kdexts.dll |

Это расширение можно использовать только в Windows Vista и более поздних версиях Windows.

## Дополнительная информация

Расширения !whea **и !errrec можно использовать для отображения дополнительных сведений о WHEA.** Общие сведения об WHEA см . в документации по архитектуре аппаратных ошибок Windows (WHEA) в комплекте драйверов Windows (WDK).

## Замечания

В следующем примере показаны выходные данные **расширения !errpkt** :

dbgcmd

```
3: kd> !errpkt fffffa8007cf44da
   WHEA Error Packet Info Section (@ fffffa8007cf44da)
   Flags            : 0x00000000
   Size             : 0x218
   RawDataLength    : 0x392
   Context          : 0x0000000000000000
   ErrorType        : 0x0 - Processor
   ErrorSeverity    : 0x1 - Fatal
   ErrorSourceId    : 0x0
   ErrorSourceType  : 0x0 - MCE
   Version          : 00000002
   Cpu              : 0000000000000002
   RawDataFormat    : 0x2 - Intel64 MCA

   Raw Data         : Located @ FFFFFA8007CF45F2

Processor Error: (Internal processor error)
This error means either the processor is damaged or perhaps
voltage and/or temperature thresholds have been exceeded.
If the problem continues to occur, replace the processor.
Processor Number : 2
Bank Number      : 0
   Status  :               0
   Address : 0000000000000000 (I)
   Misc    : 0000000000000000 (I)
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !errrec

Статья • 05.04.2024

Расширение **!errrec** отображает содержимое записи об ошибках архитектуры оборудования Windows (WHEA).

dbgcmd

```
!errrec Address
```

## Параметры

*Адрес*
Указывает адрес записи об ошибке.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Рекомендации недоступны |
| **Windows XP** | Рекомендации недоступны |
| **Windows Server 2003** | Рекомендации недоступны |
| **Windows Vista и более поздние версии** | Kdexts.dll |

Это расширение можно использовать только в Windows Vista и более поздних версиях Windows.

## Дополнительная информация

Расширения !whea **и !errpkt можно использовать для отображения дополнительных сведений WHEA.** Общие сведения об WHEA см . в документации по архитектуре аппаратных ошибок Windows (WHEA) в комплекте драйверов Windows (WDK).

## Замечания

В следующем примере показано, как расширение !whea **можно использовать для получения адреса записи об ошибке, а затем содержимое этой записи может отображаться расширением !errrec:**

dbgcmd

```
3: kd> !whea
Error Source Table @ fffff800019ca250
13 Error Sources
Error Source 0 @ fffffa80064132c0
   Notify Type      : Machine Check Exception
   Type             : 0x0 (MCE)
   Error Count      : 8
   Record Count     : 10
   Record Length    : c3e
   Error Records    : wrapper @ fffffa8007cf4000  record @ fffffa8007cf4030

. . . .

# 3: kd> !errrec fffffa8007cf4030
===============================================================================
===
## Common Platform Error Record @ fffffa8007cf4030
-------------------------------------------------------------------------------
---
Revision      : 2.1
Record Id     : 01c9c7ff04e0ff7d
Severity      : Fatal (1)
Length        : 1730
Creator       : Microsoft
Notify Type   : Machine Check Exception
Timestamp     : 4/28/2009 12:54:47
Flags         : 0x00000000
#
===============================================================================
===
## Section 0     : Processor Generic
-------------------------------------------------------------------------------
---
Descriptor    @ fffffa8007cf40b0
Section       @ fffffa8007cf4188
Offset        : 344
Length        : 192
Flags         : 0x00000001 Primary
Severity      : Fatal

No valid data fields are present.
#
===============================================================================
===
## Section 1     : {390f56d5-ca86-4649-95c4-73a408ae5834}
-------------------------------------------------------------------------------
---
```

```
Descriptor    @ fffffa8007cf40f8
Section       @ fffffa8007cf4248
Offset        : 536
Length        : 658
Flags         : 0x00000000
Severity      : Fatal

*** Unknown section format ***
#
================================================================================
===
## Section 2     : Error Packet
--------------------------------------------------------------------------------
---
Descriptor    @ fffffa8007cf4140
Section       @ fffffa8007cf44da
Offset        : 1194
Length        : 536
Flags         : 0x00000000
Severity      : Fatal

   WHEA Error Packet Info Section (@ fffffa8007cf44da)
   Flags            : 0x00000000
   Size             : 0x218
   RawDataLength    : 0x392
   Context          : 0x0000000000000000
   ErrorType        : 0x0 - Processor
   ErrorSeverity    : 0x1 - Fatal
   ErrorSourceId    : 0x0
   ErrorSourceType  : 0x0 - MCE
   Version          : 00000002
   Cpu              : 0000000000000002
   RawDataFormat    : 0x2 - Intel64 MCA

   Raw Data         : Located @ FFFFFA8007CF45F2

Processor Error: (Internal processor error)
This error means either the processor is damaged or perhaps
voltage and/or temperature thresholds have been exceeded.
If the problem continues to occur, replace the processor.
Processor Number : 2
Bank Number      : 0
   Status  :               0
   Address : 0000000000000000 (I)
   Misc    : 0000000000000000 (I)
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !exca

Статья • 05.04.2024

Расширение **!exca** отображает регистры контроллера прерываний pc-card (PCIC) с возможностью обмена карточками (ExCA).

dbgcmd

```
!exca BasePort.SocketNumber
```

## Параметры

*BasePort*
Указывает базовый порт PCIC.

*SocketNumber*
Указывает номер сокета регистра ExCA в PCIC.

## DLL-библиотеки

Kext.dll

**Расширение !exca** доступно только для целевого компьютера на основе x86.

## Дополнительная информация

**Расширение !cbreg** можно использовать для отображения регистров сокета CardBus и регистров CardBus ExCA по адресу.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !filecache

Статья • 05.04.2024

Расширение **!filecache** отображает сведения об использовании памяти кэша файлов системы и PTE.

```
dbgcmd
```

```
!filecache [Flags]
```

## Параметры

*Flags*
Необязательно. Значение по умолчанию — 0x0. Задайте *флаги для 0x1 сортировки выходных* данных по карте общего кэша. Таким образом можно найти все представления системного кэша для заданной области управления.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о драйверах файловой системы см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Каждая строка выходных данных этого расширения представляет блок управления виртуальными адресами (VACB). При сопоставлении именованных файлов с VACB отображаются имена этих файлов. Если указано "имя файла", это означает, что этот VACB используется для кэширования метаданных.

Ниже приведен пример выходных данных этого расширения из системы Windows XP:

```
dbgcmd
```

```
kd> !filecache
***** Dump file cache******
  Reading and sorting VACBs ...
  Removed 1811 nonactive VACBs, processing 235 active VACBs ...
File Cache Information
  Current size 28256 kb
  Peak size    30624 kb
  235 Control Areas
Skipping view @ c1040000 - no VACB, but PTE is valid!
  Loading file cache database (100% of 131072 PTEs)
  SkippedPageTableReads = 44
  File cache has 4056 valid pages


  Usage Summary (in Kb):
Control Valid Standby/Dirty Shared Locked Name
817da668     4      0     0     0  $MftMirr
8177ae68   304    920     0     0  $LogFile
81776160   188      0     0     0  $BitMap
817cf370     4      0     0     0  $Mft
81776a00     8      0     0     0  $Directory
817cfdd0     4      0     0     0  $Directory
81776740    36      0     0     0    No Name for File
817cf7c8    20      0     0     0  $Directory
817cfb98   304      0     0     0  $Directory
8177be40    16      0     0     0  $Directory
817dadc0  2128     68     0     0  $Mft
817cf008     4      0     0     0  $Directory
817d0258     8      4     0     0  $Directory
817763f8     4      0     0     0  $Directory
...
8173f058     4      0     0     0  $Directory
8173e628    32      0     0     0  $Directory
8173e4c8    32      0     0     0  $Directory
8173da38     4      0     0     0  $Directory
817761f8     4      0     0     0  $Directory
81740530    32      0     0     0  $Directory
8173d518     4      0     0     0  $Directory
817d9560     8      0     0     0  $Directory
8173f868     4      0     0     0  $Directory
8173fc00     4      0     0     0  $Directory
81737278     4      0     0     0  $MftMirr
81737c88    44      0     0     0  $LogFile
81735fa0    48      0     0     0  $Mft
81737e88   188      0     0     0  $BitMap
817380b0     4      0     0     0  $Mft
817399e0     4      0     0     0  $Directory
817382b8     4      0     0     0  $Directory
817388d8    12      0     0     0    No Name for File
81735500     8      0     0     0  $Directory
81718e38   232      0     0     0  default
81735d40    48     20     0     0  SECURITY
81723008  8632      0     0     0  software
816da3a0    24     44     0     0  SAM
```

```
8173dfa0     4      0     0     0  $Directory
...
8173ba90     4      0     0     0  $Directory
8170ee30     4     36     4     0  AppEvent.Evt
816223f8     4      0     0     0  $Directory
8170ec28     8     28     4     0  SecEvent.Evt
816220a8     4      0     0     0  $Directory
8170ea20     4     32     4     0  SysEvent.Evt
8170d188   232      0     0     0  NTUSER.DAT
81709f10     8      0     0     0  UsrClass.dat
81708918   232      0     0     0  NTUSER.DAT
81708748     8      0     0     0  UsrClass.dat
816c58f8    12      0     0     0  change.log
815c3880     4      0     0     0  $Directory
81706aa8     4      0     0     0  SchedLgU.Txt
815ba2d8     4      0     0     0  $Directory
815aa5f8     8      0     0     0  $Directory
8166d728    44      0     0     0  Netlogon.log
81701120     8     16     4     0  es.dll
816ff0a8     4      8     4     0  stdole2.tlb
8159a358     4      0     0     0  $Directory
8159da70     4      0     0     0  $Directory
8159c158     4      0     0     0  $Directory
815cb9b0     4      0     0     0  00000001
81779b20     4      0     0     0  $Directory
8159ac20     4      0     0     0  $Directory
815683f8     4      0     0     0  $Directory
81566978   580      0     0     0  NTUSER.DAT
81568460     4      0     0     0  $Directory
815675d8    68      0     0     0  UsrClass.dat
81567640     4      0     0     0  $Directory
...
81515878     4      0     0     0  $Directory
81516870     8      0     0     0  $Directory
8150df60     4      0     0     0  $Directory
...
816e5300     4      0     0     0  $Directory
8152afa0    16    212     0     0  msmsgs.exe
8153bbd8     4     32     0     0  stdole32.tlb
8172f950   488    392     0     0  OBJECTS.DATA
8173e9c0     4      0     0     0  $Directory
814f4538     4      0     0     0  $Directory
81650790   344     48     0     0  INDEX.BTR
814f55f8     4      0     0     0  $Directory
...
814caef8     4      0     0     0  $Directory
8171cd90  1392     36     0     0  system
815f07f0     4      0     0     0  $Directory
814a2298     4      0     0     0  $Directory
81541538     4      0     0     0  $Directory
81585288    28      0     0     0  $Directory
8173f708     4      0     0     0  $Directory
...
8158cf10     4      0     0     0  $Directory
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !filelock

Статья • 05.04.2024

Расширение **!filelock** отображает структуру блокировки файлов.

Синтаксис

dbgcmd

```
!filelock FileLockAddress
!filelock ObjectAddress
```

## Параметры

*FileLockAddress*
Указывает шестнадцатеричный адрес структуры блокировки файла.

*ObjectAddress*
Указывает шестнадцатеричный адрес объекта файла, которому принадлежит блокировка файла.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения об объектах файлов см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !fileobj

Статья • 05.04.2024

Расширение **!fileobj** отображает подробные сведения о структуре FILE_OBJECT.

```
dbgcmd
```

```
!fileobj FileObject
```

## Параметры

*FileObject*
Задает адрес структуры FILE_OBJECT .

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения об объектах файлов см. в документации по пакету SDK для Microsoft Windows, документации по комплекту драйверов Windows (WDK) и *внутренним* службам Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Если структура FILE_OBJECT имеет связанный кэш, **!fileobj** пытается проанализировать и отобразить сведения о кэше.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !filetime

Статья • 08.07.2024

**Расширение !filetime** преобразует 64-разрядную структуру FILETIME в удобочитаемое время.

dbgcmd

```
!filetime Time
```

## Параметры

*Время*
Указывает 64-разрядную структуру FILETIME.

### DLL-библиотеки

Kdexts.dll

## Замечания

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
kd> !filetime 1c4730984712348
 7/26/2004 04:10:18.712 (Pacific Standard Time)
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !finddata

Статья • 05.04.2024

Расширение **!finddata** отображает кэшированные данные в заданном смещение в указанном объекте файла.

```
dbgcmd
```

```
!finddata FileObject Offset
```

## Параметры

*FileObject*
Указывает адрес объекта файла.

*Offset*
Указывает смещение.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения об управлении кэшем см. в документации по microsoft Windows SDK и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

Дополнительные сведения о других расширениях управления кэшем см. в расширении !cchelp.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !findfilelockowner

Статья • 05.04.2024

**Расширение !findfilelockowner** пытается найти владельца блокировки файлов, проверив все потоки для потока, заблокированного в утверждении IopSynchronousServiceTail и имеющий объект файла в качестве параметра.

dbgcmd

```
!findfilelockowner [FileObject]
```

## Параметры

*FileObject*

Указывает адрес объекта файла. Если *файлObject опущен* , расширение ищет любой поток в текущем процессе, который завис в **IopAcquireFileObjectLock** и извлекает адрес объекта файла из трассировки стека.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения об объектах файлов см. в документации по пакету SDK для Microsoft Windows, документации по комплекту драйверов Windows (WDK) и *внутренним* службам Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Это расширение наиболее полезно после истечения времени ожидания критического раздела, в котором поток, в течение которого истекло время ожидания файла в **IopAcquireFileObjectLock**. После обнаружения обижающего потока расширение пытается восстановить IRP, который использовался для запроса, и отобразить драйвер, обрабатывающий IRP.

Расширение занимает некоторое время, так как он проходит стек всех потоков в системе, пока он не обнаружит обидующий поток. Вы можете остановить 'в любой момент, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

## См. также

Отображение критического раздела

Критические истечение времени ожидания разделов (режим пользователя)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !findthreads

Статья • 08.07.2024

Расширение !findthreads отображает сводную информацию об одном или нескольких потоках целевой системы на основе предоставленных условий поиска. Сведения о потоке будут отображаться, когда связанные стеками ссылаются на предоставленный объект. Эта команда может использоваться только во время отладки в режиме ядра.

## Синтаксис

dbgcmd

```
!findthreads [-v][-t  <Thread Address>|-i <IRP Address>|-d <Device Address>|
( -a <Pointer Address> -e <End Address> | -l <Range Length>)]
```

## Параметры

**-v**
Отображает подробные сведения обо всех совпадениях условий.

**-t** **** *Адрес потока*
Критерии поиска — все модули, объекты ожидания и irPs для потока, а также объекты и модули устройств, созданные из подключенных IRPs. Этот параметр обычно предоставляет самые широкие критерии поиска.

**-я** **** *Адрес IRP*
Условия поиска будут всеми модулями и устройствами для указанного IRP, а также любой ссылкой на сам IRP.

**-d** **** *Адрес устройства*
Условия поиска будут основаны на объекте устройства. Это будет включать модуль, связанный с объектом устройства (через объект драйвера), расширение устройства, любое IRP, подключенное к устройству, и аналогичную информацию для любого подключенного к объекту устройства.

**-a** **** *Адрес-указатель*
Добавьте базовый адрес в критерии. Если значение -e или -l правильно указано, это значение будет основой диапазона памяти. В противном случае он интерпретируется как указатель.

**-e** **** *Конечный адрес*

Указывает конечный адрес диапазона памяти, указанного с помощью -a.

**-l** **** *Длина диапазона*

Указывает длину диапазона памяти, указанного с помощью -a.

# DLL-библиотеки

Windows 10 и более поздние версии

Kdexts.dll

# Дополнительная информация

Сведения о потоках в режиме ядра см. в разделе "Изменение контекстов". Дополнительные сведения об анализе процессов и потоков см. в разделе "*Внутренние* компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

# Замечания

Ниже приведен пример выходных данных с помощью параметров -v и -t:

dbgcmd

```
kd> !findthreads -v -t ffffd001ee29cdc0

Added criterion for THREAD 0xffffd001ee29cdc0
  Added criterion for THREAD STACK 0xffffd001ee2bac20
  ERROR: Object 0xfffffffffffffffe0 is not an IRP
ERROR: unable to completely walk thread IRP list.
  Added criterion for MODULE kdnic(0xfffff80013120000)

Found 63 threads matching the search criteria

Found 6 criteria matches for THREAD 0xffffe0016a383740, PROCESS
0xffffe0016a220200
  Kernel stack location 0xffffd001f026a0c0 references THREAD
0xffffd001ee29cdc0
  Kernel stack location 0xffffd001f026a418 references THREAD
0xffffd001ee29cdc0
  Kernel stack location 0xffffd001f026a460 references THREAD
0xffffd001ee29cdc0
  Kernel stack location 0xffffd001f026a4d0 references THREAD
0xffffd001ee29cdc0
  Kernel stack location 0xffffd001f026a4f0 references THREAD
0xffffd001ee29cdc0
  Kernel stack location 0xffffd001f026a670 references THREAD
```

```
0xffffd001ee29cdc0


    ffffd001f026a0e0 nt!KiSwapContext+76
    ffffd001f026a190 nt!KiSwapThread+1c8
    ffffd001f026a220 nt!KiCommitThreadWait+148
    ffffd001f026a2e0 nt!KeWaitForMultipleObjects+21e
    ffffd001f026a800 nt!ObWaitForMultipleObjects+2b7
    ffffd001f026aa80 nt!NtWaitForMultipleObjects+f6
    000000c8d220fa98 nt!KiSystemServiceCopyEnd+13
    000000c8d220fa98 ntdll!ZwWaitForMultipleObjects+a
...
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !for_each_process

Статья • 05.04.2024

**Расширение !for_each_process** выполняет указанную команду отладчика один раз для каждого процесса в целевом объекте.

dbgcmd

```
!for_each_process ["CommandString"]
!for_each_process -?
```

## Параметры

*CommandString*
Указывает команды отладчика, выполняемые для каждого процесса.

Если *CommandString* включает несколько команд, разделите их точкой с запятой (;) и заключите *CommandString* в кавычки ("). Если *CommandString* заключен в кавычки, отдельные команды в *CommandString* не могут содержать кавычки. В *CommandString* **@#Process** заменяется адресом процесса.

**-?**
Отображает справку по этому расширению в окне команды отладчика.

## DLL-библиотеки

Это расширение работает только в режиме ядра, даже если оно находится в Ext.dll.

Ext.dll

## Дополнительная информация

Общие сведения о процессах см. в разделе "Потоки и процессы". Сведения об управлении или получении сведений о процессах см. в разделе "Управление процессами и потоками".

## Замечания

Если аргументы не указаны, отладчик отображает список всех процессов, а также статистику по времени и приоритету. Это эквивалентно вводу **@process @#Process**

**0** в качестве *значения CommandString* .

Чтобы завершить выполнение в любой момент, нажмите клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !for_each_thread

Статья • 05.04.2024

**Расширение !for_each_thread** выполняет указанную команду отладчика один раз для каждого потока в целевом объекте.

```
dbgcmd
```

```
!for_each_thread ["CommandString"]
!for_each_thread -?
```

## Параметры

*CommandString*
Указывает команды отладчика, выполняемые для каждого потока. Если *CommandString* включает несколько команд, разделите их точкой с запятой (;) и заключите *CommandString* в кавычки ("). Если *CommandString* заключен в кавычки, отдельные команды в *CommandString* не могут содержать кавычки. В *CommandString* **@#Thread** заменяется адресом потока.

**-?**
Отображает справку по этому расширению в окне команды отладчика.

## DLL-библиотеки

Это расширение работает только в режиме ядра, даже если оно находится в Ext.dll.

Ext.dll

## Дополнительная информация

Дополнительные сведения о потоках см. в разделе "Потоки и процессы". Дополнительные сведения об управлении потоками или получении сведений см. в разделе "Управление процессами и потоками".

## Замечания

Если аргументы не указаны, отладчик отображает список всех потоков, а также состояния ожидания потока. Это эквивалентно вводу **!thread @#Thread 2** в

качестве *значения CommandString* .

Вы можете завершить выполнение в любой момент, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !fpsearch

Статья • 08.07.2024

Расширение **!fpsearch** ищет освобожденный специальный пул для указанного адреса.

```
dbgcmd
```

```
!fpsearch [Address] [Flag]
```

# Параметры

*Адрес*
Указывает виртуальный адрес.

*Флаг*
Если задано, отладчик отображает необработанное содержимое каждой страницы в свободном списке, так как он ищет освобожденный специальный пул.

## DLL-библиотеки

Kdexts.dll

# Замечания

Отображение адреса включает виртуальный адрес, номер кадра страницы (PFN), тег пула, размер, размер данных на адресе, идентификатор потока и стек вызовов во время размещения сделки.

Если *для адреса* задано значение -1, отладчик отображает весь освобожденный специальный пул.

Если отладчик не может найти указанный адрес в освобожденном специальном пуле, он не отображает ничего. Ниже приведен пример выходных данных из этого расширения:

```
dbgcmd
```

```
kd> !fpsearch -1 1
Searching the free page list (8 entries) for all freed special pool
```

```
1EC4000  04000200 e56b6f54 000001f4 0000059c ....Tok.........
1EC4000  00000800 00000000 00000000 00000000 ................
1EC4000  bad0b0b0 82100000 00000000 00000000 ................
1EC4000  72657355 20203233 0000bac5 00000000 User32  ........
1EC4000  00028b94 00000000 0000bac9 00000000 ................
1EC4000  00000000 00000000 ffffffff 7fffffff ................
1EC4000  8153b1b8 00028aff 00000000 00000000 ..S.............
1EC4000  0000001b 00000000 00000012 00000514 ................

26A2000  000a0008 00adecb0 000e000c 00adecba ................
26A2000  000a0008 00adecc8 000e000c 00adecd2 ................
26A2000  000e000c 00adece0 000e000c 00adecee ................
26A2000  00120010 00adecfc 000e000c 00aded0e ................
26A2000  000e000c 00aded1c 000e000c 00aded2a ............*...
26A2000  000e000c 00aded38 000e000c 00aded46 ....8.......F...
26A2000  000a0008 00aded54 000e000c 00aded5e ....T.......^...
26A2000  00120010 00aded6c 000e000c 00aded7e ....l.......~...

2161000  000a0008 00adeccc 000e000c 00adecd6 ................
2161000  000a0008 00adece4 000e000c 00adecee ................
2161000  000e000c 00adecfc 000e000c 00aded0a ................
2161000  00120010 00aded18 000e000c 00aded2a ............*...
2161000  000e000c 00aded38 000e000c 00aded46 ....8.......F...
2161000  000e000c 00aded54 000e000c 00aded62 ....T.......b...
2161000  000a0008 00aded70 000e000c 00aded7a ....p.......z...
2161000  00120010 00aded88 000e000c 00aded9a ................

...

CEC8000  0311ffa4 03120000 0311c000 00000000 ................
CEC8000  00001e00 00000000 7ff88000 00000000 ................
CEC8000  00000328 00000704 00000000 00000000 (...............
CEC8000  7ffdf000 00000000 00000000 00000000 ................
CEC8000  e18ba8c0 00000000 00000000 00000000 ................
CEC8000  00000000 00000000 00000000 00000000 ................
CEC8000  00000000 00000000 00000000 00000000 ................
CEC8000  00000000 00000000 00000000 00000000 ................

CEAD000  00000000 00000000 00000000 00000000 ................
CEAD000  00000000 00000000 00000000 00000000 ................
CEAD000  00000000 00000000 00000000 00000000 ................
CEAD000  00000000 00000000 00000000 00000000 ................
CEAD000  00000000 00000000 00000000 00000000 ................
CEAD000  00000000 00000000 00000000 00000000 ................
CEAD000  00000000 00000000 00000000 00000000 ................
CEAD000  00000000 00000000 00000000 00000000 ................
```

Вы можете остановить выполнение в любой момент, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Замороженные

Статья • 05.04.2024

Расширение **!frozen** отображает состояние каждого процессора.

```
dbgcmd
```

```
!frozen
```

## DLL-библиотеки

Kdexts.dll

## Замечания

Ниже приведен пример выходных данных из этого расширения:

```
dbgcmd
```

```
0: kd> !frozen
Processor states:
       0 : Current
       1 : Frozen
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !fwver

Статья • 05.04.2024

Расширение **!fwver** отображает версию встроенного ПО Itanium.

dbgcmd

```
!fwver
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером Itanium.

## Дополнительная информация

Дополнительные сведения см. в руководстве по архитектуре Intel.

## Замечания

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
kd> !fwver

Firmware Version

   Sal Revision:        0
   SAL_A_VERSION:       0
   SAL_B_VERSION:       0
   PAL_A_VERSION:       6623
   PAL_B_VERSION:       6625
   smbiosString:        W460GXBS2.86E.0117A.P08.200107261041
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !fxdevice

Статья • 05.04.2024

Расширение !fxdevice отображает сводные сведения обо всех зарегистрированных устройствах Power Management Framework (PoFx). Эта команда может использоваться только во время отладки в режиме ядра.

Дополнительные сведения о PoFX см. в разделе "Обзор платформы управления питанием".

Синтаксис

dbgcmd

```
!fxdevice[<FxDevice Address>]
```

## Параметры

**< Адрес FxDevice>**
Предоставляет адрес отображаемого объекта FxDevice.

## DLL-библиотеки

Kdexts.dll

## Замечания

Расширение !fxdevice отображает следующие сведения при наличии в целевой системе.

- Неактивные устройства PoFx
- Неактивные устройства D0 PoFx
- Неактивные устройства, отличные от D0 PoFx

Ниже приведен пример из расширения !fxdevice с указанным адресом устройства.

dbgcmd

```
kd> !fxdevice ffffe0012ccbda60
!fxdevice 0xffffe0012ccbda60
    DevNode: 0xffffe0012bbb09f0
```

```
    UniqueId:
"HDAUDIO\FUNC_01&VEN_10EC&DEV_0662&SUBSYS_103C304A&REV_1001\4&25ff998c&0&000
1"
    InstancePath:
"HDAUDIO\FUNC_01&VEN_10EC&DEV_0662&SUBSYS_103C304A&REV_1001\4&25ff998c&0&000
1"
    Device Power State: PowerDeviceD0
    PEP Owner: Default PEP
    Acpi Plugin: 0
    Acpi Handle: 0
    Device Status Flags: IdleTimerOn DevicePowerNotRequired_ReceivedFromPEP
    Device Idle Timeout: 0x1869ffffe7960
    Device Power On: No Activity
    Device Power Off: No Activity
    Device Unregister: No Activity
    Component Count: 1
        Component 0: F0/F0 - IDLE   (RefCount = 0)
        Pep Component: 0xffffe0012cfe1800
            Active: 0   Latency: 0  Residency: 0    Wake: 0 Dx IRP: 0   WW
IRP: 0
            Component Idle State Change: No Activity
            Component Activation: No Activity
            Component Active: No Activity
```

Ниже приведены выходные данные по умолчанию из расширения !fxdevice.

dbgcmd

```
kd> !fxdevice
**************************************************************************
****
Dumping non-idle PoFx devices
**************************************************************************
****
!fxdevice 0xffffe0012bbd08d0
    DevNode: 0xffffe0012b3f87b0
    UniqueId: "\_SB.PCI0"
    InstancePath: "ACPI\PNP0A08\2&daba3ff&1"
    Device Power State: PowerDeviceD0
    Component Count: 1
        Component 0: F0/F1 - ACTIVE (RefCount = 28)

!fxdevice 0xffffe0012c587940
    DevNode: 0xffffe0012b3f9d30
    UniqueId: "\_SB.PCI0.PEGL"
    InstancePath:
"PCI\VEN_8086&DEV_D138&SUBSYS_304A103C&REV_11\3&33fd14ca&0&18"
    Device Power State: PowerDeviceD0
    Component Count: 1
        Component 0: F0/F1 - ACTIVE (RefCount = 1)

...
```

```
*******************************************************************************
****
Dumping idle D0 PoFx devices
*******************************************************************************
****
!fxdevice 0xffffe0012c5838c0
    DevNode: 0xffffe0012bbdfd30
    UniqueId: "\_SB.PCI0.PCX1"
    InstancePath:
"PCI\VEN_8086&DEV_3B42&SUBSYS_304A103C&REV_05\3&33fd14ca&0&E0"
    Device Power State: PowerDeviceD0
    Component Count: 1
        Component 0: F1/F1 - IDLE   (RefCount = 0)

!fxdevice 0xffffe0012c581ac0
    DevNode: 0xffffe0012bbdfa50
    UniqueId: "\_SB.PCI0.PCX5"
    InstancePath:
"PCI\VEN_8086&DEV_3B4A&SUBSYS_304A103C&REV_05\3&33fd14ca&0&E4"
    Device Power State: PowerDeviceD0
    Component Count: 1
        Component 0: F1/F1 - IDLE   (RefCount = 0)

...
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gbl

Статья • 05.04.2024

Расширение **!gbl** отображает сведения о заголовках из таблицы ОПИСАНИЯ корневой системы ACPI BIOS (RSDT) целевого компьютера.

dbgcmd

```
!gbl [-v]
```

## Параметры

**-v**
Verbose. Отображает подробные сведения о таблице.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о таблицах ACPI и ACPI см. в статье "Другие расширения отладки ACPI" и веб-сайт спецификации ACPI. Также ознакомьтесь с документацией по пакету SDK для Microsoft Windows, документацией по комплекту драйверов Windows (WDK) и *внутренним службам* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gentable

Статья • 08.07.2024

Расширение **!gentable** отображает RTL_GENERIC_TABLE.

Синтаксис

dbgcmd

```
!gentable Address[Flag]
```

## Параметры

*Адрес*
Указывает адрес RTL_GENERIC_TABLE.

*Флаг*
Указывает источник таблицы. Если *флаг* равен 1, используется таблица AVL. Если *флаг* равен 0 или опущен, используется таблица, не относящийся к AVL.

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !hidppd

Статья • 05.04.2024

Команда **расширения !hidppd** отображает содержимое структуры HIDP_PREPARSED_DATA.

dbgcmd

```
!hidppd Address
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес структуры HIDP_PREPARSED_DATA.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о устройствах ввода данных (HID) см. в документации по комплекту драйверов Windows (WDK).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ib, !id, !iw

Статья • 26.10.2023

Команды расширения **!ib**, **!id** и **!iw** устарели. Вместо этого используйте команды **ib, id, iw (входные данные из порта).**

# !icpleak

Статья • 05.04.2024

Расширение **!icpleak** проверяет все объекты завершения ввода-вывода в системе для объекта с наибольшим количеством записей в очереди.

```
dbgcmd
```

```
!icpleak [HandleFlag]
```

## Параметры

*HandleFlag*
Если этот флаг задан, отображение также включает все процессы, имеющие дескриптор объекта с наибольшим количеством записей в очереди.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о портах *завершения ввода-вывода см. в разделе* "Внутренние компоненты Microsoft Windows" Марк Руссинович и Дэвид Соломон.

## Замечания

Это расширение полезно при утечке в пуле завершения ввода-вывода. Утечки пула завершения ввода-вывода могут возникать, когда процесс выделяет пакеты завершения ввода-вывода путем вызова **PostQueuedCompletionStatus, но не вызывает GetQueuedCompletionStatus**, чтобы освободить их, или когда процесс очереди записей завершения в порт, но нет потока, извлекающего записи. Чтобы обнаружить утечку**, выполните расширение !poolused** и проверка значение тега пула ICP. Если использование пула с тегом ICP является значительным, возможно, произошла утечка.

Это расширение работает только в том случае, если система поддерживает списки типов. *Если параметр HandleFlag* задан и система имеет множество процессов, это расширение займет много времени для выполнения.

Вы можете остановиться в любой точке, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Idt

Статья • 05.04.2024

Расширение **!idt** отображает подпрограммы службы прерываний (ISR) для указанной таблицы отправки прерываний (IDT).

dbgcmd

```
!idt IDT
!idt [-a]
!idt -?
```

## Параметры

*IDT*
Указывает идентификатор идентификатора для отображения.

**-a**
Если *idT* не указан, отладчик отображает идентификаторы всех процессоров на целевом компьютере в сокращенном формате. Если **задано значение -a** , также отображаются isR для каждого идентификатора.

**-?**
Отображает справку по этому расширению в окне команды отладчика.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x64 или x86.

## Дополнительная информация

Сведения о поставщиках удостоверений и идентификаторах см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
0: kd> !idt

Dumping IDT:

37:806ba78c hal!PicSpuriousService37
3d:806bbc90 hal!HalpApcInterrupt
41:806bbb04 hal!HalpDispatchInterrupt
50:806ba864 hal!HalpApicRebootService
63:8641376c VIDEOPRT!pVideoPortInterrupt (KINTERRUPT 86413730)
73:862aa044 portcls!CInterruptSyncServiceRoutine (KINTERRUPT 862aa008)
82:86594314 atapi!IdePortInterrupt (KINTERRUPT 865942d8)
83:86591bec SCSIPORT!ScsiPortInterrupt (KINTERRUPT 86591bb0)
92:862b53dc serial!SerialCIsrSw (KINTERRUPT 862b53a0)
93:86435844 i8042prt!I8042KeyboardInterruptService (KINTERRUPT 86435808)
a3:863b366c i8042prt!I8042MouseInterruptService (KINTERRUPT 863b3630)
a4:8636bbec USBPORT!USBPORT_InterruptService (KINTERRUPT 8636bbb0)
b1:86585bec ACPI!ACPIInterruptServiceRoutine (KINTERRUPT 86585bb0)
b2:863c0524 serial!SerialCIsrSw (KINTERRUPT 863c04e8)
b4:86391a54 NDIS!ndisMIsr (KINTERRUPT 86391a18)
         USBPORT!USBPORT_InterruptService (KINTERRUPT 863ae890)
c1:806ba9d0 hal!HalpBroadcastCallService
d1:806b9dd4 hal!HalpClockInterrupt
e1:806baf30 hal!HalpIpiHandler
e3:806baca8 hal!HalpLocalApicErrorService
fd:806bb460 hal!HalpProfileInterrupt
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ih

Статья • 05.04.2024

Расширение **!ih** отображает запись журнала прерываний для указанного процессора.

dbgcmd

```
!ih Processor
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Процессор*
Указывает процессор. Если *обработчик* опущен, используется текущий процессор.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе Itanium.

## Замечания

Это расширение отображает запись журнала прерываний без ссылки на символы счетчика программы. Чтобы отобразить запись журнала прерываний с помощью символов счетчика программ, используйте **расширение !ego** расширение. Чтобы включить запись журнала прерываний, добавьте **/configflag=32** в параметры записи загрузки.

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
kd> !ih
Total # of interruptions = 2093185
Vector              IIP                  IPSR          ExtraField
 VHPT FAULT  e0000000830d3190      1010092a6018  IFA=      6fc00a0200c
```

```
        VHPT FAULT  e0000000830d33d0      1010092a6018  IFA=
1ffffe00001de2d0
 VHPT FAULT  e0000000830d33d0      1010092a6018  IFA= 1ffffe01befff338
        VHPT FAULT  e0000000830d3190      1010092a6018  IFA=
6fc00a0200c
        VHPT FAULT  e0000000830d33d0      1010092a6018  IFA=
1ffffe00001d9188
        VHPT FAULT  e0000000830d3880      1010092a6018  IFA=
1ffffe01befff250
        VHPT FAULT  e0000000830d3fb0      1010092a6018  IFA=
e0000165f82dc1c0
 VHPT FAULT  e000000083063390      1010092a6018  IFA= e0000000fffe0018
     THREAD SWITCH  e000000085896040  e00000008588c040  OSP=
e0000165f82dbd40
        VHPT FAULT  e00000008329b130      1210092a6018  IFA=
e0000165f7edaf30
        VHPT FAULT  e0000165f7eda640      1210092a6018  IFA=
e0000165fff968a9
    PROCESS SWITCH  e0000000818bbe10  e000000085896040  OSP=
e0000165f8281de0
        VHPT FAULT  e00000008307cfc0      1010092a2018  IFA=
e00000008189fe50
EXTERNAL INTERRUPT  e0000000830623b0      1010092a6018  IVR=
d0
        VHPT FAULT  e00000008314e310      1010092a2018  IFA=
e0000165f88203f0
        VHPT FAULT  e000000083580760      1010092a2018  IFA=
e0000000f00ff3fd
    PROCESS SWITCH  e00000008558c990  e0000000818bbe10  OSP=
e00000008189fe20
        VHPT FAULT  e00000008307cfc0      1010092a2018  IFA=
e0000165f02433f0
        VHPT FAULT          77cfbda0      1013082a6018  IFA=
77cfbda0
 VHPT FAULT          77cfbdb0      1213082a6018  IFA=      6fbfee0ff98
   DATA ACCESS BIT          77b8e150      1213082a6018  IFA=
77c16ab8
        VHPT FAULT          77ed5d60      1013082a6018  IFA=
6fbfffa6048
   DATA ACCESS BIT          77ed5d60      1213082a6018  IFA=
77c229c0
 DATA ACCESS BIT          77b8e1b0      1013082a6018  IFA=         77c1c320
      USER SYSCALL          77cafa40        10082a6018  Num=
42
 VHPT FAULT  e00000008344dc20      1010092a6018  IFA= e000010600703784
...
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !Ihs

Статья • 05.04.2024

Расширение **!его** расширение отображает запись журнала прерываний для указанного процессора с помощью символов счетчика программ.

dbgcmd

```
!ihs Processor
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Процессор*
Указывает процессор. Если *обработчик* опущен, используется текущий процессор.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером Itanium.

## Замечания

Чтобы отобразить запись журнала прерываний без использования символов счетчика программ, используйте **расширение !ih** . Чтобы включить запись журнала прерываний, добавьте **/configflag=32** в параметры записи загрузки.

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
kd> !ihs
Total # of interruptions = 2093185
Vector              IIP                  IPSR          ExtraField
IIP Symbol
        VHPT FAULT  e0000000830d3190      1010092a6018  IFA=
6fc00a0200c nt!MiAgeAndEstimateAvailableInWorkingSet+0x70
        VHPT FAULT  e0000000830d33d0      1010092a6018  IFA=
```

```
1ffffe00001de2d0 nt!MiAgeAndEstimateAvailableInWorkingSet+0x2b0
        VHPT FAULT  e0000000830d33d0      1010092a6018  IFA=
1ffffe01befff338 nt!MiAgeAndEstimateAvailableInWorkingSet+0x2b0
        VHPT FAULT  e0000000830d3190      1010092a6018  IFA=
6fc00a0200c nt!MiAgeAndEstimateAvailableInWorkingSet+0x70
        VHPT FAULT  e0000000830d33d0      1010092a6018  IFA=
1ffffe00001d9188 nt!MiAgeAndEstimateAvailableInWorkingSet+0x2b0
        VHPT FAULT  e0000000830d3880      1010092a6018  IFA=
1ffffe01befff250 nt!MiAgeAndEstimateAvailableInWorkingSet+0x760
        VHPT FAULT  e0000000830d3fb0      1010092a6018  IFA=
e0000165f82dc1c0 nt!MiCheckAndSetSystemTrimCriteria+0x190
        VHPT FAULT  e000000083063390      1010092a6018  IFA=
e0000000fffe0018 nt!KeQuerySystemTime+0x30
     THREAD SWITCH  e000000085896040  e00000008588c040  OSP=
e0000165f82dbd40
        VHPT FAULT  e00000008329b130      1210092a6018  IFA=
e0000165f7edaf30 nt!IopProcessWorkItem+0x30
 VHPT FAULT  e0000165f7eda640      1210092a6018  IFA= e0000165fff968a9
netbios!RunTimerForLana+0x60
    PROCESS SWITCH  e0000000818bbe10  e000000085896040  OSP=
e0000165f8281de0
        VHPT FAULT  e00000008307cfc0      1010092a2018  IFA=
e00000008189fe50 nt!SwapFromIdle+0x1e0
EXTERNAL INTERRUPT  e0000000830623b0      1010092a6018  IVR=
d0 nt!Kil_TopOfIdleLoop
        VHPT FAULT  e00000008314e310      1010092a2018  IFA=
e0000165f88203f0 nt!KdReceivePacket+0x10
        VHPT FAULT  e000000083580760      1010092a2018  IFA=
e0000000f00ff3fd hal!READ_PORT_UCHAR+0x80
    PROCESS SWITCH  e00000008558c990  e0000000818bbe10  OSP=
e00000008189fe20
        VHPT FAULT  e00000008307cfc0      1010092a2018  IFA=
e0000165f02433f0 nt!SwapFromIdle+0x1e0
        VHPT FAULT          77cfbda0      1013082a6018  IFA=
77cfbda0 0x0000000077cfbda0
        VHPT FAULT          77cfbdb0      1213082a6018  IFA=
6fbfee0ff98 0x0000000077cfbdb0
 DATA ACCESS BIT          77b8e150      1213082a6018  IFA=         77c16ab8
0x0000000077b8e150
 VHPT FAULT          77ed5d60      1013082a6018  IFA=      6fbfffa6048
0x0000000077ed5d60
   DATA ACCESS BIT          77ed5d60      1213082a6018  IFA=
77c229c0 0x0000000077ed5d60
 DATA ACCESS BIT          77b8e1b0      1013082a6018  IFA=         77c1c320
0x0000000077b8e1b0
      USER SYSCALL          77cafa40        10082a6018  Num=
42 0x0000000077cafa40
 VHPT FAULT  e00000008344dc20      1010092a6018  IFA= e000010600703784
nt!ExpLookupHandleTableEntry+0x20
...
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ioresdes

Статья • 05.04.2024

Расширение **!ioresdes** отображает структуру IO_RESOURCE_DESCRIPTOR по указанному адресу.

dbgcmd

```
!ioresdes Address
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес структуры IO_RESOURCE_DESCRIPTOR.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Сведения о структуре IO_RESOURCE_DESCRIPTOR см. в документации по комплекту драйверов Windows (WDK).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ioctldecode

Статья • 05.04.2024

Расширение **!ioctldecode** отображает *mun* устройства, обязательный доступ, *код* функции и *mun* передачи, указанный указанным кодом IOCTL. Дополнительные сведения о кодах управления IOCTL см. в разделе "Определение кодов элементов управления ввода-вывода".

dbgcmd

```
!ioctldecode IoctlCode
```

## Параметры

*IoctlCode*

Задает шестнадцатеричный код IOCTL. Команда **!irp** отображает код IOCTL в выходных данных.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Чтобы просмотреть информацию об IOCTL, сначала мы найдем IRP интереса. Вы можете использовать **команду !irpfind** для поиска интересующего вас идентификатора.

**Используйте команду !irp** для отображения сведений об irp.

dbgcmd

```
0: kd> !irp ffffd581a6c6cd30
Irp is active with 6 stacks 6 is current (= 0xffffd581a6c6cf68)
No Mdl: No System Buffer: Thread 00000000:  Irp stack trace.
     cmd  flg cl Device   File     Completion-Context
[N/A(0), N/A(0)]
            0  0 00000000 00000000 00000000-00000000

                                                Args: 00000000 00000000
00000000 00000000
[N/A(0), N/A(0)]
```

```
            0  0 00000000 00000000 00000000-00000000

                                                Args: 00000000 00000000
00000000 00000000
[N/A(0), N/A(0)]
            0  0 00000000 00000000 00000000-00000000

                                                Args: 00000000 00000000
00000000 00000000
[N/A(0), N/A(0)]
            0  0 00000000 00000000 00000000-00000000

                                                Args: 00000000 00000000
00000000 00000000
[N/A(0), N/A(0)]
            0  0 00000000 00000000 00000000-00000000

                                                Args: 00000000 00000000
00000000 00000000
>[IRP_MJ_INTERNAL_DEVICE_CONTROL(f), N/A(0)]
            0 e1 ffffd581a5fbd050 00000000 fffff806d2412cf0-ffffd581a5cce050
Success Error Cancel pending
                      \Driver\usbehci        (IopUnloadSafeCompletion)
                                                Args: ffffd581a6c61a50
00000000 0x220003 00000000
```

Третий аргумент, отображаемый в данном случае *0x220003*, является кодом IOCTL. Используйте код IOCTL для отображения сведений о IOCTL, в этом случае **IOCTL_INTERNAL_USB_SUBMIT_URB**.

dbgcmd

```
0: kd> !ioctldecode 0x220003

IOCTL_INTERNAL_USB_SUBMIT_URB

Device Type    : 0x22 (FILE_DEVICE_WINLOAD) (FILE_DEVICE_USER_MODE_BUS)
(FILE_DEVICE_USB) (FILE_DEVICE_UNKNOWN)
Method         : 0x3 METHOD_NEITHER
Access         : FILE_ANY_ACCESS
Function       : 0x0
```

Если вы предоставляете код IOCTL, который недоступен, вы увидите этот тип выходных данных.

dbgcmd

```
0: kd> !ioctldecode 0x1280ce

Unknown IOCTL  : 0x1280ce
```

```
Device Type    : 0x12 (FILE_DEVICE_NETWORK)
Method         : 0x2 METHOD_OUT_DIRECT
Access         : FILE_WRITE_ACCESS
Function       : 0x33
```

Хотя IOCTL не определен, отображаются сведения о полях IOCTL.

Обратите внимание, что только подмножество общедоступных определяемых IOCTLs можно определить с помощью **команды !ioctldecode** .

Дополнительные сведения о ioCTLs см. в разделе "Введение в коды управления ввода-вывода".

Дополнительные сведения об IRPs и IOCTLs см. в разделе "*Внутренние* " Windows Марком Э. Руссиновичем, Дэвидом А. Соломоном и Алексом Ионеску.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ioreslist

Статья • 05.04.2024

Расширение **!ioreslist** отображает структуру IO_RESOURCE_REQUIREMENTS_LIST.

```
dbgcmd
```

```
!ioreslist Address
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес структуры IO_RESOURCE_REQUIREMENTS_LIST.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Сведения о структуре IO_RESOURCE_REQUIREMENTS_LIST см. в документации по комплекту драйверов Windows (WDK).

## Замечания

Ниже приведен пример выходных данных из этого расширения:

```
dbgcmd
```

```
kd> !ioreslist 0xe122b768

IoResList at 0xe122b768 : Interface 0x5  Bus 0  Slot 0xe
  Alternative 0 (Version 1.1)
    Preferred Descriptor 0 - Port (0x1) Device Exclusive (0x1)
      Flags (0x01) - PORT_IO
      0x000100 byte range with alignment 0x000100
      1000 - 0x10ff
    Alternative Descriptor 1 - Port (0x1) Device Exclusive (0x1)
      Flags (0x01) - PORT_IO
      0x000100 byte range with alignment 0x000100
      0 - 0xffffffff
```

```
    Descriptor 2 - DevicePrivate (0x81) Device Exclusive (0x1)
      Flags (0000) -
      Data:              : 0x1 0x0 0x0
    Preferred Descriptor 3 - Memory (0x3) Device Exclusive (0x1)
      Flags (0000) - READ_WRITE
      0x001000 byte range with alignment 0x001000
      40080000 - 0x40080fff
    Alternative Descriptor 4 - Memory (0x3) Device Exclusive (0x1)
      Flags (0000) - READ_WRITE
      0x001000 byte range with alignment 0x001000
      0 - 0xffffffff
    Descriptor 5 - DevicePrivate (0x81) Device Exclusive (0x1)
      Flags (0000) -
      Data:              : 0x1 0x1 0x0
    Descriptor 6 - Interrupt (0x2) Shared (0x3)
      Flags (0000) - LEVEL_SENSITIVE
      0xb - 0xb
```

IO_RESOURCE_REQUIREMENTS_LIST содержит сведения о следующих параметрах:

- Типы ресурсов

  Существует четыре типа ресурсов: операций ввода-вывода, памяти, IRQ, DMA.

- Дескрипторы

  Каждый предпочтительный параметр имеет дескриптор "Предпочтительный" и ряд дескрипторов "Альтернативный".

Этот список ресурсов содержит следующие запросы:

- Диапазоны операций ввода-вывода

  Предпочитает диапазон 0x1000 0x10FF включительно, но может использовать любой диапазон 0x100 от 0 до 0xFFFFFFFF, если он 0x100 выровнен. (Например, 0x1100 0x11FF приемлемо.)

- Память

  Предпочитает диапазон 0x40080000 0x40080FFF, но может использовать любой диапазон, который имеет размер 0x1000, 0x1000 выравнивание и расположен от 0 до 0xFFFFFFFF.

- IRQ

  Необходимо использовать 0xB IRQ.

Прерывания и каналы DMA представлены в виде диапазонов с одинаковым началом и концом.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !iovirp

Статья • 05.04.2024

Расширение **!iovirp** отображает подробные сведения для указанного IRP проверяющего ввода-вывода.

dbgcmd

```
!iovirp [IRP]
```

## Параметры

*IRP*
Указывает адрес IRP, отслеживаемый проверяющей машиной драйвера. Если *значение IRP равно 0 или опущено, отображается сводная информация для каждого выдающегося IRP* .

## DLL-библиотеки

Kdexts.dll

## Замечания

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
kd> !iovirp 947cef68
IovPacket       84509af0
TrackedIrp      947cef68
HeaderLock      84509d61
LockIrql        0
ReferenceCount  1
PointerCount    1
HeaderFlags     00000000
ChainHead       84509af0
Flags           00200009
DepartureIrql   0
ArrivalIrql     0
StackCount      1
QuotaCharge     00000000
QuotaProcess    0
RealIrpCompletionRoutine        0
```

```
RealIrpControl                  0
RealIrpContext                  0
TopStackLocation        2
PriorityBoost           0
LastLocation            0
RefTrackingCount        0
SystemDestVA            0
VerifierSettings        84509d08
pIovSessionData         84509380
Allocation Stack:
  nt!IovAllocateIrp+1a  (817df356)
 nt!IopXxxControlFile+40c  (8162de20)
  nt!NtDeviceIoControlFile+2a  (81633090)
 nt!KiFastCallEntry+164  (81513c64)
  nt!EtwpFlushBuffer+10f  (817606d7)
  nt!EtwpFlushBuffersWithMarker+bd  (817608cb)
  nt!EtwpFlushActiveBuffers+2b4  (81760bc2)
  nt!EtwpLogger+213  (8176036f)
```

Вы можете остановить выполнение в любой момент, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ipi

Статья • 05.04.2024

Расширение **!ipi отображает состояние прерывания межпроцессора (IPI** ) для указанного процессора.

dbgcmd

```
!ipi [Processor]
```

## Параметры

*Процессор*
Указывает процессор. Если *обработчик* опущен, отображается состояние IPI для каждого процессора.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x86.

## Дополнительная информация

Дополнительные сведения об IP-адресах *см. в разделе* "Внутренние компоненты Microsoft Windows" марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
0: kd> !ipi
IPI State for Processor 0
  Worker Routine:  nt!KiFlushTargetMultipleTb [Stale]
  Parameter[0]:    0
  Parameter[1]:    3
  Parameter[2]:    F7C98770
  Ipi Trap Frame:  F7CCCCDC [.trap F7CCCCDC]
```

```
  Signal Done:     0
  IPI Frozen:      24 [FreezeActive] [Owner]
  Request Summary: 0
  Target Set:      0
  Packet Barrier:  0

IPI State for Processor 1
  Worker Routine:  nt!KiFlushTargetMultipleTb [Stale]
  Parameter[0]:    1
  Parameter[1]:    3
  Parameter[2]:    F7CDCD28
  Ipi Trap Frame:  F7C8CCC4 [.trap F7C8CCC4]
  Signal Done:     0
  IPI Frozen:      2 [Frozen]
  Request Summary: 0
  Target Set:      0
  Packet Barrier:  0
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Irp

Статья • 05.04.2024

Расширение !irp отображает сведения о пакете **запросов ввода-вывода (IRP** ).

dbgcmd

```
!irp Address [Detail]
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес IRP.

*Detail*
Если этот параметр включен с любым значением, например 1, выходные данные включают состояние IRP, адрес списка дескрипторов памяти (MDL), собственный поток и сведения о стеке для всех стеков ввода-вывода и сведения о каждом расположении стека для IRP, включая шестнадцатеричные версии основного кода функции и дополнительный код функции. Если этот параметр опущен, выходные данные содержат только сводку сведений.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о самонастраивающийся отладке и отладке штормов прерываний для приложений этой команды расширения. Дополнительные сведения об IRP см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном. Дополнительные сведения о основных и дополнительных кодах функций см. в документации по комплекту драйверов Windows (WDK).

В этом разделе описывается структура IRP, **IRP**.

Подробные сведения о декодировании структуры IRP, включая возвращенные Args, см. в следующих ресурсах.

- Windows Internals марк Э. Руссинович, Дэвид А. Соломон и Алекс Ионеску
- Разработка драйверов с помощью Windows Driver Foundation Гай Смит и Пенни Орвик

# Замечания

Выходные данные также указывают, в каких условиях подпрограмма завершения для каждого расположения стека будет вызываться после завершения IRP и обработки расположения стека. Существует три возможных причины:

**Успешно**
Указывает, что подпрограмма завершения будет вызываться при завершении IRP с кодом успешного выполнения.

**Ошибка**
Указывает, что подпрограмма завершения будет вызываться при завершении IRP с кодом ошибки.

**Cancel**
Указывает, что подпрограмма завершения будет вызвана, если была предпринята попытка отменить IRP.

Любое сочетание этих трех может появиться, и если выполнены какие-либо из указанных условий, будет вызвана подпрограмма завершения. Соответствующие значения отображаются в конце первой строки сведений о каждом расположении стека сразу после **записи "Завершение-контекст** ".

Ниже приведен пример выходных данных из этого расширения для Windows 10:

dbgcmd

```
0: kd> !irp ac598dc8
Irp is active with 2 stacks 1 is current (= 0xac598e38)
 No Mdl: No System Buffer: Thread 8d1c7bc0:  Irp stack trace.
     cmd  flg cl Device   File     Completion-Context
>[IRP_MJ_FILE_SYSTEM_CONTROL(d), N/A(0)]
            1 e1 8a6434d8 ac598d40 853220cb-a89682d8 Success Error Cancel
pending
           \FileSystem\Npfs fltmgr!FltpPassThroughCompletion
            Args: 00000000 00000000 00110008 00000000
 [IRP_MJ_FILE_SYSTEM_CONTROL(d), N/A(0)]
            1  0 8a799710 ac598d40 00000000-00000000
           \FileSystem\FltMgr
            Args: 00000000 00000000 0x00110008 00000000
```

Начиная с Windows 10 отображается основной и дополнительный текст кода IRP, например "IRP_MJ_FILE_SYSTEM_CONTROL" Значение кода также отображается в шестнадцатеричном виде в этом примере "(d)".

Третий аргумент, отображаемый в выходных данных, — это код IOCTL. **Используйте команду !ioctldecode** для отображения сведений о IOCTL.

Ниже приведен пример выходных данных из этого расширения из Windows Vista.

dbgcmd

```
0: kd> !irp 0x831f4a00
Irp is active with 8 stacks 5 is current (= 0x831f4b00)
 Mdl = 82b020d8 Thread 8c622118:  Irp stack trace.
     cmd  flg cl Device   File     Completion-Context
 [  0, 0]   0  0 00000000 00000000 00000000-00000000

                        Args: 00000000 00000000 00000000 00000000
 [  0, 0]   0  0 00000000 00000000 00000000-00000000

                        Args: 00000000 00000000 00000000 00000000
 [  0, 0]   0  0 00000000 00000000 00000000-00000000

                        Args: 00000000 00000000 00000000 00000000
 [  0, 0]   0  0 00000000 00000000 00000000-00000000

                        Args: 00000000 00000000 00000000 00000000
>[  3,34]  40 e1 828517a8 00000000 842511e0-00000000 Success Error Cancel
pending
               \Driver\disk     partmgr!PmReadWriteCompletion
 Args: 00007000 00000000 fe084e00 00000004
 [  3, 0]  40 e0 82851450 00000000 842414d4-82956350 Success Error Cancel
 \Driver\PartMgr  volmgr!VmpReadWriteCompletionRoutine
                        Args: 129131bb 000000de fe084e00 00000004
 [  3, 0]   0 e0 82956298 00000000 847eeed0-829e2ba8 Success Error Cancel
 \Driver\volmgr   Ntfs!NtfsMasterIrpSyncCompletionRoutine
                        Args: 00007000 00000000 1bdae400 00000000
 [  3, 0]   0  0 82ac2020 8e879410 00000000-00000000
               \FileSystem\Ntfs
                        Args: 00007000 00000000 00018400 00000000
```

Обратите внимание, что подпрограмма завершения рядом с именем драйвера задана в этом расположении стека, и она была задана драйвером в строке ниже. В предыдущем примере **ntfs!NtfsMasterIrpSyncCompletionRoutine** был задан параметром **\FileSystem\Ntfs**. Запись "Завершение-контекст" над **ntfs!NtfsMasterIrpSyncCompletionRoutine**, **847eeed0-829e2ba8**, указывает адрес подпрограммы завершения, а также контекст, который будет передан **в Ntfs!NtfsMasterIrpSyncCompletionRoutine**. Из этого мы видим, что адрес

**Ntfs!NtfsMasterIrpSyncCompletionRoutine** равен **847eeed0**, и контекст, который будет передан в эту подпрограмму при вызове **829e2ba8**.

**Коды основных функций IRP**

Ниже приведены сведения, которые помогут вам интерпретировать выходные данные из этой команды расширения.

Основные коды функций IRP приведены следующим образом:

Развернуть таблицу

| Основной код функции | Шестнадцатеричный код |
|---|---|
| IRP_MJ_CREATE | 0x00 |
| IRP_MJ_CREATE_NAMED_PIPE | 0x01 |
| IRP_MJ_CLOSE | 0x02 |
| IRP_MJ_READ | 0x03 |
| IRP_MJ_WRITE | 0x04 |
| IRP_MJ_QUERY_INFORMATION | 0x05 |
| IRP_MJ_SET_INFORMATION | 0x06 |
| IRP_MJ_QUERY_EA | 0x07 |
| IRP_MJ_SET_EA | 0x08 |
| IRP_MJ_FLUSH_BUFFERS | 0x09 |
| IRP_MJ_QUERY_VOLUME_INFORMATION | 0x0A |
| IRP_MJ_SET_VOLUME_INFORMATION | 0x0B |
| IRP_MJ_DIRECTORY_CONTROL | 0x0C |
| IRP_MJ_FILE_SYSTEM_CONTROL | 0x0D |
| IRP_MJ_DEVICE_CONTROL | 0x0E |
| IRP_MJ_INTERNAL_DEVICE_CONTROL<br>IRP_MJ_SCSI | 0x0F |
| IRP_MJ_SHUTDOWN | 0x10 |
| IRP_MJ_LOCK_CONTROL | 0x11 |
| IRP_MJ_CLEANUP | 0x12 |

| Основной код функции | Шестнадцатеричный код |
|---|---|
| IRP_MJ_CREATE_MAILSLOT | 0x13 |
| IRP_MJ_QUERY_SECURITY | 0x14 |
| IRP_MJ_SET_SECURITY | 0x15 |
| IRP_MJ_POWER | 0x16 |
| IRP_MJ_SYSTEM_CONTROL | 0x17 |
| IRP_MJ_DEVICE_CHANGE | 0x18 |
| IRP_MJ_QUERY_QUOTA | 0x19 |
| IRP_MJ_SET_QUOTA | 0x1A |
| IRP_MJ_PNP IRP_MJ_MAXIMUM_FUNCTION | 0x1B |

Коды дополнительных функций самонастраивающийся приведены следующим образом:

Развернуть таблицу

| Дополнительный код функции | Шестнадцатеричный код |
|---|---|
| IRP_MN_START_DEVICE | 0x00 |
| IRP_MN_QUERY_REMOVE_DEVICE | 0x01 |
| IRP_MN_REMOVE_DEVICE | 0x02 |
| IRP_MN_CANCEL_REMOVE_DEVICE | 0x03 |
| IRP_MN_STOP_DEVICE | 0x04 |
| IRP_MN_QUERY_STOP_DEVICE | 0x05 |
| IRP_MN_CANCEL_STOP_DEVICE | 0x06 |
| IRP_MN_QUERY_DEVICE_RELATIONS | 0x07 |
| IRP_MN_QUERY_INTERFACE | 0x08 |
| IRP_MN_QUERY_CAPABILITIES | 0x09 |
| IRP_MN_QUERY_RESOURCES | 0x0A |
| IRP_MN_QUERY_RESOURCE_REQUIREMENTS | 0x0B |
| IRP_MN_QUERY_DEVICE_TEXT | 0x0C |

| Дополнительный код функции | Шестнадцатеричный код |
|---|---|
| IRP_MN_FILTER_RESOURCE_REQUIREMENTS | 0x0D |
| IRP_MN_READ_CONFIG | 0x0F |
| IRP_MN_WRITE_CONFIG | 0x10 |
| IRP_MN_EJECT | 0x11 |
| IRP_MN_SET_LOCK | 0x12 |
| IRP_MN_QUERY_ID | 0x13 |
| IRP_MN_QUERY_PNP_DEVICE_STATE | 0x14 |
| IRP_MN_QUERY_BUS_INFORMATION | 0x15 |
| IRP_MN_DEVICE_USAGE_NOTIFICATION | 0x16 |
| IRP_MN_SURPRISE_REMOVAL | 0x17 |
| IRP_MN_QUERY_LEGACY_BUS_INFORMATION | 0x18 |

Ниже приведены коды дополнительных функций WMI:

Развернуть таблицу

| Дополнительный код функции | Шестнадцатеричный код |
|---|---|
| IRP_MN_QUERY_ALL_DATA | 0x00 |
| IRP_MN_QUERY_SINGLE_INSTANCE | 0x01 |
| IRP_MN_CHANGE_SINGLE_INSTANCE | 0x02 |
| IRP_MN_CHANGE_SINGLE_ITEM | 0x03 |
| IRP_MN_ENABLE_EVENTS | 0x04 |
| IRP_MN_DISABLE_EVENTS | 0x05 |
| IRP_MN_ENABLE_COLLECTION | 0x06 |
| IRP_MN_DISABLE_COLLECTION | 0x07 |
| IRP_MN_REGINFO | 0x08 |
| IRP_MN_EXECUTE_METHOD | 0x09 |

Ниже приведены коды дополнительных функций управления питанием:

Развернуть таблицу

| Дополнительный код функции | Шестнадцатеричный код |
|---|---|
| IRP_MN_WAIT_WAKE | 0x00 |
| IRP_MN_POWER_SEQUENCE | 0x01 |
| IRP_MN_SET_POWER | 0x02 |
| IRP_MN_QUERY_POWER | 0x03 |

Ниже приведены коды дополнительных функций SCSI:

Развернуть таблицу

| Дополнительный код функции | Шестнадцатеричный код |
|---|---|
| IRP_MN_SCSI_CLASS | 0x01 |

# См. также

IRP

!irpfind

!ioctldecode

# Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !irpfind

Статья • 05.04.2024

Расширение **!irpfind** отображает сведения обо всех пакетах запросов ввода-вывода (IRP), выделенных в целевой системе, или об этих irPs, соответствующих указанным критериям поиска.

Синтаксис

```
dbgcmd

!irpfind [-v][PoolType[RestartAddress[CriteriaData]]]
```

# Параметры

**-v**
Отображает подробные сведения.

*PoolType*
Указывает тип пула для поиска. Разрешены следующие значения:

0
Указывает непагаченный пул памяти. Это значение по умолчанию.

1
Указывает пул памяти с страницами.

2
Указывает специальный пул.

4
Указывает пул сеансов.

*ПерезапускAddress*
Указывает шестнадцатеричный адрес, с которого начинается поиск. Это полезно, если предыдущий поиск был завершен преждевременно. По умолчанию используется значение 0.

*Критерии*
Задает критерии для поиска. Будут отображаться только те irPs, которые удовлетворяют заданному совпадению.

Развернуть таблицу

| Критерии | Поиск совпадений (Match) |
|---|---|
| **arg** | Находит все irPs с расположением стека, где один из аргументов равен *данным*. |
| **device** | Находит все irPs с расположением стека, где **DeviceObject** равен *данным*. |
| **fileobject** | Находит все irPs, Irp.Tail.Overlay.OriginalFileObject равно *Data*. |
| **mdlprocess** | Находит все IRP, Irp.MdlAddress.Process равно *данным*. |
| **thread** | Находит все IRP, Irp.Tail.Overlay.Thread равно *данным*. |
| **userevent** | Находит все IRP, Irp.UserEvent которых равно *данным*. |

*Данные*
Указывает данные, которые должны быть сопоставлены в поиске.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Дополнительные сведения об IRP см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Этот пример находит IRP в непагованном пуле, который собирается задать событие пользователя FF9E4F48 после завершения:

dbgcmd

```
kd> !irpfind 0 0 userevent ff9e4f48
```

В следующем примере создается полный список всех IRP в непакованном пуле:

dbgcmd

```
kd> !irpfind
Searching NonPaged pool (8090c000 : 8131e000) for Tag: Irp
8097c008 Thread 8094d900 current stack belongs to  \Driver\symc810
8097dec8 Thread 8094dda0 current stack belongs to  \FileSystem\Ntfs
809861a8 Thread 8094dda0 current stack belongs to  \Driver\symc810
809864e8 Thread 80951ba0 current stack belongs to  \Driver\Mouclass
80986608 Thread 80951ba0 current stack belongs to  \Driver\Kbdclass
80986728 Thread 8094dda0 current stack belongs to  \Driver\symc810
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !irpzone

Статья • 26.10.2023

Команда **расширения !irpzone** устарела. Вместо этого используйте **!irpfind** .

# !Irql

Статья • 05.04.2024

Расширение **!irql отображает уровень запроса прерывания (IRQL** ) процессора на целевом компьютере перед разрывом отладчика.

dbgcmd

```
!irql [Processor]
```

## Параметры

*Процессор*
Задает процессор. Введите номер процессора. Если этот параметр опущен, отладчик отображает IRQL текущего процессора.

## DLL-библиотеки

**Расширение !irql** доступно только в Windows Server 2003 и более поздних версиях Windows.

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Рекомендации недоступны |
| **Windows XP** | Рекомендации недоступны |
| **Windows Server 2003 и более поздних версий** | Kdexts.dll |

## Дополнительная информация

Дополнительные сведения об IRQLs см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Когда целевой компьютер прерывается в отладчик, irQL изменяется, но IRQL, действующий непосредственно перед сохранением останова отладчика. Расширение **!irql отображает сохраненный IRQL** .

Аналогичным образом, когда возникает ошибка проверка и создается файл аварийного дампа, IRQL, сохраненный в файле аварийного дампа, является одним непосредственно перед ошибкой проверка, а не IRQL, в котором была выполнена подпрограмма KeBugCheckEx.

В обоих случаях текущий IRQL вызывается на DISPATCH_LEVEL, за исключением архитектур x86. Таким образом, если возникает несколько таких событий, отображаемый IRQL также будет DISPATCH_LEVEL, что делает его бесполезным для целей отладки.

Расширение **!pcr** отображает текущий IRQL во всех версиях Windows, но текущий IRQL обычно не полезен. IRQL, который существовал незадолго до проверка ошибки или подключения отладчика, более интересен, и это отображается только с **!irql**.

Если указан недопустимый номер процессора или произошла повреждение ядра, отладчик отображает сообщение "Не удается получить адрес PRCB".

Ниже приведен пример выходных данных этого расширения с компьютера x86 с двумя процессорами:

dbgcmd

```
kd> !irql 0
Debugger saved IRQL for processor 0x0 -- 28 (CLOCK2_LEVEL)

kd> !irql 1
Debugger saved IRQL for processor 0x1 -- 0 (LOW_LEVEL)
```

Если отладчик находится в подробном режиме, включено описание самого IRQL.

Значение номера IRQL часто зависит от процессора. Ниже приведен пример процессора x64. Обратите внимание, что номер IRQL совпадает с предыдущим примером, но значение IRQL отличается:

dbgcmd

```
kd> !irql
Debugger saved IRQL for processor 0x0 -- 12 (SYNCH_LEVEL) [Synchronization
level]
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !isainfo

Статья • 05.04.2024

Расширение **!isainfo** отображает сведения о карта или устройствах PNPISA, присутствующих в системе.

dbgcmd

```
!isainfo [Card]
```

## Параметры

*Card*

Указывает карточку PNPISA. Если *карточка* 0 или опущена, отображаются все устройства и карта на шине PNPISA (т. е. в шине ввода-вывода компьютера).

### DLL-библиотеки

Kdexts.dll

## Замечания

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
0: kd> !isainfo
ISA PnP FDO @ 0x867b9938, DevExt @ 0x867b99f0, Bus # 0
Flags (0x80000000)  DF_BUS

  ISA PnP PDO @ 0x867B9818, DevExt @ 0x86595388
  Flags (0x40000818)  DF_ENUMERATED, DF_ACTIVATED,
                      DF_REQ_TRIMMED, DF_READ_DATA_PORT
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !Isr

Статья • 05.04.2024

Расширение !isr отображает регистр состояния прерывания Itanium (ISR) по указанному адресу.

```
dbgcmd
```

```
!isr Expression [DisplayLevel]
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Выражение*
Указывает шестнадцатеричный адрес регистра ISR для отображения. Выражение **@isr** также можно использовать для этого параметра. В этом случае отображаются сведения о текущем регистре ISR процессора.

*DisplayLevel*
Может быть одним из следующих вариантов:

**0**
Отображает только значения каждого поля ISR. Это значение по умолчанию.

**1**
Отображает подробные сведения о полях ISR, которые не зарезервированы или игнорируются.

**2**
Отображает сведения обо всех полях ISR, включая те, которые игнорируются или зарезервированы.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером Itanium.

# Замечания

Ниже приведены несколько примеров выходных данных из этого расширения:

dbgcmd

```
kd> !isr @isr
isr:ed ei so ni ir rs sp na r w x vector code
  0  0  0  0  0  0  0  0 0 0 0      0   0

kd> !isr @isr 2

 cod : 0 : interruption Code
 vec : 0 : IA32 exception vector number
  rv : 0 : reserved0
   x : 0 : eXecute exception
   w : 0 : Write exception
   r : 0 : Read exception
  na : 0 : Non-Access exception
  sp : 0 : Speculative load exception
  rs : 0 : Register Stack
  ir : 0 : Invalid Register frame
  ni : 0 : Nested Interruption
  so : 0 : IA32 Supervisor Override
  ei : 0 : Exception IA64 Instruction
  ed : 0 : Exception Deferral
  rv : 0 : reserved1
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ivt

Статья • 05.04.2024

Расширение !ivt отображает таблицу вектора прерываний Itanium.

dbgcmd

```
!ivt [-v] [-a] [Vector]
!ivt -?
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Вектор*
Указывает запись векторной таблицы прерываний для текущего процессора. Если *вектор* опущен, отображается вся таблица вектора прерываний для текущего процессора на целевом компьютере. Векторы прерываний, которые не были назначены, не отображаются, если **параметр -a** не указан.

**-a**
Отображает все векторы прерываний, включая те, которые не назначены.

**-v**
Отображает подробные выходные данные.

**-?**
Отображает справку по этому расширению в окне команды отладчика.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером Itanium.

## Дополнительная информация

Дополнительные сведения о отображении таблиц отправки прерываний на целевом компьютере x64 или x86 см. в разделе **!idt**.

# Замечания

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
kd> !ivt

Dumping IA64 IVT:

00:e000000083005f60 nt!KiPassiveRelease
0f:e000000083576830 hal!HalpPCIISALine2Pin
10:e0000000830067f0 nt!KiApcInterrupt
20:e000000083006790 nt!KiDispatchInterrupt
30:e000000083576b30 hal!HalpCMCIHandler
31:e000000083576b20 hal!HalpCPEIHandler
41:e000000085039680 i8042prt!I8042KeyboardInterruptService (KINTERRUPT
e000000085039620)
51:e000000085039910 i8042prt!I8042MouseInterruptService (KINTERRUPT
e0000000850398b0)
61:e0000000854484f0 VIDEOPRT!pVideoPortInterrupt (KINTERRUPT
e000000085448490)
71:e0000000856c9450 NDIS!ndisMIsr (KINTERRUPT e0000000856c93f0)
81:e0000000857fd000 SCSIPORT!ScsiPortInterrupt (KINTERRUPT e0000000857fcfa0)
91:e0000000857ff510 atapi!IdePortInterrupt (KINTERRUPT e0000000857ff4b0)
a1:e0000000857d84b0 atapi!IdePortInterrupt (KINTERRUPT e0000000857d8450)
a2:e0000165fff2cab0 portcls!CInterruptSyncServiceRoutine (KINTERRUPT
e0000165fff2ca50)
b1:e0000000858c7460 ACPI!ACPIInterruptServiceRoutine (KINTERRUPT
e0000000858c7400)
b2:e0000000850382e0 USBPORT!USBPORT_InterruptService (KINTERRUPT
e000000085038280)
d0:e0000000835768d0 hal!HalpClockInterrupt
e0:e000000083576850 hal!HalpIpiInterruptHandler
f0:e0000000835769c0 hal!HalpProfileInterrupt
f1:e000000083576830 hal!HalpPCIISALine2Pin
fd:e000000083576b10 hal!HalpMcRzHandler
fe:e000000083576830 hal!HalpPCIISALine2Pin
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Работу

Статья • 05.04.2024

Расширение **!job** отображает объект задания.

```
dbgcmd
```

```
!job [Process [Flags]]
```

# Параметры

*Обработать*

Указывает шестнадцатеричный адрес процесса или потока, связанный с объектом задания которого требуется проверить. Если это опущено или равно нулю, отображается задание, связанное с текущим процессом.

*Flags*

Указывает, что должен содержать дисплей. Это может быть сумма любого из следующих битовых значений. Значение по умолчанию — 0x1:

Bit 0 (0x1)
Приводит к включению параметров задания и статистики.

Бит 1 (0x2)

Приводит к включению списка всех процессов в задание.

## DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения об объектах *заданий см. в разделе* "Внутренние компоненты Microsoft Windows", Марк Руссинович и Дэвид Соломон.

# Замечания

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
kd> !process 52c
Searching for Process with Cid == 52c
PROCESS 8276c550  SessionId: 0  Cid: 052c    Peb: 7ffdf000  ParentCid: 0060
    DirBase: 01289000  ObjectTable: 825f0368  TableSize:  24.
    Image: cmd.exe
    VadRoot 825609e8 Vads 30 Clone 0 Private 77. Modified 0. Locked 0.
    DeviceMap e1733f38
    Token                             e1681610
    ElapsedTime                       0:00:12.0949
    UserTime                          0:00:00.0359
    .....
    CommitCharge                      109
    Job                               8256e1f0

kd> !job 8256e1f0
Job at ffffffff8256e1f0
  TotalPageFaultCount      0
  TotalProcesses           1
  ActiveProcesses          1
  TotalTerminatedProcesses 0
  LimitFlags               0
  MinimumWorkingSetSize    0
  MaximumWorkingSetSize    0
  ActiveProcessLimit       0
  PriorityClass            0
  UIRestrictionsClass      0
  SecurityLimitFlags       0
  Token                    00000000
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !kb, !kv

Статья • 05.04.2024

Команды расширения !kb и !kv устарели. Используйте вместо этого команды kb (Display Stack Backtrace) **и kv (Display Stack Backtrace).**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !loadermemorylist

Статья • 05.04.2024

Расширение **!loadermemorylist** отображает список выделения памяти, который загрузчик Windows передает в Windows.

dbgcmd

```
!loadermemorylist ListHeadAddress
```

## Параметры

*ListHeadAddress*
Указывает адрес заголовка списка.

## DLL-библиотеки

Kdexts.dll

## Замечания

Это расширение предназначено для использования в начале процесса загрузки системы во время выполнения Ntldr. В нем отображается список выделения памяти, включающий начальный, конечный и тип каждого диапазона страниц.

Вы можете остановить выполнение в любой момент, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !lockedpages

Статья • 08.07.2024

Расширение **!lockedpages** отображает заблокированные драйвером страницы для указанного процесса.

Синтаксис

```
dbgcmd
```

```
!lockedpages [Process]
```

## Параметры

*Процесс*
Задает процесс. Если *процесс* опущен, используется текущий процесс.

## DLL-библиотеки

Kdexts.dll

## Замечания

Вы можете остановить выполнение в любой момент, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !locks (!kdext*.locks)

Статья • 05.04.2024

**Расширение !locks** в Kdextx86.dll и Kdexts.dll отображает сведения о блокировках ERESOURCE ядра.

Эта команда расширения не должна путаться с командой **расширения !ntsdexts.locks** .

dbgcmd

```
!locks [Options] [Address]
```

# Параметры

*Параметры* задает объем отображаемых сведений. Можно использовать любое сочетание следующих параметров:

**-v**
Отображает подробные сведения о каждой блокировке.

**-p**
Отображение всех доступных сведений о блокировках, включая статистику производительности.

**-d**
Отображение сведений обо всех блокировках. В противном случае отображаются только блокировки с конфликтом.)

*Адрес*
Указывает шестнадцатеричный адрес отображаемой блокировки ERESOURCE. Если *адрес* равен 0 или опущен, будут отображаться сведения обо всех блокировках ERESOURCE в системе.

## DLL-библиотеки

Kdexts.dll

# Замечания

Расширение **!locks** отображает все блокировки, удерживаемые на ресурсах по потокам. Блокировка может быть общей или эксклюзивной, что означает, что другие потоки не могут получить доступ к этому ресурсу. Эта информация полезна при возникновении взаимоблокировки в системе. Взаимоблокировка вызвана одним неисполнимым потоком, включающим монопольную блокировку ресурса, которому требуется выполнение потока.

Обычно можно определить взаимоблокировку в Microsoft Windows 2000, найдя один неисполняющий поток, содержащий монопольную блокировку ресурса, необходимого для выполнения потока. Большинство блокировок являются общими.

Ниже приведен пример базовых **выходных данных !locks** :

dbgcmd

```
kd> !locks
**** DUMP OF ALL RESOURCE OBJECTS ****
KD: Scanning for held locks......

Resource @ 0x80e97620    Shared 4 owning threads
     Threads: ff688da0-01<*> ff687da0-01<*> ff686da0-01<*> ff685da0-01<*>
KD: Scanning for held
locks..................................................

Resource @ 0x80e23f38    Shared 1 owning threads
     Threads: 80ed0023-01<*> *** Actual Thread 80ed0020
KD: Scanning for held locks.

Resource @ 0x80d8b0b0    Shared 1 owning threads
     Threads: 80ed0023-01<*> *** Actual Thread 80ed0020
2263 total locks, 3 locks currently held
```

Обратите внимание, что за адресом для каждого потока следует число потоков (например, "-01"). Если за потоком следует "<*>", этот поток является одним из владельцев блокировки. В некоторых случаях начальный адрес потока содержит смещение. В этом случае отображается фактический адрес потока.

Если вы хотите найти дополнительные сведения об одном из этих объектов ресурсов, используйте адрес, следующий за ресурсом @, в качестве аргумента для будущих команд. Для изучения второго ресурса, показанного в предыдущем примере, можно использовать dt ERESOURCE 80d8b0b0 **или !thread 80ed0020.** Или вы можете использовать **расширение !locks** еще раз с параметром **-v** :

dbgcmd

```
kd> !locks -v 80d8b0b0

Resource @ 0x80d8b0b0    Shared 1 owning threads
     Threads: 80ed0023-01<*> *** Actual Thread 80ed0020

     THREAD 80ed0020  Cid 4.2c  Teb: 00000000 Win32Thread: 00000000 WAIT:
(WrQueue) KernelMode Non-Alertable
         8055e100  Unknown
     Not impersonating
GetUlongFromAddress: unable to read from 00000000
     Owning Process 80ed5238
     WaitTime (ticks)          44294977
     Context Switch Count      147830
     UserTime                  0:00:00.0000
     KernelTime                0:00:02.0143
     Start Address nt!ExpWorkerThread (0x80506aa2)
     Stack Init fafa4000 Current fafa3d18 Base fafa4000 Limit fafa1000 Call
0
     Priority 13 BasePriority 13 PriorityDecrement 0
ChildEBP RetAddr
fafa3d30 804fe997 nt!KiSwapContext+0x25 (FPO: [EBP 0xfafa3d48] [0,0,4])
[D:\NT\base\ntos\ke\i386\ctxswap.asm @ 139]
fafa3d48 80506a17 nt!KiSwapThread+0x85 (FPO: [Non-Fpo]) (CONV: fastcall)
[d:\nt\base\ntos\ke\thredsup.c @ 1960]
fafa3d78 80506b36 nt!KeRemoveQueue+0x24c (FPO: [Non-Fpo]) (CONV: stdcall)
[d:\nt\base\ntos\ke\queueobj.c @ 542]
fafa3dac 805ad8bb nt!ExpWorkerThread+0xc6 (FPO: [Non-Fpo]) (CONV: stdcall)
[d:\nt\base\ntos\ex\worker.c @ 1130]
fafa3ddc 8050ec72 nt!PspSystemThreadStartup+0x2e (FPO: [Non-Fpo]) (CONV:
stdcall) [d:\nt\base\ntos\ps\create.c @ 2164]
00000000 00000000 nt!KiThreadStartup+0x16
[D:\NT\base\ntos\ke\i386\threadbg.asm @ 81]

1 total locks, 1 locks currently held
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !logonsession

Статья • 05.04.2024

Расширение **!logonsession** отображает сведения о указанном сеансе входа.

Синтаксис бесплатной сборки

```
dbgcmd
```

```
!logonsession LUID
```

Проверенный синтаксис сборки

```
dbgcmd
```

```
!logonsession LUID [InfoLevel]
```

## Параметры

*LUID*
Указывает локальный уникальный идентификатор (LUID) сеанса входа в систему для отображения. Если *LUID*равно 0, отображаются сведения обо всех сеансах входа.

Чтобы отобразить сведения о системном сеансе и всех системных маркерах в проверка сборке, введите **!logonsession 3e7 1**. Проверенные сборки были доступны в более ранних версиях Windows до Windows 10 версии 1803.

*InfoLevel*
(Только установленная сборка) Указывает, сколько сведений о маркере отображается. Параметр *InfoLevel* может принимать значения от 0 до 4, с 0, представляющей наименьшую информацию и 4, представляющую большую информацию. Проверенные сборки были доступны в более ранних версиях Windows до Windows 10 версии 1803.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о сеансах входа см. в документации по пакету SDK Для Microsoft Windows и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Ниже приведен пример выходных данных этого расширения для бесплатной сборки:

dbgcmd

```
kd> !logonsession 0

Dumping all logon sessions.

** Session   0 = 0x0
   LogonId     = {0x0 0x0}
   References  = 0
** Session   1 = 0x8ebb50
 LogonId     = {0xe9f1 0x0}
   References  = 21
** Session   2 = 0x6e31e0
   LogonId     = {0x94d1 0x0}
   References  = 1
** Session   3 = 0x8ecd60
   LogonId     = {0x6b31 0x0}
   References  = 0
** Session   4 = 0xe0000106
   LogonId     = {0x0 0x0}
   References  = 0
** Session   5 = 0x0
   LogonId     = {0x0 0x0}
   References  = 0
** Session   6 = 0x8e9720
   LogonId     = {0x3e4 0x0}
   References  = 6
** Session   7 = 0xe0000106
   LogonId     = {0x0 0x0}
   References  = 0
** Session   8 = 0xa2e160
   LogonId     = {0x3e5 0x0}
   References  = 3
** Session   9 = 0xe0000106
   LogonId     = {0x0 0x0}
   References  = 0
** Session  10 = 0x3ca0
   LogonId     = {0x3e6 0x0}
   References  = 2
** Session  11 = 0xe0000106
   LogonId     = {0x0 0x0}
   References  = 0
```

```
** Session  12 = 0x1cd0
   LogonId     = {0x3e7 0x0}
   References  = 33
** Session  13 = 0xe0000106
 LogonId     = {0x0 0x0}
 References  = 0
14 sessions in the system.
```

Вы можете остановить выполнение в любой момент, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !lookaside

Статья • 05.04.2024

Расширение **!lookaside** отображает сведения о списках поиска в сторону, сбрасывает счетчики списков в сторону или изменяет глубину списка взгляда в сторону.

dbgcmd

```
!lookaside [Address [Options [Depth]]]
!lookaside [-all]
!lookaside 0 [-all]
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес отображаемого или измененного списка в стороне.

Если *адрес* опущен (или 0) и **параметр -all** не указан, отображается набор известных стандартных системных списков поиска в сторону. Набор списков не является исчерпывающим; то есть он не включает все системные списки поиска в сторону. Кроме того, набор не включает настраиваемые списки поиска, созданные вызовами ExInitializePagedLookasideList **или ExInitializeNPagedLookasideList.**

Если *адрес* опущен (или 0) и **указан параметр -all** , отображаются все списки в стороне.

*Параметры* управляют тем, какая операция будет выполнена. Поддерживаются следующие возможные *параметры* . Значение по умолчанию равно нулю:

0
Отображает сведения о указанном списке или списках поиска.

1
Сбрасывает счетчики указанного списка поиска в сторону.

2
Изменяет глубину указанного списка взгляда в сторону. Этот параметр можно использовать только в том случае, если *адрес* не является ненулевой.

*Глубина*
Задает новую максимальную глубину указанного списка поиска в стороне. Этот параметр разрешен только в том случае, если *адрес* ненулевой и *параметры* равны 2.

# Дополнительная информация

Дополнительные сведения о списках поиска см. в разделе "Использование списков Lookaside" и *внутренних* элементов Microsoft Windows, от Марка Руссиновича и Дэвида Соломона.

# Замечания

Списки с поиском в стороне — это многопроцессорные механизмы для управления пулами записей фиксированного размера из страницы или непагрегированного памяти.

Списки поиска эффективны, так как подпрограммы не используют блокировки спина на большинстве платформ.

Обратите внимание, что если текущая глубина списка в стороне превышает максимальную глубину этого списка, то освобождение структуры, связанной с этим списком, приведет к освобождению его в памяти пула, а не в памяти списка.

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
!lookaside 0xffff88001294f80

Lookaside "" @ 0xffff88001294f80  Tag(hex): 0x7366744e "Ntfs"
    Type           =       0011  PagedPool RaiseIfAllocationFailure
    Current Depth  =          0  Max Depth  =          4
    Size           =        496  Max Alloc  =       1984
    AllocateMisses =          8  FreeMisses =          0
    TotalAllocates =     272492  TotalFrees =     272488
    Hit Rate       =         99% Hit Rate   =        100%
```

# Требования

Развернуть таблицу

| | |
|---|---|
| DLL-библиотеки | Kdexts.dll |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Lpc

Статья • 26.10.2023

**Важно** Теперь Lpc эмулируется в alpc, используйте расширение !alpc.

Расширение **!lpc** отображает сведения обо всех портах вызова локальных процедур (LPC) и сообщениях в целевой системе.

dbgcmd

```
!lpc message MessageID
!lpc port Port
!lpc scan Port
!lpc thread Thread
!lpc PoolSearch
!lpc
```

# Параметры

**Сообщение**
(Только Windows Server 2003, Windows XP и Windows 2000) Отображает сведения о сообщении, например порт сервера, который содержит сообщение в очереди, и поток, ожидающий этого сообщения, если таковой имеется.

*Messageid*
(Только Windows Server 2003, Windows XP и Windows 2000) Указывает идентификатор отображаемого сообщения. Если значение этого параметра равно 0 или этот параметр опущен, команда **!lpc message** отображает сводный список сообщений. (В Windows 2000 с пакетом обновления 1 (SP1) сводка включает все сообщения в зоне LPC. В Windows 2000 с пакетом обновления 2 (SP2), Windows XP и более поздних версиях Windows сводка включает все сообщения в пуле ядра. Постраничные сообщения не включаются.)

**Порт**
(Только Windows Server 2003, Windows XP и Windows 2000) Отображает сведения о порте, такие как имя порта, его состояние семафора, сообщения в очередях, потоки в очереди запуска, число дескрипторов, ссылки и связанные порты.

**Сканирования**
(Только Windows Server 2003, Windows XP и Windows 2000) Отображает сводные сведения об указанном порту и обо всех подключенных к нему портах.

*Порт*
(Только Windows Server 2003, Windows XP и Windows 2000) Указывает шестнадцатеричный адрес отображаемого порта. Если используется команда **!lpc port** и *порт* равен 0 или опущен, отображается сводный список всех портов LPC. Если используется команда **!lpc scan** , *порт* должен указать адрес фактического порта.

**Поток**
(Только Windows Server 2003, Windows XP и Windows 2000) Отображает сведения о портах для всех портов, содержащих указанный поток, в очередях портов с очисткой.

*Поток*
(Только Windows Server 2003, Windows XP и Windows 2000) Задает шестнадцатеричный адрес потока. Если значение равно 0 или опущено, команда **потока !lpc** отображает сводный список всех потоков, выполняющих любые операции LPC.

**PoolSearch**
(Только Windows Server 2003 и Windows XP) Определяет, выполняет ли команда **!lpc** message поиск сообщений в пуле ядра. При каждом использовании **!lpc PoolSearch** этот параметр переключается или отключается (начальный параметр заключается в том, чтобы не выполнять поиск в пуле ядра). Это влияет только на команды **сообщений !lpc** , которые указывают ненулевое значение для *MessageID*.

## DLL

| | |
|---|---|
| **Windows 2000** | Kdextx86.dll |
| **Windows XP**<br>**Windows Server 2003** | Kdexts.dll |

## Дополнительные сведения

Сведения о LPCs см. в документации по комплекту драйверов Windows (WDK) и *Microsoft Windows Internals* Марка Руссиновича и Дэвида Соломона.

# Комментарии

Это расширение не поддерживается в Windows Vista и более поздних версиях Windows.

В Windows Server 2003, Windows XP и Windows 2000 при использовании **!lpc** без аргументов отображается справка по этому расширению в окне Команды отладчика.

Если у вас есть поток, который помечается как ожидающий ответа на сообщение, используйте команду **!lpc message** с идентификатором отложенного сообщения. Эта команда отображает указанное сообщение, порт, который его содержит, и все связанные потоки.

Если сообщение не найдено и не было ошибок чтения (например, "Не удалось получить доступ к сегменту зоны"), сервер получил сообщение.

В этом случае порт сервера обычно можно найти с помощью команды **потока !lpc** . Потоки, ожидающие ответов, связываются с очередью связи сервера. Эта команда отобразит все порты, содержащие указанный поток. После того как вы узнаете адрес порта, используйте команду **!lpc port** . Затем можно получить более конкретные сведения о каждом потоке с помощью команды **!lpc thread**с адресом каждого потока.

Ниже приведено несколько примеров выходных данных этого расширения из системы Windows XP:

В этом примере отображаются все порты LPC.

dbgcmd

```
kd> !lpc port
Scanning 225 objects
       1  Port: 0xe1405650 Connection: 0xe1405650  Communication: 0x00000000
'SeRmCommandPort'
       1  Port: 0xe141ef50 Connection: 0xe141ef50  Communication: 0x00000000
'SmApiPort'
       1  Port: 0xe13c5740 Connection: 0xe13c5740  Communication: 0x00000000
'ApiPort'
       1  Port: 0xe13d9550 Connection: 0xe13d9550  Communication: 0x00000000
'SbApiPort'
       3  Port: 0xe13d8830 Connection: 0xe141ef50  Communication: 0xe13d8910
'
80000004  Port: 0xe13d8910 Connection: 0xe141ef50  Communication: 0xe13d8830
'
       3  Port: 0xe13d8750 Connection: 0xe13d9550  Communication: 0xe13a4030
'
       .....
```

В предыдущем примере порт по адресу e14ae238 не содержит сообщений; то есть все сообщения были приняты, и никаких новых сообщений не поступало.

dbgcmd

```
kd> !lpc port e14ae238

Server connection port e14ae238  Name: ApiPort
 Handles: 1   References: 107
    Server process  : 84aa0140 (csrss.exe)
    Queue semaphore : 84a96da8
 Semaphore state 0 (0x0)
    The message queue is empty
    The LpcDataInfoChainHead queue is empty
```

В предыдущем примере порт на 0xe14ae238 содержит сообщения, которые были поставлены в очередь, но еще не были приняты сервером.

dbgcmd

```
kd> !lpc port 0xe14ae238

Server connection port e14ae238  Name: ApiPort
 Handles: 1   References: 108
    Server process  : 84aa0140 (csrss.exe)
    Queue semaphore : 84a96da8
 Semaphore state 0 (0x0)
        Messages in queue:
 0000 e20d9b80 - Busy  Id=0002249c  From: 0584.0680  Context=00000021
[e14ae248 . e14ae248]
 Length=0098007c  Type=00000001 (LPC_REQUEST)
                  Data: 00000000 0002021e 00000584 00000680 002f0001
00000007
    The message queue contains 1 messages
    The LpcDataInfoChainHead queue is empty
```

Остальные примеры Windows XP относятся к другим параметрам, которые можно использовать с этим расширением.

dbgcmd

```
kd> !lpc message 222be
Searching message 222be in threads ...
Client thread 842a4db0 waiting a reply from 222be
Searching thread 842a4db0 in port rundown queues ...

Server communication port 0xe114a3c0
    Handles: 1   References: 1
    The LpcDataInfoChainHead queue is empty
        Connected port: 0xe1e7b948      Server connection port: 0xe14ae238

Client communication port 0xe1e7b948
    Handles: 1   References: 3
```

```
    The LpcDataInfoChainHead queue is empty

Server connection port e14ae238  Name: ApiPort
 Handles: 1   References: 107
    Server process  : 84aa0140 (csrss.exe)
    Queue semaphore : 84a96da8
 Semaphore state 0 (0x0)
    The message queue is empty
    The LpcDataInfoChainHead queue is empty
Done.
```

dbgcmd

```
kd> !lpc thread 842a4db0
Searching thread 842a4db0 in port rundown queues ...

Server communication port 0xe114a3c0
    Handles: 1   References: 1
    The LpcDataInfoChainHead queue is empty
        Connected port: 0xe1e7b948      Server connection port: 0xe14ae238

Client communication port 0xe1e7b948
    Handles: 1   References: 3
    The LpcDataInfoChainHead queue is empty

Server connection port e14ae238  Name: ApiPort
 Handles: 1   References: 107
    Server process  : 84aa0140 (csrss.exe)
    Queue semaphore : 84a96da8
 Semaphore state 0 (0x0)
    The message queue is empty
    The LpcDataInfoChainHead queue is empty
```

dbgcmd

```
kd> !lpc scan e13d8830
Scanning 225 objects
       3  Port: 0xe13d8830 Connection: 0xe141ef50  Communication: 0xe13d8910
'
80000004  Port: 0xe13d8910 Connection: 0xe141ef50  Communication: 0xe13d8830
'
Scanning 3 objects
```

# !Mca

Статья • 05.04.2024

Расширение !mca отображает регистры проверка архитектуры компьютера (MCA).

```
dbgcmd
```

```
!mca
```

# Параметры

*Адрес*
(Только целевой объект Itanium) Указывает адрес записи об ошибке MCA.

*Flags*
(Только целевой объект Itanium) Указывает уровень выходных данных. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0xFF, которое отображает все разделы, присутствующих в журнале.

Bit 0 (0x1)
Отображает раздел процессора.

Бит 1 (0x2)
Отображает раздел для конкретной платформы.

Бит 2 (0x4)
Отображает раздел памяти.

Бит 3 (0x8)
Отображает раздел компонента PCI.

Бит 4 (0x10)
Отображает раздел шины PCI.

Бит 5 (0x20)
Отображает раздел "Журнал SystemEvent".

Бит 6 (0x40)
Отображает раздел контроллера узла платформы.

Бит 7 (0x80)
Отображается для включения раздела шины платформы.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x86.

# Замечания

В целевом объекте x86 !mca **отображает проверка регистры компьютера,** поддерживаемые активным процессором. В нем также отображаются основные сведения о ЦП (идентичные данным, отображаемым **в !cpuinfo**). Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
0: kd> !mca
MCE: Enabled, Cycle Address: 0x00000001699f7a00, Type: 0x0000000000000000

MCA: Enabled, Banks 5, Control Reg: Supported, Machine Check: None.
Bank  Error  Control Register     Status Register
  0. None   0x000000000000007f   0x0000000000000000

  1. None   0x00000000ffffffff   0x0000000000000000

  2. None   0x000000000000ffff   0x0000000000000000

  3. None   0x0000000000000007   0x0000000000000000

  4. None   0x0000000000003fff   0x0000000000000000

No register state available.

CP F/M/S Manufacturer   MHz Update Signature Features
 0 15,5,0 SomeBrandName 1394 0000000000000000 a0017fff
```

Обратите внимание, что для этого расширения требуются частные символы HAL. Без этих символов расширение отобразит сообщение "HalpFeatureBits не найдено" вместе с основными сведениями о ЦП. Например:

dbgcmd

```
kd> !mca
HalpFeatureBits not found
CP F/M/S Manufacturer  MHz Update Signature Features
 0 6,5,1 GenuineIntel  334 0000004000000000 00001fff
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !memlist

Статья • 08.07.2024

**Расширение !memlist** сканирует списки физической памяти из базы данных кадров страницы (PFN), чтобы проверить их согласованность.

Синтаксис

```
!memlist Flags
```

## Параметры

*Флаги*
Указывает, какие списки памяти необходимо проверить. В настоящее время реализовано только одно значение:

Bit 0 (0x1)
Приводит к проверке списка страниц с нуля.

## DLL-библиотеки

Kdexts.dll

## Замечания

Это расширение проверяет только список страниц с нуля, чтобы убедиться, что все страницы в этом списке ноль. Правильный синтаксис:

dbgcmd

```
kd> !memlist 1
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !memusage

Статья • 05.04.2024

Расширение **!memusage** отображает сводную статистику использования физической памяти.

Синтаксис

```
!memusage [Flags]
```

## Параметры

*Flags*
Может быть любым из следующих значений. Значение по умолчанию — 0x0.

0x0
Отображает общие сводные сведения, а также более подробное описание страниц в базе данных PFN. Пример этого типа выходных данных см. в разделе "Примечания".

0x1
Отображает только сводные сведения об измененных страницах без записи в базе данных PFN.

0x2
Отображает только подробные сведения об измененных страницах без записи в базе данных PFN.

0x8
Отображает только общие сводные сведения об использовании памяти.

## Среда

**Режимы**: только режим ядра

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Статистика физической памяти собирается из таблицы страницы диспетчера памяти (PFN).

Эта команда занимает много времени, особенно если целевой компьютер работает в 64-разрядном режиме, из-за большего объема данных для получения. При загрузке базы данных PFN счетчик показывает его ход выполнения. Чтобы ускорить загрузку, используйте сетевое подключение или увеличьте скорость COM-порта с **помощью клавиш CTRL+A (переключение скорости Baud) или используйте команду кэша (задать размер кэша)** для увеличения размера кэша (возможно, около 10 МБ).

Команду **!memusage** также можно использовать при выполнении локальной отладки ядра.

Ниже приведен пример выходных данных из этого расширения:

dbgcmd

```
kd> !memusage
loading PFN database
loading (100% complete)
Compiling memory usage data (99% Complete).
              Zeroed:      218 (     872 kb)
                Free:      831 (    3324 kb)
             Standby:   124049 (  496196 kb)
            Modified:    55101 (  220404 kb)
    ModifiedNoWrite:       58 (     232 kb)
       Active/Valid:   321846 ( 1287384 kb)
         Transition:        8 (      32 kb)
         SLIST/Temp:     1533 (    6132 kb)
                Bad:        0 (       0 kb)
            Unknown:        0 (       0 kb)
              TOTAL:   503644 ( 2014576 kb)

Dangling Yes Commit:      184 (     736 kb)
 Dangling No Commit:    81706 (  326824 kb)
  Building kernel map
  Finished building kernel map
Scanning PFN database - (100% complete)

...
```

Кроме того, в отчете содержатся подробные сведения об использовании памяти, видимой отладчику.

dbgcmd

```
  Usage Summary (in Kb):
Control       Valid Standby Dirty Shared Locked PageTables  name
ffffaf0fb369f010   204    956     0    32   204     0  mapped_file(
shell32.dll )
ffffaf0fb369f270   492     60     0   252   492     0  mapped_file(
KernelBase.dll )
ffffaf0fb36ad050    20     36     0     0    20     0  mapped_file(
WMIsvc.dll )
ffffaf0fb36adad0    88    144     0    40    88     0  mapped_file( Can't
read file name buffer at ffffc10e0497e170 )
ffffaf0fb36b5670   780   1012     0   560   780     0  mapped_file(
KernelBase.dll )
ffffaf0fb36b5910    44    144     0    28    44     0  mapped_file(
cfgmgr32.dll )
ffffaf0fb36bc270     8      0     0     0     8     0  mapped_file( Can't
read file name buffer at ffffc10e061a17d0 )
ffffaf0fb36bc520    24     56     0     4    24     0  mapped_file(
ShareHost.dll )

...
```

Первый столбец отображает адрес структуры области управления, описывающей каждую сопоставленную структуру. **Используйте команду расширения !ca** для отображения этих областей управления.

## Замечания

Для анализа использования виртуальной **памяти можно использовать команду расширения !vm** . Обычно это расширение более полезно, чем **!memusage**. Дополнительные сведения об управлении памятью см. в статье *Microsoft Windows Internals*, Павел Yosifovich, Андреа Allievi, Алекс Ионеску, Марк Руссинович и Дэвид Соломон.

**Команда расширения !pfn** может использоваться для отображения определенной записи кадра страницы в базе данных PFN.

Расширение **!pool** отображает сведения о выделении определенного пула или о всем системном пуле.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !Депутаты

Статья • 05.04.2024

Расширение **!mps** отображает сведения BIOS для спецификации Intel Multiprocessor (MPS) целевого компьютера.

dbgcmd

```
!mps [Address]
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес таблицы MPS в BIOS. Если это опущено, информация получается из HAL. Для этого потребуются символы HAL.

### DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x86.

## Дополнительная информация

Дополнительные сведения об отладке BIOS см. в разделе "Отладка BIOS Code". Дополнительные сведения о MPS см. в соответствующей версии спецификации Intel MultiProcessor.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !mtrr

Статья • 05.04.2024

Расширение **!mtrr отображает содержимое регистра MTRR** .

dbgcmd

```
!mtrr
```

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x86.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !npx

Статья • 05.04.2024

Расширение **!npx** отображает содержимое области сохранения регистра с плавающей запятой.

dbgcmd

```
!npx Address
```

## Параметры

*Адрес*
Задает шестнадцатеричный адрес структуры FLOATING_SAVE_AREA.

## DLL-библиотеки

Kdexts.dll

## Замечания

Эта команда расширения может использоваться только с целевым компьютером на основе x86.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ob, !od, !ow

Статья • 05.04.2024

**Команды расширения !ob**, **!od** и **!ow** устарели. **Вместо этого используйте команды ob, od, ow (output to Port).**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Объекта

Статья • 05.04.2024

Расширение **!object** отображает сведения о системном объекте.

dbgcmd

```
!object Address [Flags]
!object Path
!object 0 Name
!object -p
!object {-h|-?}
```

# Параметры

*Адрес*
Если первый аргумент является шестнадцатеричным числом ненулевого, он указывает шестнадцатеричный адрес отображаемого системного объекта.

*Flags*
Указывает уровень детализации в выходных данных команды.

Задайте *флаги* битовой или из следующих значений:

**0x0**
Отображение типа объекта.

**0x1**
Отображение типа объекта, имени объекта и счетчиков ссылок.

**0x8**
Отображение содержимого каталога объекта или целевого объекта символьной ссылки. Этот флаг действует только в том случае, если **0x1** также задано.

**0x10**
Отображение необязательных заголовков объектов.

**0x20**
Отображение полного пути к именованном объекту. Этот флаг действует только в том случае, если **0x1** также задано.

Параметр *Flags* является необязательным. Значение по умолчанию — 0x9.

*Путь*
Если первый аргумент начинается с обратной косой черты (), **!object** интерпретирует его как имя пути объекта. Если этот параметр используется, отображение будет упорядочено в соответствии со структурой каталогов, используемой диспетчером объектов.

*Имя*
Если первый аргумент равен нулю, второй аргумент интерпретируется как имя класса системных объектов, для которых отображаются все экземпляры.

**-p**
Отображение пространств имен частных объектов.

{-**h**|-**?**}
Отображение справки по этой команде.

# DLL-библиотеки

Kdexts.dll

## Примеры

В этом примере передается путь к каталогу \Device в **!object**. Выходные данные перечисляют все объекты в каталоге \Device.

dbgcmd

```
0: kd> !object \Device
Object: ffffc00b074166a0  Type: (ffffe0083b768690) Directory
    ObjectHeader: ffffc00b07416670 (new version)
    HandleCount: 0  PointerCount: 224
    Directory Object: ffffc00b074092e0  Name: Device

    Hash Address          Type          Name
    ---- -------          ----          ----
     00  ffffe0083e6a61f0 Device        00000044
         ffffe0083dcc4050 Device        00000030
         ffffe0083d34f050 Device        NDMP2
         ffffe0083bdf7060 Device        NTPNP_PCI0002
         ...
         ffffe0083b85d060 Device        USBPDO-8
         ffffe0083d33d050 Device        USBFDO-6
         ...
         ffffe0083bdf0060 Device        NTPNP_PCI0001
```

Выберите один из перечисленных объектов, например USBPDO-8. Передайте адрес USBPDO-8 (ffffe0083b85d060) в **!objec**t. Задайте *флаги* для 0x0 получения минимальной информации.

dbgcmd

```
0: kd> !object ffffe0083b85d060 0x0
Object: ffffe0083b85d060  Type: (ffffe0083b87df20) Device
    ObjectHeader: ffffe0083b85d030 (new version)
```

Включите сведения о имени и счетчике ссылок для одного объекта, задав *флаги* для 0x1.

dbgcmd

```
0: kd> !object ffffe0083b85d060 0x1
Object: ffffe0083b85d060  Type: (ffffe0083b87df20) Device
    ObjectHeader: ffffe0083b85d030 (new version)
    HandleCount: 0  PointerCount: 6
    Directory Object: ffffc00b074166a0  Name: USBPDO-8
```

Получите необязательные сведения о заголовке для одного объекта, задав *флаги* для 0x10.

dbgcmd

```
0: kd> !object ffffe0083b85d060 0x10
Object: ffffe0083b85d060  Type: (ffffe0083b87df20) Device
    ObjectHeader: ffffe0083b85d030 (new version)
Optional Headers:
    NameInfo(ffffe0083b85d010)
```

В следующем примере дважды вызывается **!object** для объекта Directory. При первом отображении содержимого каталога не отображается, так как флаг 0x8 не задан. Во второй раз содержимое каталога отображается, так как 0x8 и флаги 0x1 заданы (флаги = 0x9).

dbgcmd

```
0: kd> !object ffffc00b07481d00 0x1
Object: ffffc00b07481d00  Type: (ffffe0083b768690) Directory
    ObjectHeader: ffffc00b07481cd0 (new version)
    HandleCount: 0  PointerCount: 3
    Directory Object: ffffc00b07481eb0  Name: Filters

0: kd> !object ffffc00b07481d00 0x9
Object: ffffc00b07481d00  Type: (ffffe0083b768690) Directory
```

```
    ObjectHeader: ffffc00b07481cd0 (new version)
    HandleCount: 0  PointerCount: 3
    Directory Object: ffffc00b07481eb0  Name: Filters

    Hash Address          Type          Name
    ---- -------          ----          ----
     19  ffffe0083c5f56e0 Device        FltMgrMsg
     21  ffffe0083c5f5060 Device        FltMgr
```

В следующем примере дважды вызывается **!object** для объекта SymbolicLink. Первый раз целевой объект символьной ссылки не отображается, так как флаг 0x8 не задан. Во второй раз целевой объект символьной ссылки воспроизводится, так как заданы флаги 0x8 и 0x1 (Флаги = 0x9).

dbgcmd

```
0: kd> !object ffffc00b07628fb0 0x1
Object: ffffc00b07628fb0  Type: (ffffe0083b769450) SymbolicLink
    ObjectHeader: ffffc00b07628f80 (new version)
    HandleCount: 0  PointerCount: 1
    Directory Object: ffffc00b074166a0  Name: Ip6

0: kd> !object ffffc00b07628fb0 0x9
Object: ffffc00b07628fb0  Type: (ffffe0083b769450) SymbolicLink
    ObjectHeader: ffffc00b07628f80 (new version)
    HandleCount: 0  PointerCount: 1
    Directory Object: ffffc00b074166a0  Name: Ip6
    Target String is '\Device\Tdx'
```

# Дополнительная информация

Дополнительные сведения об объектах и диспетчере объектов см. в документации по пакету SDK для Microsoft Windows, документации по комплекту драйверов Windows (WDK) и *внутренним службам* Microsoft Windows, марку Руссиновичу и Дэвиду Соломону.

# См. также

Трассировка ссылок на объекты

**!obtrace**

**!Обрабатывать**

Определение ACL объекта

Команды расширения в режиме ядра

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !obtrace

Статья • 05.04.2024

Расширение **!obtrace** отображает данные трассировки ссылок на объекты для указанного объекта.

```
dbgcmd
```

```
!obtrace Object
```

## Параметры

*Объект*
Указатель на объект или путь.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения о программе Global Flags (GFlags) см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Функция трассировки ссылок на объект для записей Windows записывает последовательные трассировки стека при добавочном или уменьшении счетчика ссылок на объект.

Перед использованием этого расширения для отображения данных трассировки ссылок на объект необходимо использовать GFlags , чтобы включить трассировку ссылок на объект для указанного объекта. Можно включить трассировку ссылок на объект в качестве параметра флага ядра (время выполнения), в котором изменение действует немедленно, но исчезает при завершении работы или перезапуске; или как параметр реестра, который требует перезапуска, но остается эффективным, пока не измените его.

Ниже приведен пример выходных данных расширения **!obtrace** :

dbgcmd

```
kd> !obtrace 0xfa96f700
Object: fa96f700        Image: cmd.exe
Sequence  (+/-)  Stack
--------  -----  --------------------------------------------------
   2421d    +1  nt!ObCreateObject+180
                nt!NtCreateEvent+92
                nt!KiFastCallEntry+104
                nt!ZwCreateEvent+11
                win32k!UserThreadCallout+6f
                win32k!W32pThreadCallout+38
                nt!PsConvertToGuiThread+174
                nt!KiBBTUnexpectedRange+c

   2421e    -1  nt!ObfDereferenceObject+19
                nt!NtCreateEvent+d4
                nt!KiFastCallEntry+104
                nt!ZwCreateEvent+11
                win32k!UserThreadCallout+6f
                win32k!W32pThreadCallout+38
                nt!PsConvertToGuiThread+174
                nt!KiBBTUnexpectedRange+c

   2421f    +1  nt!ObReferenceObjectByHandle+1c3
                win32k!xxxCreateThreadInfo+37d
                win32k!UserThreadCallout+6f
                win32k!W32pThreadCallout+38
                nt!PsConvertToGuiThread+174
                nt!KiBBTUnexpectedRange+c

   24220    +1  nt!ObReferenceObjectByHandle+1c3
                win32k!ProtectHandle+22
                win32k!xxxCreateThreadInfo+3a0
                win32k!UserThreadCallout+6f
                win32k!W32pThreadCallout+38
                nt!PsConvertToGuiThread+174
                nt!KiBBTUnexpectedRange+c

   24221    -1  nt!ObfDereferenceObject+19
                win32k!xxxCreateThreadInfo+3a0
                win32k!UserThreadCallout+6f
                win32k!W32pThreadCallout+38
                nt!PsConvertToGuiThread+174
                nt!KiBBTUnexpectedRange+c

----  -----------------------------------------------------------
References: 3, Dereferences 2
```

Основные индикаторы в **отображении !obtrace 0xfa96f700** перечислены в следующей таблице.

Развернуть таблицу

| Параметр | Значение |
|---|---|
| **Последовательность** | Представляет порядок операций. |
| +/- | Указывает ссылку или операцию разыменования.<br><br>+1 указывает на справочную операцию.<br><br>-1 указывает операцию разыменования.<br><br>+/- n указывает несколько операций ссылок и разыменовок. |

Трассировки ссылок на объекты на целевых компьютерах на основе x64 могут быть неполными, так как не всегда невозможно получить трассировки стека на уровнях IRQL выше, чем PASSIVE_LEVEL.

Вы можете остановить выполнение в любое время, нажав клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !openmaps

Статья • 08.07.2024

Расширение **!openmaps отображает блоки управления буфером со ссылкой (BCB) и блоки** управления виртуальными адресами (VACB) для указанной карты общего кэша.

dbgcmd

```
!openmaps Address [Flag]
```

# Параметры

*Адрес*
Указывает адрес карты общего кэша.

*Флаг*
Определяет, какие блоки управления отображаются. Если *флаг* равен **1**, отладчик отображает все блоки управления. Если *флаг* равен **0**, отладчик отображает только блоки управления, на которые ссылается ссылка. По умолчанию установлено значение **0**.

## DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения об управлении кэшем см. в документации по microsoft Windows SDK и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

Дополнительные сведения о других расширениях управления кэшем см. в расширении !cchelp.

---

# Обратная связь

**Были ли сведения на этой странице полезными?**

Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Парс

Статья • 05.04.2024

Расширение **!pars** отображает указанный файл регистров приложения обработчика.

dbgcmd

```
!pars Address
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Адрес*
Указывает адрес файла регистров приложения обработчика.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе Itanium.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !похлопывать

Статья • 08.07.2024

Расширение **!pat** отображает регистры таблицы атрибутов страницы (PAT) для целевого процессора.

dbgcmd

```
!pat Flag
!pat
```

## Параметры

*Флаг*
Если *установлен флаг* , отладчик проверяет наличие функции PAT перед отображением PAT.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x86.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Pci

Статья • 05.04.2024

Расширение **!pci** отображает текущее состояние периферийных компонентов шины взаимодействия (PCI), а также все устройства, подключенные к этим автобусам.

dbgcmd

```
!pci [Flags [Segment] [Bus [Device [Function [MinAddress MaxAddress]]]]]
```

## Параметры

*Flags*
Указывает уровень выходных данных. Может быть любое сочетание следующих битов:

Bit 0 (0x1)
Приводит к подробному отображению.

Бит 1 (0x2)
Вызывает отображение для включения всех автобусов в диапазон от 0 (ноль) до указанной *шины*.

Бит 2 (0x4)
Приводит к включению сведений в необработанный байтовый формат. Если *задано значение MinAddress, MaxAddress* или пометка бита 0x8, этот бит также устанавливается автоматически.

Бит 3 (0x8)
Приводит к включению сведений в необработанный формат DWORD.

Бит 4 (0x10)
Приводит к включению недопустимых номеров устройств. Если *указано устройство* , этот флаг игнорируется.

Бит 5 (0x20)
Приводит к тому, что отображение включает недопустимые номера функций.

Бит 6 (0x40)
Приводит к включению возможностей дисплея.

Бит 7 (0x80)
Приводит к включению сведений об устройстве Intel 8086.

Бит 8 (0x100)
Приводит к включению пространства конфигурации PCI.

Bit 9 (0x200)
Приводит к включению сведений о сегменте. Если этот бит включен, *необходимо включить параметр Segment* .

Бит 10 (0x400)
Приводит к включению всех допустимых сегментов в диапазон от сегмента 0 до указанного сегмента. Если этот бит включен, *необходимо включить параметр Segment* .

*Сегмент*
Указывает число отображаемых сегментов. Номера сегментов варьируются от 0 до 0xFFFF. Если *сегмент* опущен, отображаются сведения о первичном сегменте (сегменте 0). Если *флаги* включают бит 10 (0x400), *сегмент* указывает самый высокий допустимый сегмент для отображения.

*Автобус*
Задает отображаемую шину. *Шина* может варьироваться от 0 до 0xFF. Если он опущен, отображаются сведения о первичной шине (шине 0). Если *флаги* включают бит 1 (0x2), *шина* указывает самый высокий номер шины для отображения.

*Устройство*
Указывает номер устройства слота для устройства. Если это опущено, данные обо всех устройствах печатаются.

*Function*
Указывает номер функции слота для устройства. Если это опущено, все сведения обо всех функциях устройства печатаются.

*MinAddress*
Указывает первый адрес, из которого должны отображаться необработанные байты или DWORD. Это должно быть от 0 до 0xFF.

*MaxAddress*
Указывает последний адрес, из которого должны отображаться необработанные байты или DWORD. Это должно быть от 0 до 0xFF, а не меньше *MinAddress*.

# DLL-библиотеки

Kext.dll

Эта команда расширения может использоваться только с целевым компьютером на основе x86.

# Дополнительная информация

Дополнительные примеры см. в разделе самонастраивающийся Отладка для приложений этой команды расширения. Сведения о шинах PCI см. в документации по комплекту драйверов Windows (WDK).

# Замечания

Чтобы изменить пространство конфигурации PCI, используйте !ecb, **!ecd или !ecw.**

В следующем примере отображается список всех автобусов и их устройств. Эта команда займет много времени. Вы увидите счетчик перемещения в нижней части дисплея, пока отладчик сканирует целевую систему для автобусов PCI:

dbgcmd

```
kd> !pci 2 ff
PCI Bus 0
00:0  8086:1237.02  Cmd[0106:.mb..s]  Sts[2280:.....]  Device  Host bridge
0d:0  8086:7000.01  Cmd[0007:imb...]  Sts[0280:.....]  Device  ISA bridge
0d:1  8086:7010.00  Cmd[0005:i.b...]  Sts[0280:.....]  Device  IDE
controller
0e:0  1011:0021.02  Cmd[0107:imb..s]  Sts[0280:.....]  PciBridge 0->1-1
PCI-PCI bridge
10:0  102b:0519.01  Cmd[0083:im....]  Sts[0280:.....]  Device  VGA
compatible controller
PCI Bus 1
08:0  10b7:9050.00  Cmd[0107:imb..s]  Sts[0200:.....]  Device  Ethernet
09:0  9004:8178.00  Cmd[0117:imb..s]  Sts[0280:.....]  Device  SCSI
controller
```

В этом примере отображаются подробные сведения об устройствах на первичной шине. Двухзначное число в начале каждой строки — номер устройства; Однозначное число, следующее за номером функции:

dbgcmd

```
kd> !pci 1 0
PCI Bus 0
00:0  8086:1237.02  Cmd[0106:.mb..s]  Sts[2280:.....]  Device  Host bridge
      cf8:80000000  IntPin:0  IntLine:0  Rom:0  cis:0  cap:0

0d:0  8086:7000.01  Cmd[0007:imb...]  Sts[0280:.....]  Device  ISA bridge
      cf8:80006800  IntPin:0  IntLine:0  Rom:0  cis:0  cap:0

0d:1  8086:7010.00  Cmd[0005:i.b...]  Sts[0280:.....]  Device  IDE
controller
      cf8:80006900  IntPin:0  IntLine:0  Rom:0  cis:0  cap:0
      IO[4]:fff1

0e:0  1011:0021.02  Cmd[0107:imb..s]  Sts[0280:.....]  PciBridge 0->1-1
PCI-PCI bridge
      cf8:80007000  IntPin:0  IntLine:0  Rom:0  cap:0  2sts:2280  BCtrl:6
ISA
      IO:f000-ffff  Mem:fc000000-fdffffff  PMem:fff00000-fffff

10:0  102b:0519.01  Cmd[0083:im....]  Sts[0280:.....]  Device  VGA
compatible controller
      cf8:80008000  IntPin:1  IntLine:9  Rom:80000000  cis:0  cap:0
      MEM[0]:fe800000  MPF[1]:fe000008
```

В этом примере показаны еще более подробные сведения об шине 0 (ноль), 0x0D устройства и функции 0x1, включая необработанные DWORDS из адресов между 0x00 и 0x3F:

dbgcmd

```
kd> !pci f 0 d 1 0 3f
PCI Bus 0
0d:1  8086:7010.00  Cmd[0005:i.b...]  Sts[0280:.....]  Device  IDE
controller
      cf8:80006900  IntPin:0  IntLine:0  Rom:0  cis:0  cap:0
      IO[4]:fff1
      00000000:  70108086 02800005 01018000 00002000
      00000010:  00000000 00000000 00000000 00000000
      00000020:  0000fff1 00000000 00000000 00000000
      00000030:  00000000 00000000 00000000 00000000
```

В этом примере отображается пространство конфигурации для сегмента 1, шины 0, устройства 1:

dbgcmd

```
0: kd> !pci 301 1 0 1

PCI Configuration Space (Segment:0001 Bus:00 Device:01 Function:00)
Common Header:
```

```
    00: VendorID       14e4 Broadcom Corporation
    02: DeviceID       16c7
    04: Command        0146 MemSpaceEn BusInitiate PERREn SERREn
    06: Status         02b0 CapList 66MHzCapable FB2BCapable DEVSELTiming:1
.
.
.
    5a: MsgCtrl        64BitCapable MultipleMsgEnable:0 (0x1)
MultipleMsgCapable:3 (0x8)
    5c: MsgAddr        2d4bff00
    60: MsgAddrHi      1ae09097
    64: MsData         9891
```

Чтобы отобразить все устройства и автобусы в допустимых сегментах, выполните команду **!pci 602 ffff ffff**:

dbgcmd

```
0: kd> !pci 602 ffff ff
Scanning the following PCI segments: 0 0x1
PCI Segment 0 Bus 0
01:0  14e4:16c7.10  Cmd[0146:.mb.ps]  Sts[02b0:c6...]  Ethernet Controller
SubID:103c:1321
02:0  1000:0030.08  Cmd[0147:imb.ps]  Sts[0230:c6...]  LSI SCSI Controller
SubID:103c:1323
02:1  1000:0030.08  Cmd[0147:imb.ps]  Sts[0230:c6...]  LSI SCSI Controller
SubID:103c:1323
03:0  1000:0030.08  Cmd[0147:imb.ps]  Sts[0230:c6...]  LSI SCSI Controller
SubID:103c:1323
03:1  1000:0030.08  Cmd[0147:imb.ps]  Sts[0230:c6...]  LSI SCSI Controller
SubID:103c:1323
PCI Segment 0 Bus 0x38
01:0  14e4:1644.12  Cmd[0146:.mb.ps]  Sts[02b0:c6...]  Ethernet Controller
SubID:10b7:1000
PCI Segment 0 Bus 0x54
00:0  103c:403b.00  Cmd[0547:imb.ps]  Sts[0010:c....]  HP PCI-PCI Bridge
0x54->0x55-0x55
PCI Segment 0 Bus 0x70
00:0  103c:403b.00  Cmd[0547:imb.ps]  Sts[0010:c....]  HP PCI-PCI Bridge
0x70->0x71-0x71
PCI Segment 0 Bus 0xa9
01:0  8086:b154.00  Cmd[0147:imb.ps]  Sts[0ab0:c6.A.]  Intel PCI-PCI Bridge
0xa9->0xaa-0xaa
PCI Segment 0 Bus 0xaa
04:0  1033:0035.41  Cmd[0146:.mb.ps]  Sts[0210:c....]  NEC USB Controller
SubID:103c:1293
04:1  1033:0035.41  Cmd[0146:.mb.ps]  Sts[0210:c....]  NEC USB Controller
SubID:103c:aa55
04:2  1033:00e0.02  Cmd[0146:.mb.ps]  Sts[0210:c....]  NEC USB2 Controller
SubID:103c:aa55
05:0  1002:5159.00  Cmd[0187:imb..s]  Sts[0290:c....]  ATI VGA Compatible
Controller  SubID:103c:1292
PCI Segment 0 Bus 0xc6
```

```
00:0  103c:403b.00  Cmd[0547:imb.ps]  Sts[0010:c....]  HP PCI-PCI Bridge
0xc6->0xc7-0xc7
PCI Segment 0 Bus 0xe3
00:0  103c:403b.00  Cmd[0547:imb.ps]  Sts[0010:c....]  HP PCI-PCI Bridge
0xe3->0xe4-0xe4
PCI Segment 0x1 Bus 0
01:0  14e4:16c7.10  Cmd[0146:.mb.ps]  Sts[02b0:c6...]  Ethernet Controller
SubID:103c:1321
02:0  1000:0030.08  Cmd[0147:imb.ps]  Sts[0230:c6...]  LSI SCSI Controller
SubID:103c:1323
02:1  1000:0030.08  Cmd[0147:imb.ps]  Sts[0230:c6...]  LSI SCSI Controller
SubID:103c:1323
03:0  1000:0030.08  Cmd[0147:imb.ps]  Sts[0230:c6...]  LSI SCSI Controller
SubID:103c:1323
03:1  1000:0030.08  Cmd[0147:imb.ps]  Sts[0230:c6...]  LSI SCSI Controller
SubID:103c:1323
PCI Segment 0x1 Bus 0x54
00:0  103c:403b.00  Cmd[0547:imb.ps]  Sts[0010:c....]  HP PCI-PCI Bridge
0x54->0x55-0x55
PCI Segment 0x1 Bus 0x55
00:0  8086:10b9.06  Cmd[0147:imb.ps]  Sts[0010:c....]  Intel Ethernet
Controller  SubID:8086:1083
PCI Segment 0x1 Bus 0x70
00:0  103c:403b.00  Cmd[0547:imb.ps]  Sts[0010:c....]  HP PCI-PCI Bridge
0x70->0x71-0x71
PCI Segment 0x1 Bus 0xc6
00:0  103c:403b.00  Cmd[0547:imb.ps]  Sts[0010:c....]  HP PCI-PCI Bridge
0xc6->0xc7-0xc7
PCI Segment 0x1 Bus 0xe3
00:0  103c:403b.00  Cmd[0547:imb.ps]  Sts[0010:c....]  HP PCI-PCI Bridge
0xe3->0xe4-0xe4
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pciir

Статья • 05.04.2024

Расширение **!pciir** отображает содержимое аппаратной маршрутизации периферийных компонентов межсоединения (PCI) устройств для прерывания входных данных контроллера.

dbgcmd

```
!pciir
```

## DLL-библиотеки

Windows XP — Kdexts.dll

Windows Vista и более поздних версий — недоступно

## Дополнительная информация

Эта команда расширения может использоваться только с целевым компьютером на основе x86, который не включает расширенную конфигурацию и power Interface (ACPI).

Для аналогичных сведений о любом компьютере с поддержкой ACPI используйте **расширение !acpiirqarb** .

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pcitree

Статья • 05.04.2024

Расширение **!pcitree** отображает сведения о объектах устройств PCI, включая дочерние автобусы PCI и автобусы CardBus, а также устройства, подключенные к ним.

dbgcmd

```
!pcitree
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Сведения о шинах PCI и объектах устройств PCI см. в документации по комплекту драйверов Windows (WDK).

## Замечания

Рассмотрим пример:

dbgcmd

```
kd> !pcitree

Bus 0x0 (FDO Ext fe517338)
  0600 12378086 (d=0,  f=0) devext fe4f4ee8 Bridge/HOST to PCI
  0601 70008086 (d=d,  f=0) devext fe4f4ce8 Bridge/PCI to ISA
  0101 70108086 (d=d,  f=1) devext fe4f4ae8 Mass Storage Controller/IDE
  0604 00211011 (d=e,  f=0) devext fe4f4788 Bridge/PCI to PCI

Bus 0x1 (FDO Ext fe516998)
  0200 905010b7 (d=8,  f=0) devext fe515ee8 Network Controller/Ethernet
  0100 81789004 (d=9,  f=0) devext fe515ce8 Mass Storage Controller/SCSI
  0300 0519102b (d=10, f=0) devext fe4f4428 Display Controller/VGA

Total PCI Root busses processed = 1
```

Чтобы понять это отображение, рассмотрим окончательное устройство. Его базовый класс равен 03, его подкласс равен 00, его идентификатор устройства 0x0519, и его идентификатор поставщика 0x102B. Эти значения являются встроенными для самого устройства.

Число после "d=" — это номер устройства; число после "f=" — это номер функции. После "devext" — адрес расширения устройства, 0xFE4F4428. Наконец, появится имя базового класса и подкласс.

Чтобы получить дополнительные сведения об устройстве, используйте **команду расширения !devext** с адресом расширения устройства в качестве аргумента. Для этого конкретного устройства будет использоваться команда:

dbgcmd

```
kd> !devext fe4f4428 pci
```

**Если расширение !pcitree** создает ошибку, это часто означает, что символы PCI не были загружены должным образом. Чтобы устранить эту проблему, используйте **reload pci.sys** .

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Pcm

Статья • 05.04.2024

Расширение **!pcm** отображает указанную карту частного кэша. Это расширение доступно только в Windows 2000.

```
dbgcmd
```

```
!pcm Address
```

## Параметры

*Адрес*
Указывает адрес карты частного кэша.

## DLL-библиотеки

Windows XP — Kdexts.dll

Windows Vista и более поздних версий — недоступно

## Дополнительная информация

Сведения об управлении кэшем см. в документации по microsoft Windows SDK и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

Дополнительные сведения о других расширениях управления кэшем см. в справочнике **по расширению !cchelp** .

## Замечания

Это расширение поддерживается только в Windows 2000. В Windows XP и более поздних версиях Windows используйте **команду dt nt!_PRIVATE_CACHE_MAP Address** .

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pcr

Статья • 08.07.2024

Расширение **!pcr** отображает текущее состояние области управления процессором (PCR) на определенном процессоре.

```
dbgcmd
```

```
!pcr [Processor]
```

## Параметры

*Процессор*
Указывает процессор, из которого извлекаются данные PCR. Если *обработчик* опущен, используется текущий процессор.

> ⓘ **Примечание**
>
> Эта команда в настоящее время не поддерживается и может отображать неправильные выходные данные.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о PCR и PRCB см. в microsoft *Windows Internals*, Марк Руссинович и Дэвид Соломон.

## Замечания

Блок управления процессором (PRCB) является расширением PCR. Его можно отобразить с расширением **!prcb** .

Ниже приведен пример **расширения !pcr** на целевом компьютере x86:

```
dbgcmd
```

```
kd> !pcr 0
KPCR for Processor 0 at ffdff000:
    Major 1 Minor 1
      NtTib.ExceptionList: 801626e0
          NtTib.StackBase: 801628f0
         NtTib.StackLimit: 8015fb00
       NtTib.SubSystemTib: 00000000
            NtTib.Version: 00000000
        NtTib.UserPointer: 00000000
            NtTib.SelfTib: 00000000

                  SelfPcr: ffdff000
                     Prcb: ffdff120
                     Irql: 00000000
                      IRR: 00000000
                      IDR: ffffffff
            InterruptMode: 00000000
                      IDT: 80043400
                      GDT: 80043000
                      TSS: 803cc000

            CurrentThread: 8015e8a0
               NextThread: 00000000
               IdleThread: 8015e8a0

                DpcQueue:  0x80168ee0 0x80100d04 ntoskrnl!KiTimerExpiration
```

Одна из записей в этом отображении показывает уровень запроса прерывания (IRQL). Расширение **!pcr** показывает текущий IRQL, но текущий IRQL обычно не является большим интересом. IRQL, который существовал непосредственно перед проверкой ошибок или подключением отладчика, более интересно. Это отображается **в !irql**, который доступен только на компьютерах под управлением Windows Server 2003 или более поздних версий Windows.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pcrs

Статья • 05.04.2024

Расширение **!pcrs** отображает регистры управления процессором Intel Itanium для конкретного процессора.

dbgcmd

```
!pcrs Address
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Адрес*
Указывает адрес файла регистров элемента управления процессором.

## DLL-библиотеки

Kdexts.dll

Эта команда расширения может использоваться только с целевым компьютером на основе Itanium.

## Замечания

Не путайте **расширение !pcrs** с **расширением !pcr** , которое отображает текущее состояние области управления процессором.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Pfn

Статья • 05.04.2024

Расширение **!pfn** отображает сведения о определенной рамке страницы или всей базе данных кадров страницы.

```
dbgcmd
```

```
!pfn PageFrame
```

## Параметры

*PageFrame*
Указывает шестнадцатеричное число отображаемого кадра страницы.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о таблицах страниц, каталогах страниц и кадрах *страниц см. в разделе* "Внутренние элементы Microsoft Windows", по Марку Руссиновичу и Дэвиду Соломону.

## Замечания

Номер кадра страницы для виртуального адреса можно получить с помощью **расширения !pte** .

Ниже приведен пример выходных данных из этого расширения:

```
dbgcmd
```

```
kd> !pte 801544f4
801544F4  - PDE at C0300800        PTE at C0200550
          contains 0003B163      contains 00154121
        pfn 3b G-DA--KWV    pfn 154 G--A--KRV

kd> !pfn 3b
    PFN 0000003B at address 82350588
```

```
    flink       00000000  blink / share count 00000221  pteaddress C0300800
    reference count 0001                                color 0
    restore pte 00000000  containing page        000039  Active


kd> !pfn 154
    PFN 00000154 at address 82351FE0
    flink       00000000  blink / share count 00000001  pteaddress C0200550
    reference count 0001                                color 0
    restore pte 00000060  containing page        00003B  Active     M
    Modified
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Pmc

Статья • 05.04.2024

Расширение **!pmc** отображает регистр счетчика Монитор производительности (PMC) по указанному адресу.

Это расширение поддерживается только на целевом компьютере на основе Itanium.

dbgcmd

```
!pmc [- Option] Expression [DisplayLevel]
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

# Параметры

*Параметр*
Может быть одним из следующих значений:

**Gen**
Отображает регистр в качестве универсального регистра PMC.

**Btb**
Отображает регистр в виде буфера трассировки ветви (В ТБ) PMC.

*Выражение*
Указывает шестнадцатеричный адрес PMC. Выражения **@kpfcgen** и **@kpfcbtb** можно использовать в качестве значений для этого параметра.

Если *выражение* @kpfcgen, отладчик отображает текущий регистр PMC процессора в качестве универсального регистра PMC. Вы также можете отобразить текущий регистр PMC процессора в качестве универсального регистра PMC, задав *параметр* **параметров ведения** и использования **@kpfc4**, @kpfc5, **@kpfc6** или **@kpfc7** для *значения выражения*.

Если *выражение* @kpfcbtb, отладчик отображает текущий регистр PMC процессора в качестве регистра В ТБ PMC. Вы также можете отобразить текущий регистр PMC процессора в качестве регистра В ТБ PMC, задав *параметр* "***Параметр*** **btb**" и используя @kpfc12 значение *выражения*.

*DisplayLevel*
Может быть одним из следующих значений:

**0**
Отображает только значения каждого поля регистрации PMC. Это значение по умолчанию.

**1**
Отображает подробные сведения о полях регистрации PMC, которые не зарезервированы или игнорируются.

**2**
Отображает подробные сведения обо всех полях регистрации PMC, включая те, которые игнорируются или зарезервированы.

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pmssa

Статья • 05.04.2024

Расширение **!pmssa** отображает указанную область сохранения состояния процессора (также называемую областью Min-StateSave).

Это расширение можно использовать только с целевым компьютером на основе Itanium.

dbgcmd

```
!pmssa Address
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Адрес*
Указывает адрес области Min-StateSave процессора.

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !powertriage

Статья • 05.04.2024

Расширение !powertriage отображает сводную информацию о компонентах, связанных с системой и устройством. Он также предоставляет ссылки на связанные команды, которые можно использовать для сбора дополнительных сведений. Команда !powertriage не имеет параметров. Эту команду можно использовать как с отладкой в динамическом режиме ядра, так и для анализа файлов аварийного дампа.

Синтаксис

```
dbgcmd
```

```
!powertriage
```

## Параметры

нет

## DLL-библиотеки

Windows 10 и более поздних версий — Kdexts.dll

## Замечания

Расширение !powertriage отображает следующие сведения.

1. Состояние питания узла устройства вместе с !podev для всех объектов устройства.
2. Ссылки на **!rcdrkd.rcdrlogdump** , если драйвер включил IFR. Дополнительные сведения об IFR см. в статье Об использовании средства записи трассировки inflight (IFR) в драйверах KMDF и UMDF 2.
3. Ссылки на **!wdfkd.wdfdriverinfo** и **!wdfkd.wdflogdump** для драйверов WDF.
4. Ссылки на !fxdevice для устройств PoFx. Дополнительные сведения о PoFX см. в разделе "Обзор платформы управления питанием". Ниже приведен пример выходных данных из команды !powertriage.

```
dbgcmd
```

```
kd> !powertriage

System Capabilities :
  Machine is not AOAC capable.

Power Capabilities:
PopCapabilities @ 0xfffff8022f6c4380
  Misc Supported Features:  PwrButton S1 S3 S4 S5 HiberFile FullWake
  Processor Features:
  Disk Features:
  Battery Features:
  Wake Caps
    Ac OnLine Wake:         Sx
    Soft Lid Wake:          Sx
    RTC Wake:               S4
    Min Device Wake:        Sx
    Default Wake:           Sx



Power Action:

PopAction :fffff8022f6ba550
    Current System State..: Working
    Target System State...: Unspecified
    State.................: - Idle(0)

Devices with allocated Power IRPs:

    +  ACPI\PNP0C0C\2&daba3ff&1
       0xffffe00023939ad0 ACPI D0 !podev  WAIT_WAKE_IRP !irp Related Threads

    +  USB\ROOT_HUB30\5&2c60645a&0&0
       0xffffe0002440ac40 USBXHCI D2 !podev  WAIT_WAKE_IRP !irp Related
Threads !rcdrlogdump !wdfdriverinfo !wdflogdump
         Upper DO 0xffffe00024415a10 USBHUB3 !podev


    +  USB\ROOT_HUB30\5&d91dce5&0&0
       0xffffe00023ed4d30 USBXHCI D2 !podev  WAIT_WAKE_IRP !irp Related
Threads !rcdrlogdump !wdfdriverinfo !wdflogdump
         Upper DO 0xffffe000249d8040 USBHUB3 !podev

    +  PCI\VEN_8086&DEV_27E2&SUBSYS_01DE1028&REV_01\3&172e68dd&0&E5
       0xffffe000239e5880 pci D0 !podev FxDevice: !fxdevice  WAIT_WAKE_IRP
!irp Related Threads
         Upper DO 0xffffe000239c0e50 ACPI !podev
           Upper DO 0xffffe000239f7040 pci !podev


    +  PCI\VEN_14E4&DEV_167A&SUBSYS_01DE1028&REV_02\4&24ac2e11&0&00E5
       0xffffe000231e6060 pci D0 !podev  WAIT_WAKE_IRP !irp Related Threads
         Upper DO 0xffffe00024359050 b57nd60a !podev
```

```
Device Tree Info:

    !devpowerstate

    !devpowerstate Complete


Links:
!poaction
!cstriage
!pdctriage
!pdcclients
!fxdevice
!pnptriage
```

## Анализ отказа файловой мощности дампа

Расширение !powertriage может оказаться полезным при проверке сбоев системы, связанных с неправильными сведениями о состоянии питания. Например, в **случае проверки ошибок 0x9F: DRIVER_POWER_STATE_FAILURE** расширение будет отображать все выделенные ip-адреса питания, связанные с устройствами стеками устройств, а также:

1. Ссылки на **команду !irp** для связанных irPs.
2. Ссылки на **команду !findthreads** с соответствующим IRP. IRP добавляется в рамках критериев поиска и отображает потоки, начиная с более высокой корреляции с потоками критериев поиска, перечисленными в первую очередь. Дамп всех стеков устройств с питанием irPs может помочь в отладке случаев, когда !анализ не смог правильно определить IRP, связанный с сбоем.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pnpevent

Статья • 05.04.2024

Расширение **!pnpevent** отображает очередь событий устройства самонастраивающийся.

dbgcmd

```
!pnpevent [DeviceEvent]
```

## Параметры

*DeviceEvent*
Указывает адрес отображаемого события устройства. Если это значение равно нулю или опущено, отображается дерево всех событий устройства в очереди.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Сведения о драйверах самонастраивающийся см. в документации по комплекту драйверов Windows (WDK).

## См. также

команды самонастраивающийся и Power Debugger

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pocaps

Статья • 05.04.2024

Расширение **!pocaps** отображает возможности питания целевого компьютера.

dbgcmd

```
!pocaps
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Чтобы просмотреть политику питания системы, используйте **команду расширения !popolicy** . Сведения о возможностях питания и политике питания см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, марку Руссиновичу и Дэвиду Соломону.

## Замечания

Ниже приведен пример выходных данных этой команды:

dbgcmd

```
kd> !pocaps
PopCapabilities @ 0x8016b100
  Misc Supported Features:  S4 FullWake
  Processor Features:
  Disk Features:            SpinDown
  Battery Features:
  Wake Caps
    Ac OnLine Wake:         Sx
    Soft Lid Wake:          Sx
    RTC Wake:               Sx
    Min Device Wake:        Sx
    Default Wake:           Sx
```

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Бассейн

Статья • 05.04.2024

Расширение **!pool** отображает сведения о выделении определенного пула или о всем системном пуле.

dbgcmd

```
!pool [Address [Flags]]
```

## Параметры

*Адрес*
Указывает отображаемую запись пула. Если *адрес* равен -1, эта команда отображает сведения обо всех кучах в процессе.

Если *адрес* равен 0 или опущен, эта команда отображает сведения о куче процесса.

*Flags*
Указывает уровень детализации, используемый. Это может быть любое сочетание следующих битовых значений; Значение по умолчанию равно нулю:

Bit 0 (0x1)
Позволяет отображать содержимое пула, а не только заголовки пула.

Бит 1 (0x2)
Приводит к подавлению сведений о заголовке пула для всех пулов, за исключением того, который фактически содержит указанный *адрес*.

Бит 31 (0x80000000)
Подавляет описание типа пула и тега пула в отображении.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о пулах памяти см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, марку Руссиновичу и Дэвиду

Соломону.

# Замечания

В Windows XP и более поздних версиях Windows **расширение пула !отображает тег пула** , связанный с каждым выделением. Также отображается владелец этого тега пула. Это отображение основано на содержимом файла pooltag.txt. Этот файл находится в подкаталоге для установки средств отладки для Windows. Если вы хотите, можно изменить этот файл, чтобы добавить дополнительные теги пула, относящиеся к проекту.

**Предупреждение при** установке обновленной версии средств отладки для Windows в том же каталоге, что и текущая версия, она перезаписывает все файлы в этом каталоге, включая pooltag.txt. При изменении или замене примера файла pooltag.txt обязательно сохраните копию в другом каталоге. После переустановки отладчиков можно скопировать сохраненные pooltag.txt по умолчанию.

Если пул расширений !pool **сообщает о повреждении пула, следует использовать !poolval для изучения.**

Ниже приведен пример. Если *адрес* указывает 0xE1001050, отображаются заголовки всех пулов в этом блоке, а сам 0xE1001050 помечается звездочкой (*).

dbgcmd

```
kd> !pool e1001050
 e1001000 size:   40 previous size:    0  (Allocated)  MmDT
 e1001040 size:   10 previous size:   40  (Free)       Mm
*e1001050 size:   10 previous size:   10  (Allocated) *ObDi
 e1001060 size:   10 previous size:   10  (Allocated)  ObDi
 e1001070 size:   10 previous size:   10  (Allocated)  Symt
 e1001080 size:   40 previous size:   10  (Allocated)  ObDm
 e10010c0 size:   10 previous size:   40  (Allocated)  ObDi
.....
```

В этом примере в правом столбце показан тег пула. Столбец слева от этого показывает, является ли пул свободным или выделенным.

Следующая команда показывает заголовки пула и содержимое пула:

dbgcmd

```
kd> !pool e1001050 1
 e1001000 size:   40 previous size:    0  (Allocated)  MmDT
 e1001008  ffffffff 0057005c 004e0049 004f0044
    e1001018  ffffffff 0053005c 00730079 00650074
```

```
 e1001040 size:   10 previous size:   40  (Free)       Mm
 e1001048  ffffffff e1007ba8 e1501a58 01028101
    e1001058  ffffffff 00000000 e1000240 01028101

*e1001050 size:   10 previous size:   10  (Allocated) *ObDi
 e1001058  ffffffff 00000000 e1000240 01028101
    e1001068  ffffffff 00000000 e10009c0 01028101

 e1001060 size:   10 previous size:   10  (Allocated)  ObDi
 e1001068  ffffffff 00000000 e10009c0 01028101
    e1001078  ffffffff 00000000 00000000 04028101

 ......
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !poolfind

Статья • 05.04.2024

Расширение **!poolfind** находит все экземпляры определенного тега пула в непагированных или страничных пулах памяти.

dbgcmd

```
!poolfind TagString [PoolType]
!poolfind TagValue [PoolType]
```

## Параметры

*TagString*
Указывает тег пула. *TagString* — это строка ASCII с учетом регистра. Звездочка (*) может использоваться для представления любого числа символов; Вопросительный знак (?) можно использовать для представления точно одного символа. Если звездочка не используется, *TagString* должен иметь ровно четыре символа в длину.

*TagValue*
Указывает тег пула. *TagValue* должен начинаться с "0x", даже если по умолчанию радикс равен 16. Если этот параметр начинается с любого другого значения (включая "0X"), он будет интерпретирован как строка тега ASCII.

*PoolType*
Указывает тип пула для поиска. Разрешены следующие значения:

0
Указывает непагаченный пул памяти. Это значение по умолчанию.

1
Указывает пул памяти с страницами.

2
Указывает специальный пул.

4
Указывает пул сеансов.

## DLL-библиотеки

# Дополнительная информация

Сведения о пулах памяти и тегах пула см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, по Марку Руссиновичу и Дэвиду Соломону.

# Замечания

Эта команда может занять значительное время для выполнения, в зависимости от размера памяти пула, который необходимо выполнить. Чтобы ускорить выполнение, увеличьте скорость COM-порта с **помощью клавиш CTRL+A (переключение скорости Baud)** или используйте **команду кэша (задать размер кэша)** для увеличения размера кэша (примерно до 10 МБ).

Тег пула является тем же тегом, который передается **в семейство подпрограмм ExAllocate***Xxx* .

Ниже приведен пример. Весь непагированный пул выполняется поиск, а затем выполняется поиск по страницам пула, но команда завершается до завершения (после часа операции):

dbgcmd

```
kd> !poolfind SeSd 0

Scanning large pool allocation table for Tag: SeSd (827d1000 : 827e9000)

Searching NonPaged pool (823b1000 : 82800000) for Tag: SeSd

826fa130 size:   c0 previous size:   40  (Allocated) SeSd
82712000 size:   c0 previous size:    0  (Allocated) SeSd
82715940 size:   a0 previous size:   60  (Allocated) SeSd
8271da30 size:   c0 previous size:   10  (Allocated) SeSd
82721c00 size:   10 previous size:   30  (Free)      SeSd
8272b3f0 size:   60 previous size:   30  (Allocated) SeSd
8272d770 size:   60 previous size:   40  (Allocated) SeSd
8272d7d0 size:   a0 previous size:   60  (Allocated) SeSd
8272d960 size:   a0 previous size:   70  (Allocated) SeSd
82736f30 size:   a0 previous size:   10  (Allocated) SeSd
82763840 size:   a0 previous size:   10  (Allocated) SeSd
8278b730 size:  100 previous size:  290  (Allocated) SeSd
8278b830 size:   10 previous size:  100  (Free)      SeSd
82790130 size:   a0 previous size:   20  (Allocated) SeSd
82799180 size:   a0 previous size:   10  (Allocated) SeSd
827c00e0 size:   a0 previous size:   30  (Allocated) SeSd
```

```
827c8320 size:   a0 previous size:   60  (Allocated) SeSd
827ca180 size:   a0 previous size:   50  (Allocated) SeSd
827ec140 size:   a0 previous size:   10  (Allocated) SeSd

Searching NonPaged pool (fe7c3000 : ffbe0000) for Tag: SeSd

kd> !poolfind SeSd 1

Scanning large pool allocation table for Tag: SeSd (827d1000 : 827e9000)

Searching Paged pool (e1000000 : e4400000) for Tag: SeSd

e10000b0 size:   d0 previous size:   20  (Allocated) SeSd
e1000260 size:   d0 previous size:   60  (Allocated) SeSd
......
e1221dc0 size:   a0 previous size:   60  (Allocated) SeSd
e1224250 size:   a0 previous size:   30  (Allocated) SeSd

...terminating - searched pool to e1224000
kd>
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !poolused

Статья • 05.04.2024

Расширение **!poolused** отображает сводки использования памяти на основе тега, используемого для каждого выделения пула.

dbgcmd

```
!poolused [Flags [TagString]]
```

## Параметры

*Flags*

Указывает объем отображаемых выходных данных и метод сортировки выходных данных. Это может быть любое сочетание следующих битовых значений, за исключением того, что биты 1 (0x2) и 2 (0x4) нельзя использовать вместе. По умолчанию используется 0x0, которая создает сводную информацию, отсортированную по тегу пула.

Bit 0 (0x1)

Отображает более подробные (подробные) сведения.

Бит 1 (0x2)

Сортирует отображение по объему непагрегированного использования памяти.

Бит 2 (0x4)

Сортирует отображение по объему использования страницы памяти.

Бит 3 (0x8)

Отображает пул сеансов вместо стандартного пула. Для переключения между сеансами можно использовать **команду !session** .

*TagString*

Указывает тег пула. *TagString* — это строка ASCII с учетом регистра. Звездочка (*) может использоваться для представления любого числа символов; Вопросительный знак (?) можно использовать для представления точно одного

символа. Если звездочка не используется, *TagString* должен иметь ровно четыре символа в длину.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о пулах памяти и тегах пула *см. в разделе* "Внутренние компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

## Замечания

Расширение **!poolused** собирает данные из функции тегов пула Windows. Теги пула постоянно включены в Windows Server 2003 и более поздних версиях Windows.

Если вы перестанете выполнять расширение до завершения, отладчик отображает частичные результаты.

В отображении этой команды показана память, используемая для каждого тега в пуле страниц и непагрегированном пуле. В обоих случаях отображение включает количество невыполненных выделений для данного тега и количество байтов, потребляемых этими выделениями.

Ниже приведен частичный пример выходных данных из этого расширения:

dbgcmd

```
0: kd> !poolused
   Sorting by  Tag

  Pool Used:
            NonPaged            Paged
 Tag    Allocs     Used    Allocs     Used
 1394        1      520         0        0UNKNOWN pooltag '1394', please
update pooltag.txt
 1MEM        1     3368         0        0UNKNOWN pooltag '1MEM', please
update pooltag.txt
 2MEM        1     3944         0        0UNKNOWN pooltag '2MEM', please
update pooltag.txt
 3MEM        3      248         0        0UNKNOWN pooltag '3MEM', please
update pooltag.txt
 8042        4     3944         0        0PS/2 kb and mouse , Binary:
i8042prt.sys
```

```
 AGP          1      344          2      384UNKNOWN pooltag 'AGP ', please
update pooltag.txt
 AcdN         2     1072          0        0TDI AcdObjectInfoG
 AcpA         3      192          1      504ACPI Pooltags , Binary: acpi.sys
 AcpB         0        0          4      576ACPI Pooltags , Binary: acpi.sys
 AcpD        40    13280          0        0ACPI Pooltags , Binary: acpi.sys
 AcpF         6      240          0        0ACPI Pooltags , Binary: acpi.sys
 AcpM         0        0          1      128ACPI Pooltags , Binary: acpi.sys
 AcpO         4      208          0        0ACPI Pooltags , Binary: acpi.sys

...

 WmiG        30     6960          0        0Allocation of WMIGUID
 WmiR        63     4032          0        0Wmi Registration info blocks
 Wmip       146     3504        182    18600Wmi General purpose allocation
 Wmit         1     4096          7    49480Wmi Trace
 Wrpa         2      720          0        0WAN_ADAPTER_TAG
 Wrpc         1       72          0        0WAN_CONN_TAG
 Wrpi         1      120          0        0WAN_INTERFACE_TAG
 Wrps         2      128          0        0WAN_STRING_TAG
 aEoP         1      672          0        0UNKNOWN pooltag 'aEoP', please
update pooltag.txt
 fEoP         1       16          0        0UNKNOWN pooltag 'fEoP', please
update pooltag.txt
 hSVD         0        0          1       40Shared Heap Tag , Binary:
mrxdav.sys
 hibr         0        0          1    24576UNKNOWN pooltag 'hibr', please
update pooltag.txt
 iEoP         1       24          0        0UNKNOWN pooltag 'iEoP', please
update pooltag.txt
 idle         2      208          0        0Power Manager idle handler
 jEoP         1       24          0        0UNKNOWN pooltag 'jEoP', please
update pooltag.txt
 mEoP         1       88          0        0UNKNOWN pooltag 'mEoP', please
update pooltag.txt
 ohci         1      136          0        01394 OHCI host controller driver
 rx..         3     1248          0        0UNKNOWN pooltag '  rx', please
update pooltag.txt
 sidg         2       48          0        0GDI spooler events
 thdd         0        0          1    20480DirectDraw/3D handle manager table
 usbp        18    77056          2       96UNKNOWN pooltag 'usbp', please
update pooltag.txt
 vPrt         0        0         18    68160UNKNOWN pooltag 'vPrt', please
update pooltag.txt
 TOTAL     3570214 209120008     38769 13066104
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !poolval

Статья • 05.04.2024

Расширение **!poolval** анализирует заголовки страницы пула и диагностировать любые возможные повреждения.

dbgcmd

```
!poolval Address [DisplayLevel]
```

# Параметры

*Адрес*
Указывает адрес пула, заголовок которого необходимо проанализировать.

*DisplayLevel*
Указывает сведения, которые необходимо включить в дисплей. Это может быть любое из следующих значений (значение по умолчанию равно нулю):

0
Приводит к отображению основных сведений.

1
Приводит к отображению основных сведений и связанных списков заголовков.

2
Приводит к отображению основных сведений, связанных списков заголовков и основных сведений о заголовках.

3
Приводит к отображению основных сведений, связанных списков заголовков и полных сведений о заголовках.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о пулах памяти см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, марку Руссиновичу и Дэвиду Соломону.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !popolicy

Статья • 05.04.2024

Расширение **!popolicy** отображает политику питания целевого компьютера.

dbgcmd

```
!popolicy [Address]
```

## Параметры

*Адрес*
Указывает адрес отображаемой структуры политики питания. Если это опущено, то nt! Отображается PopPolicy.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Чтобы просмотреть возможности питания системы, используйте **команду расширения !pocaps** . Сведения о возможностях питания и политике питания см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, марку Руссиновичу и Дэвиду Соломону.

## Замечания

Ниже приведен пример выходных данных этой команды:

dbgcmd

```
kd> !popolicy
SYSTEM_POWER_POLICY (R.1) @ 0x80164d58
  PowerButton:      Shutdown  Flags: 00000003   Event: 00000000   Query UI
  SleepButton:          None  Flags: 00000003   Event: 00000000   Query UI
  LidClose:             None  Flags: 00000001   Event: 00000000   Query
  Idle:                 None  Flags: 00000001   Event: 00000000   Query
  OverThrottled:        None  Flags: c0000004   Event: 00000000   Override
NoWakes Critical
  IdleTimeout:             0  IdleSensitivity:        50%
```

```
    MinSleep:               S0  MaxSleep:               S0
    LidOpenWake:            S0  FastSleep:              S0
    WinLogonFlags:           1  S4Timeout:               0
    VideoTimeout:            0  VideoDim:               209
    SpinTimeout:             0  OptForPower:             1
    FanTolerance:            0% ForcedThrottle:         0%
    MinThrottle:             0%
```

## См. также

команды самонастраивающийся и Power Debugger

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pplookaside

Статья • 05.04.2024

Команда **расширения !pplookaside отображает списки Lookaside** для процессоров на целевом компьютере.

dbgcmd

```
!pplookaside <address>
```

## Параметры

*ParamName*
Адрес процессора.

## DLL-библиотеки

kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ppmidle

Статья • 05.04.2024

Команда **расширения !ppmidle** отображает состояния простоя процессора.

```
dbgcmd
```

```
!ppmidle
```

## Параметры

нет

Эта команда поддерживается в следующих версиях Windows:

- Windows 7
- Windows 8
- Windows 8.1
- Windows 10 версии 1511
- Windows 10 версии 1607
- Windows 10 версии 1703

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ppmidleaccounting

Статья • 05.04.2024

Команда **расширения !ppmidleaccounting** отображает сведения о состоянии простоя процессора.

dbgcmd

```
!ppmidleaccounting
```

## Параметры

нет

Эта команда поддерживается в следующих версиях Windows:

- Windows 7
- Windows 8
- Windows 8.1
- Windows 10 версии 1511
- Windows 10 версии 1607
- Windows 10 версии 1703

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ppmidlepolicy

Статья • 05.04.2024

Команда **расширения !ppmidlepolicy** отображает политику состояния C.

dbgcmd

```
!ppmidlepolicy
```

## Параметры

нет

Эта команда поддерживается в следующих версиях Windows:

- Windows 7

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ppmlpscenarios

Статья • 05.04.2024

Команда **расширения !ppmlpscenarios** отображает переопределения политик для каждого сценария.

dbgcmd

```
!ppmlpscenarios
```

## Параметры

нет

Эта команда поддерживается в следующих версиях Windows:

- Windows 8

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ppmperf

Статья • 05.04.2024

Команда **расширения !ppmperf** отображает сведения о ограничении питания и производительности.

dbgcmd

```
!ppmperf
```

## Параметры

нет

## DLL-библиотеки

kedexts.dll

**ПРИМЕЧАНИЕ.** Эта команда не поддерживается во всех версиях Windows, таких как Windows 10, версия 1703.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ppmperfpolicy

Статья • 05.04.2024

Команда **расширения !ppmperfpolicy** больше не поддерживается, используйте вместо этого команду !ppmsettings .

dbgcmd

```
!ppmperfpolicy
```

## Параметры

нет

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ppmsettings

Статья • 05.04.2024

Команда **расширения !ppmsettings** отображает активные параметры ppm для процессора.

dbgcmd

```
!ppmsettings
```

## Параметры

нет

## Замечания

Эта команда поддерживается в следующих версиях Windows:

- Windows 10 версии 1511
- Windows 10 версии 1607
- Windows 10 версии 1703

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ppmstate

Статья • 05.04.2024

Команда **расширения !ppmstate** отображает сведения о состоянии питания.

dbgcmd

```
!ppmstate
```

## Параметры

нет

## DLL-библиотеки

Kdexts.dll

**ПРИМЕЧАНИЕ.** Эта команда не поддерживается во всех версиях Windows, таких как Windows 10, версия 1703.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !prcb

Статья • 05.04.2024

Расширение **!prcb** отображает блок управления процессором (PRCB).

dbgcmd

```
!prcb [Processor]
```

## Параметры

*Процессор*
Указывает процессор для получения сведений PRCB из. Если *обработчик* опущен, будет использоваться ноль процессора.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о PCR и PRCB см. в microsoft *Windows Internals*, Марк Руссинович и Дэвид Соломон.

## Замечания

PRCB — это расширение области управления процессором (PCR). Чтобы отобразить PCR, используйте **расширение !pcr** .

Рассмотрим пример:

dbgcmd

```
kd> !prcb
PRCB for Processor 0 at e0000000818ba000:
Threads--  Current e0000000818bbe10 Next 0000000000000000 Idle
e0000000818bbe10
Number 0 SetMember 00000001
Interrupt Count -- 0000b81f
```

```
Times -- Dpc    00000028 Interrupt 000003ff
         Kernel 00005ef4 User      00000385
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Процесс

Статья • 05.04.2024

Расширение !process отображает сведения о указанном процессе или обо всех процессах, включая блок EPROCESS.

Это расширение можно использовать только во время отладки в режиме ядра.

Синтаксис

dbgcmd

```
!process [/s Session] [/m Module] [Process [Flags]]
!process [/s Session] [/m Module] 0 Flags ImageName
```

# Параметры

**Сеанс /s** ****
Указывает сеанс, принадлежащий требуемому процессу.

**Модуль /m** ****
Указывает модуль, который владеет требуемым процессом.

*Обработать*
Указывает шестнадцатеричный адрес или идентификатор процесса процесса на целевом компьютере.

Значение *процесса* определяет, отображается ли расширение !process адрес или идентификатор процесса. Если *процесс* опущен в любой версии Windows, отладчик отображает данные только о текущем системном процессе. Если *параметр Process* is 0 и *ImageName* опущен, отладчик отображает сведения обо всех активных процессах. Если для сведений о текущем процессе отображается значение -1.

*Flags*
Указывает уровень детализации для отображения. *Флаги* могут быть любым сочетанием следующих битов. Если *флаги* равно 0, отображается только минимальное количество сведений. Значение по умолчанию зависит от версии Windows и значения *Process*. Значение по умолчанию 0x3, если *процесс опущен или если процесс* равен 0 или -1; в противном случае значение по умолчанию 0xF.

Bit 0 (0x1)
Отображает статистику времени и приоритета.

Бит 1 (0x2)
Отображает список потоков и событий, связанных с процессом, и их состояния ожидания.

Бит 2 (0x4)
Отображает список потоков, связанных с процессом. Если это включается без бита 1 (0x2), каждый поток отображается в одной строке. Если это включается вместе с битом 1, каждый поток отображается с трассировкой стека.

Бит 3 (0x8)
Отображает возвращаемый адрес и указатель стека для каждой функции. Отображение аргументов функции отключается.

Бит 4 (0x10)
Задает контекст процесса, равный указанному процессу в течение этой команды. Это приводит к более точному отображению стека потоков. Так как этот флаг эквивалентен использованию **.process /p /r** для указанного процесса, любой существующий список модулей пользовательского режима будет отключен карта. Если *процесс* равен нулю, отладчик отображает все процессы и контекст процесса изменяется для каждого из них. Если вы отображаете только один процесс и его состояние пользовательского режима уже обновлено (например, с **.process /p /r**), не обязательно использовать этот флаг. Этот флаг действует только при использовании с битом 0 (0x1).

*ImageName*
Указывает имя отображаемого процесса. Отладчик отображает все процессы, имена исполняемых изображений которых соответствуют *ImageName*. Имя изображения должно совпадать с именем в блоке EPROCESS. Как правило, это исполняемое имя, которое было вызвано для запуска процесса, включая расширение файла (обычно .exe), и усечено после пятнадцатого символа. Невозможно указать имя изображения, содержащее пробел. При *указании ImageName процесс* должен быть равен нулю.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о процессах в режиме ядра см. в разделе "Изменение контекстов". Дополнительные сведения об анализе процессов и потоков см. в разделе

"*Внутренние* компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

# Замечания

Ниже приведен пример **отображения !process 0 0** :

dbgcmd

```
kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
PROCESS 80a02a60  Cid: 0002    Peb: 00000000  ParentCid: 0000
    DirBase: 00006e05  ObjectTable: 80a03788  TableSize: 150.
    Image: System
PROCESS 80986f40  Cid: 0012    Peb: 7ffde000  ParentCid: 0002
    DirBase: 000bd605  ObjectTable: 8098fce8  TableSize:  38.
    Image: smss.exe
PROCESS 80958020  Cid: 001a    Peb: 7ffde000  ParentCid: 0012
    DirBase: 0008b205  ObjectTable: 809782a8  TableSize: 150.
    Image: csrss.exe
PROCESS 80955040  Cid: 0020    Peb: 7ffde000  ParentCid: 0012
    DirBase: 00112005  ObjectTable: 80955ce8  TableSize:  54.
    Image: winlogon.exe
PROCESS 8094fce0  Cid: 0026    Peb: 7ffde000  ParentCid: 0020
    DirBase: 00055005  ObjectTable: 80950cc8  TableSize: 222.
    Image: services.exe
PROCESS 8094c020  Cid: 0029    Peb: 7ffde000  ParentCid: 0020
    DirBase: 000c4605  ObjectTable: 80990fe8  TableSize: 110.
    Image: lsass.exe
PROCESS 809258e0  Cid: 0044    Peb: 7ffde000  ParentCid: 0026
    DirBase: 001e5405  ObjectTable: 80925c68  TableSize:  70.
    Image: SPOOLSS.EXE
```

В следующей таблице описаны некоторые элементы выходных **данных !process 0 0** .

Развернуть таблицу

| Элемент | Значение |
|---|---|
| Адрес процесса | Шестнадцатеричное число восьми символов после слова PROCESS — адрес блока EPROCESS. В последнем примере в предыдущем примере адрес процесса 0x809258E0. |
| Идентификатор процесса (PID) | Шестнадцатеричное число после слова Cid. В последнем примере в предыдущем примере идентификатор пин-кода 0x44 или десятичное значение 68. |

| Элемент | Значение |
| --- | --- |
| Блок среды обработки (PEB) | Шестнадцатеричное число после слова Peb — адрес блока среды процесса. В заключительной записи в предыдущем примере PEB находится по адресу 0x7FFDE000. |
| Родительский процесс PID | Шестнадцатеричное число после слова ParentCid является идентификатором идентификатора родительского процесса. В последнем примере в предыдущем примере родительский процесс PID 0x26 или десятичное значение 38. |
| Изображения | Имя модуля, которому принадлежит процесс. В последнем примере в предыдущем примере владелец spoolss.exe. В первой записи владелец является самой операционной системой. |
| Адрес объекта процесса | Шестнадцатеричное число после слова ObjectTable. В последнем примере в предыдущем примере адрес объекта процесса 0x80925c68. |

Чтобы отобразить полные сведения об одном процессе, задайте *для флагов* значение 7. Сам процесс можно указать, установив "*Процесс* " равным адресу процесса, параметру *Process* равно идентификатору процесса или параметру *ImageName* равным имени исполняемого образа. Рассмотрим пример:

dbgcmd

```
kd> !process fb667a00 7
PROCESS fb667a00 Cid: 0002  Peb: 00000000 ParentCid: 0000
  DirBase: 00030000 ObjectTable: e1000f88 TableSize: 112.
  Image: System
  VadRoot fb666388 Clone 0 Private 4. Modified 9850. Locked 0.
  FB667BBC MutantState Signalled OwningThread 0
  Token                e10008f0
  ElapsedTime            15:06:36.0338
  UserTime             0:00:00.0000
  KernelTime            0:00:54.0818
  QuotaPoolUsage[PagedPool]     1480
Working Set Sizes (now,min,max) (3, 50, 345)
  PeakWorkingSetSize        118
  VirtualSize            1 Mb
  PeakVirtualSize          1 Mb
  PageFaultCount          992
  MemoryPriority          BACKGROUND
```

```
  BasePriority           8
  CommitCharge           8

    THREAD fb667780 Cid 2.1 Teb: 00000000 Win32Thread: 80144900 WAIT:
(WrFreePage) KernelMode Non-Alertable
    80144fc0 SynchronizationEvent
    Not impersonating
    Owning Process fb667a00
    WaitTime (seconds)   32278
    Context Switch Count  787
    UserTime         0:00:00.0000
    KernelTime        0:00:21.0821
    Start Address Phase1Initialization (0x801aab44)
    Initial Sp fb26f000 Current Sp fb26ed00
    Priority 0 BasePriority 0 PriorityDecrement 0 DecrementCount 0

    ChildEBP RetAddr Args to Child
    fb26ed18 80118efc c0502000 804044b0 00000000 KiSwapThread+0xb5
    fb26ed3c 801289d9 80144fc0 00000008 00000000 KeWaitForSingleObject+0x1c2
```

Обратите внимание, что адрес объекта процесса можно использовать в качестве входных данных для других расширений, таких как **!handle**, для получения дополнительных сведений.

В следующей таблице описаны некоторые элементы в предыдущем примере.

Развернуть таблицу

| Элемент | Значение |
|---|---|
| ПОДОЖДИ | Скобка после этого заголовка дает причину ожидания. **Команда dt nt!_KWAIT_REASON** отобразит список всех причин ожидания. |
| ElapsedTime | Выводит список времени, истекшего с момента создания процесса. Это отображается в единицах часов:Minutes:Seconds.Milliseconds. |
| UserTime | Выводит список времени выполнения процесса в пользовательском режиме. Если значение UserTime является исключительно высоким, он может определить процесс, который истощает системные ресурсы. Единицы совпадают с единицами elapsedTime. |
| KernelTime | Выводит список времени выполнения процесса в режиме ядра. Если значение для KernelTime является исключительно |

| Элемент | Значение |
| --- | --- |
| | высоким, он может определить процесс, который истощает системные ресурсы. Единицы совпадают с единицами elapsedTime. |
| Размеры рабочих наборов | Перечисляет текущий, минимальный и максимальный размер рабочего набора для процесса на страницах. Исключительно большой размер рабочего набора может быть признаком процесса утечки памяти или истощения системных ресурсов. |
| Записи QuotaPoolUsage | Выводит список страничного и непагированного пула, используемого процессом. В системе с утечкой памяти поиск чрезмерного использования непагрегированного пула во всех процессах можно определить, какой процесс имеет утечку памяти. |
| Клонировать | Указывает, был ли процесс создан подсистемами POSIX или Interix. |
| Private | Указывает количество частных (не совместно используемых) страниц, используемых в данный момент процессом. Это включает в себя как страницы, так и выстраивает память. |

Помимо сведений о списке процессов, сведения о потоке содержат список ресурсов, на которых блокируются потоки. Эти сведения перечислены в третьей строке выходных данных после заголовка потока. В этом примере поток имеет блокировку для одного ресурса, синхронизацииEvent с адресом 80144fc0. Сравнивая этот адрес со списком блокировок, отображаемых расширением **!kdext*.locks** , можно определить, какие потоки имеют монопольные блокировки для ресурсов.

Расширение **!stacks** содержит краткую сводку о состоянии каждого потока. Это можно использовать вместо расширения !process, чтобы получить краткий обзор системы, особенно при отладке многопоточных проблем, таких как конфликты ресурсов или взаимоблокировки.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !processfields

Статья • 05.04.2024

Расширение **!processfields** отображает имена и смещения полей в блоке исполнительного процесса (EPROCESS).

```
dbgcmd

!processfields
```

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Kdextx86.dll |
| **Windows XP и более поздние версии** | Недоступно (см. раздел "Примечания") |

## Дополнительная информация

Сведения о блоке EPROCESS см. в разделе "*Внутренние* компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

## Замечания

Эта команда расширения недоступна в Windows XP или более поздних версиях Windows. Вместо этого используйте **команду dt (тип отображения)** для отображения структуры EPROCESS напрямую:

```
dbgcmd

kd> dt nt!_EPROCESS
```

Ниже приведен пример **!processfields** из системы Windows 2000:

```
dbgcmd

kd> !processfields
 EPROCESS structure offsets:

    Pcb:                                0x0
```

```
    ExitStatus:                     0x6c
    LockEvent:                      0x70
    LockCount:                      0x80
    CreateTime:                     0x88
    ExitTime:                       0x90
    LockOwner:                      0x98
    UniqueProcessId:                0x9c
    ActiveProcessLinks:             0xa0
    QuotaPeakPoolUsage[0]:          0xa8
    QuotaPoolUsage[0]:              0xb0
    PagefileUsage:                  0xb8
    CommitCharge:                   0xbc
    PeakPagefileUsage:              0xc0
    PeakVirtualSize:                0xc4
    VirtualSize:                    0xc8
    Vm:                             0xd0
    DebugPort:                      0x120
    ExceptionPort:                  0x124
    ObjectTable:                    0x128
    Token:                          0x12c
    WorkingSetLock:                 0x130
    WorkingSetPage:                 0x150
    ProcessOutswapEnabled:          0x154
    ProcessOutswapped:              0x155
    AddressSpaceInitialized:        0x156
    AddressSpaceDeleted:            0x157
    AddressCreationLock:            0x158
    ForkInProgress:                 0x17c
    VmOperation:                    0x180
    VmOperationEvent:               0x184
    PageDirectoryPte:               0x1f0
    LastFaultCount:                 0x18c
    VadRoot:                        0x194
    VadHint:                        0x198
    CloneRoot:                      0x19c
    NumberOfPrivatePages:           0x1a0
    NumberOfLockedPages:            0x1a4
    ForkWasSuccessful:              0x182
    ExitProcessCalled:              0x1aa
    CreateProcessReported:          0x1ab
    SectionHandle:                  0x1ac
    Peb:                            0x1b0
    SectionBaseAddress:             0x1b4
    QuotaBlock:                     0x1b8
    LastThreadExitStatus:           0x1bc
    WorkingSetWatch:                0x1c0
    InheritedFromUniqueProcessId:   0x1c8
    GrantedAccess:                  0x1cc
    DefaultHardErrorProcessing      0x1d0
    LdtInformation:                 0x1d4
    VadFreeHint:                    0x1d8
    VdmObjects:                     0x1dc
    DeviceMap:                      0x1e0
    ImageFileName[0]:               0x1fc
    VmTrimFaultValue:               0x20c
```

```
    Win32Process:                   0x214
    Win32WindowStation:             0x1c4
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !processirps

Статья • 05.04.2024

Расширение **!processirps** отображает сведения о пакетах запросов ввода-вывода (IRPs), связанных с процессами.

dbgcmd

```
!processirps
!processirps ProcessAddress [Flags]
```

## Параметры

**** *ProcessAddress*
Адрес процесса. При указании *ProcessAddress* отображаются только irPs, связанные с этим процессом. Если не указать *ProcessAddress*, отображаются irPs для всех процессов.

**** *Flags*
Побитовое ИЛИ одного или нескольких следующих флагов.

Bit 0 (0x1)
Отображение irPs в очереди в потоки.

Бит 1 (0x2)
Отображение irPs в очереди для объектов файлов.

При указании *флагов* необходимо также указать *ProcessAddress*. Если флаги не указаны , *отображаются ip-адреса* irPs, в очереди как для потоков, так и для файловых объектов.

## DLL-библиотеки

kdexts.dll

## Замечания

Эта команда позволяет быстро определить все очереди IRP для процесса, как те, которые помещаются в потоки, так и те, которые помещаются в очередь для

файловых объектов. IrPs помещаются в очередь в объект файла, если объект файла имеет порт завершения, связанный с ним.

# Примеры

Для получения адресов процесса можно использовать **команду !process** . Например, можно получить адрес процесса для explorer.exe.

dbgcmd

```
2: kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
...
PROCESS fffffa800688c940
    SessionId: 1  Cid: 0bbc    Peb: 7f70da5e000  ParentCid: 0b84
    DirBase: 2db10000  ObjectTable: fffff8a0025bd440  HandleCount: 1056.
    Image: explorer.exe
```

Теперь можно передать адрес процесса для explorer.exe команде **!processirps** . В следующих выходных данных показано, что explorer.exe имеет очереди irPs в потоки и irPs в очереди в объекты файлов.

dbgcmd

```
2: kd> !processirps fffffa800688c940
**** PROCESS fffffa800688c940 (Image: explorer.exe) ****

Checking threads for IRPs.

  Thread fffffa800689f080:

    IRP fffffa80045ccc10 - Owned by \FileSystem\Ntfs for device
fffffa8004f5c030
    IRP fffffa800454f650 - Owned by \FileSystem\Ntfs for device
fffffa8004f5c030
    ...
    IRP fffffa80068e9c10 - Owned by \FileSystem\Ntfs for device
fffffa8004f5c030

Checking file objects for IRPs.

  FileObject fffffa80068795e0 (handle 8bc):

    IRP fffffa8006590cf0 - Owned by \Driver\DeviceApi for device DeviceApi
(fffffa800363ae40)

  ...

  FileObject fffffa8005bf59c0 (handle 900):
```

```
    IRP fffffa8006659010 - Owned by \Driver\DeviceApi for device DeviceApi
(fffffa800363ae40)
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Psp

Статья • 05.04.2024

Расширение **!psp отображает регистр состояния процессора (PSP** ) по указанному адресу.

Это расширение поддерживается только на целевых компьютерах на основе Itanium.

dbgcmd

```
!psp Address [DisplayLevel]
```

**Важно, что** эта команда не рекомендуется использовать в отладчике Windows версии 10.0.14257 и более поздних версий и больше не доступна.

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес регистра PSP для отображения.

*DisplayLevel*
Может быть одним из следующих вариантов:

**0**
Отображает только значения каждого поля PSP. Это значение по умолчанию.

**1**
Отображает более подробные сведения о каждом из полей PSP, которые не зарезервированы или игнорируются.

**2**
Отображает более подробные сведения обо всех полях PSP, включая те, которые игнорируются или зарезервированы.

## DLL-библиотеки

Kdexts.dll

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Pte

Статья • 05.04.2024

Расширение **!pte** отображает запись таблицы страницы (PTE) и запись каталога страницы (PDE) для указанного адреса.

Синтаксис

dbgcmd

```
!pte VirtualAddress
!pte PTE
!pte LiteralAddress 1
```

## Параметры

*VirtualAddress*
Указывает виртуальный адрес, таблица страниц которого является требуемой.

*PTE*
Указывает адрес фактического PTE.

*ЛитералAddress* **** **1**
Указывает адрес фактического PTE или PDE.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о таблицах страниц, каталогах страниц и объяснении битов состояния см. в разделе "*Внутренние* элементы Microsoft Windows", Марк Руссинович и Дэвид Соломон.

## Замечания

Если указан один параметр и этот параметр является адресом в области памяти, в которой хранятся таблицы страниц, отладчик обрабатывает это как *параметр PTE* .

Этот параметр принимается в качестве фактического адреса требуемого PTE, а отладчик будет отображать этот PTE и соответствующий PDE.

Если указан один параметр и этот параметр не является адресом в этом регионе, отладчик обрабатывает это как *параметр VirtualAddress* . Отображаются PTE и PDE, которые содержат сопоставление для этого адреса.

Если заданы два параметра, а второй параметр равен **1** (или другому небольшому числу), отладчик обрабатывает первый параметр как *LiteralAddress*. Этот адрес интерпретируется как фактический адрес PTE, а также как фактический адрес PDE, а соответствующие (и, возможно, недопустимые) данные будут отображаться.

(только на целевом компьютере x86 или x64) Если заданы два параметра, а второй параметр больше первого, отладчик обрабатывает два параметра как StartAddress и *EndAddress*. Затем команда отображает PTEs для каждой страницы в указанном диапазоне памяти.

Для списка всех системных PTEs используйте **расширение !sysptes** .

Ниже приведен пример на целевом компьютере x86:

dbgcmd

```
kd> !pte 801544f4
801544F4  - PDE at C0300800        PTE at C0200550
          contains 0003B163      contains 00154121
        pfn 3b G-DA--KWV    pfn 154 G--A--KRV
```

В первой строке этого примера выполняется переоследоваемый виртуальный адрес. Затем он предоставляет виртуальный адрес PDE и PTE, которые содержат сведения о виртуальном физическом сопоставлении этого адреса.

Вторая строка предоставляет фактическое содержимое PDE и PTE.

Третья строка принимает эти содержимое и анализирует их, разбивая их на номер кадра страницы (PFN) и биты состояния.

Сведения о интерпретации и использовании PFN см. в **разделе "Расширение !pfn**" или "Преобразование виртуальных адресов в физические адреса".

На целевом компьютере x86 или x64 биты состояния для PDE и PTE отображаются в следующей таблице. Отображение **!pte** указывает эти биты с прописными буквами или дефисами, а также добавляет дополнительные сведения.

**Развернуть таблицу**

| бит | Отображение при установке | Отображение при очистке | Значение |
|---|---|---|---|
| 0x200 | O | - | Скопируйте запись. |
| 0x100 | G | - | Глобальный. |
| 0x80 | L | - | Большая страница. Это происходит только в PDEs, никогда не в PTEs. |
| 0x40 | D | - | Грязные. |
| 0x20 | a | - | Доступ. |
| 0x10 | N | - | Кэш отключен. |
| 0x8 | T | - | Запись через. |
| 0x4 | U | тыс. | Владелец (режим пользователя или режим ядра). |
| 0x2 | Ср. | R | Доступная для записи или доступная только для чтения. Только на компьютерах с несколькими процессорами и на любом компьютере под управлением Windows Vista или более поздней версии. |
| 0x1 | V | | Действительны. |
| | E | - | Страница исполняемого файла. Для платформ, которые не поддерживают аппаратное выполнение/noexecute bit, включая многие системы x86, всегда отображается E. |

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !pte2va

Статья • 05.04.2024

Расширение **!pte2va** отображает виртуальный адрес, соответствующий указанной записи таблицы страницы (PTE).

```
dbgcmd
```

```
!pte2va Address
```

## Параметры

*Адрес*
Указывает PTE.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о таблицах *страниц и PTEs см. в разделе* "Внутренние компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

## Замечания

Чтобы проверить содержимое определенного PTE, используйте **расширение !pte** .

Ниже приведен пример выходных данных расширения **!pte2va** :

```
dbgcmd
```

```
kd> !pte2va 9230
000800000248c000
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ptov

Статья • 05.04.2024

Расширение **!ptov** отображает всю физическую и виртуальную карту для данного процесса.

```
dbgcmd

!ptov DirBase
```

## Параметры

*DirBase*
Указывает базу каталогов для процесса. Чтобы определить базу каталогов, используйте **команду !process** и просмотрите значение, отображаемое для DirBase.

## DLL-библиотеки

Kdexts.dll

## Замечания

Ниже приведен пример. Во-первых, используйте процесс .process и **!process, чтобы определить базу каталогов текущего процесса:**

```
dbgcmd

1: kd> .process
Implicit process is now 852b4040
1: kd> !process 852b4040 1
PROCESS 852b4040  SessionId: none  Cid: 0004    Peb: 00000000  ParentCid:
0000
    DirBase: 00185000  ObjectTable: 83203000  HandleCount: 663.
    Image: System
    ...
```

В этом случае база каталогов 0x00185000. Передайте этот адрес **в !ptov:**

```
dbgcmd

1: kd> !ptov 185000
X86PtoV: pagedir 185000, PAE enabled.
```

```
15e11000 10000
549e6000 20000
...
60a000 210000
40b000 211000
...
54ad3000 25f000
548d3000 260000
...
d71000 77510000
...
```

Числа в левом столбце — это физические адреса каждой страницы памяти, которая имеет сопоставление для этого процесса. Числа в правом столбце — это виртуальные адреса, с которыми они сопоставляются.

Общий дисплей очень длинный.

Ниже приведен 64-разрядный пример.

dbgcmd

```
3: kd> .process
Implicit process is now fffffa80`0361eb30
3: kd> !process fffffa80`0361eb30 1
PROCESS fffffa800361eb30
    SessionId: none  Cid: 0004    Peb: 00000000  ParentCid: 0000
    DirBase: 00187000  ObjectTable: fffff8a000002870  HandleCount: 919.
    Image: System
    ...
3: kd> !ptov 187000
Amd64PtoV: pagedir 187000
00000001`034fb000 1d0000
a757c000 1d1000
00000001`0103d000 1d2000
c041e000 1d3000
...
2ed6f000 fffff680`00001000
00000001`13939000 fffff680`00003000
ceefb000 fffff680`00008000
...
```

База каталогов — это физический адрес первой таблицы, которая используется в переводе виртуальных адресов. Эта таблица имеет разные имена в зависимости от разрядности целевой операционной системы и включения расширения физического адреса (PAE) для целевой операционной системы.

Для 64-разрядной версии Windows база каталогов — это физический адрес таблицы уровня карты страницы 4 (PML4). Для 32-разрядной версии Windows с

поддержкой PAE база каталогов — это физический адрес таблицы указателей каталога страниц (PDP). Для 32-разрядной версии Windows с отключенным paE база каталогов — это физический адрес таблицы page Directory (PD).

Дополнительные сведения см. в статье **!vtop** и преобразование виртуальных адресов в физические адреса. Сведения о переводе виртуальных *адресов см. в разделе* "Внутренние компоненты Microsoft Windows", Марк Руссинович и Дэвид Соломон.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !qlocks

Статья • 05.04.2024

Расширение **!qlocks** отображает состояние всех блокировок спина в очереди.

dbgcmd

```
!qlocks
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о спин-блокировках см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, Марку Руссиновичу и Дэвиду Соломону.

## Замечания

Эта команда полезна только в многопроцессорной системе.

Рассмотрим пример:

dbgcmd

```
0: kd> !qlocks
Key: O = Owner, 1-n = Wait order, blank = not owned/waiting, C = Corrupt

                         Processor Number
    Lock Name         0  1  2  3

KE   - Dispatcher
KE   - Unused Spare
MM   - PFN
MM   - System Space
CC   - Vacb
CC   - Master
EX   - NonPagedPool
IO   - Cancel
EX   - WorkQueue
IO   - Vpb
IO   - Database
```

```
IO   - Completion
NTFS - Struct
AFD  - WorkQueue
CC   - Bcb
MM   - MM NonPagedPool
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Готов

Статья • 05.04.2024

Расширение **!ready** отображает сводные сведения о каждом потоке в системе в состоянии READY.

dbgcmd

```
!ready [Flags]
```

## Параметры

*Flags*
Указывает уровень детализации для отображения. *Флаги* могут быть любым сочетанием следующих битов. Если *флаги* равно 0, отображается только минимальное количество сведений. Значение по умолчанию — 0x6.

Бит 1 (0x2)
Приводит к включению состояния ожидания потока.

Бит 2 (0x4)
Если это включается без бита 1 (0x2), это не влияет. Если это включается вместе с битом 1, поток отображается с трассировкой стека.

Бит 3 (0x8)
Приводит к тому, что отображение каждой функции включает возвращаемый адрес и указатель стека. Отображение аргументов функции подавляется.

Бит 4 (0x10)
Приводит к тому, что отображение каждой функции включает только возвращаемый адрес; аргументы и указатели стека подавляются.

## DLL-библиотеки

Kdexts.dll

## Замечания

Выходные данные этого расширения похожи на **!thread**, за исключением того, что отображаются только готовые потоки, и они отсортированы в порядке

уменьшения приоритета.

## Дополнительная информация

Сведения о планировании потоков и состоянии READY см. в разделе "*Внутренние* компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Reg

Статья • 05.04.2024

Расширение **!reg** отображает и выполняет поиск по данным реестра.

dbgcmd

```
!reg {querykey|q} FullKeyPath
!reg keyinfo HiveAddress KeyNodeAddress
!reg kcb Address
!reg knode Address
!reg kbody Address
!reg kvalue Address
!reg valuelist HiveAddress KeyNodeAddress
!reg subkeylist HiveAddress KeyNodeAddress
!reg baseblock HiveAddress
!reg seccache HiveAddress
!reg hashindex [HiveAddress]HashKey
!reg openkeys {HiveAddress|0}
!reg openhandles {HiveAddress|0}
!reg findkcb FullKeyPath
!reg hivelist
!reg viewlist HiveAddress
!reg freebins HiveAddress
!reg freecells BinAddress
!reg dirtyvector HiveAddress
!reg cellindex HiveAddress Index
!reg freehints HiveAddress Storage Display
!reg translist {RmAddress|0}
!reg uowlist TransactionAddress
!reg locktable KcbAddress ThreadAddress
!reg convkey KeyPath
!reg postblocklist
!reg notifylist
!reg ixlock LockAddress
!reg dumppool [s|r]
```

# Параметры

{**querykey**|**q**} *** *FullKeyPath*
Отображает вложенные ключи и значения ключа, если ключ кэшируется.
*FullKeyPath* указывает полный путь к ключу.

**keyinfo***HiveAddress KeyNodeAddress* ****
Отображает вложенные ключи и значения узла ключа. *HiveAddress* указывает адрес куста. *KeyNodeAddress* указывает адрес узла ключа.

**Адрес kcb** ****
Отображает блок управления разделом реестра. *Адрес* указывает адрес блока управления ключом.

**knode** **** *Address*
Отображает структуру узла раздела реестра. *Адрес* указывает адрес узла ключа.

**адрес kbody** ****
Отображает структуру текста раздела реестра. *Адрес* указывает адрес основного текста. (Тела разделов реестра являются фактическими объектами, связанными с дескрипторами.)

**Адрес kvalue** ****
Отображает структуру значений раздела реестра. *Адрес* указывает адрес значения.

**valuelist** **** *HiveAddress KeyNodeAddress* ****
Отображает список значений в указанном узле ключа. *HiveAddress* указывает адрес куста. *KeyNodeAddress* указывает адрес узла ключа.

**subkeylist** **** *HiveAddress KeyNodeAddress* ****
Отображает список вложенных ключей указанного узла ключа. *HiveAddress* указывает адрес куста. *KeyNodeAddress* указывает адрес узла ключа.

**baseblock** **** *HiveAddress*
Отображает базовый блок для hive (также известный *как заголовок* hive). *HiveAddress* указывает адрес куста.

**seccache** **** *HiveAddress*
Отображает кэш безопасности для hive. *HiveAddress* указывает адрес куста.

**hashindex** *** [*HiveAddress*] **** *HashKey*
Вычисляет запись хэш-индекса для хэш-ключа. *HiveAddress* указывает адрес куста. *HashKey* задает ключ.

**Примечание***HiveAddress* является обязательным, если целевой компьютер работает под управлением Windows 7 или более поздней версии.

**openkeys** {*HiveAddress*|**0}**
Отображает все открытые ключи в кусте. *HiveAddress* указывает адрес куста. Если вместо этого используется ноль, отображается вся хэш-таблица реестра; Эта таблица содержит все открытые разделы в реестре.

**findkcb** **** *FullKeyPath*
Отображает блок управления разделом реестра, соответствующий пути реестра.

*FullKeyPath* указывает полный путь к ключу. Этот путь должен присутствовать в хэш-таблице.

**hivelist**
Отображает список всех кустов в системе, а также подробные сведения о каждом кусте.

**viewlist** **** *HiveAddress*
Отображает все закрепленные и сопоставленные представления для куста с подробными сведениями для каждого представления. *HiveAddress* указывает адрес куста.

**freebins HiveAddress** ****
Отображает все бесплатные ячейки для куста с подробными сведениями для каждого контейнера. *HiveAddress* указывает адрес куста.

**freecells BinAddress** ****
Выполняет итерацию по ячейке и отображает все свободные ячейки внутри него. *BinAddress* указывает адрес ячейки.

**грязное vector** **** *HiveAddress*
Отображает вектор грязное для куста. *HiveAddress* указывает адрес куста.

**Index cellindex HiveAddress** **** ****
Отображает виртуальный адрес ячейки в кусте. *HiveAddress* указывает адрес куста. *Индекс* задает индекс ячейки.

**freehints***HiveAddress* **** *служба хранилища* **** *Display*
Отображает сведения о бесплатном указании.

**translist** {*RmAddress*|**0}**
Отображает список активных транзакций в RM. *RmAddress* указывает адрес RM.

**uowlist***TransactionAddress*
Отображает список UoW, подключенных к транзакции. *TransactionAddress* указывает адрес транзакции.

**locktable***KcbAddress ThreadAddress*
Отображает соответствующее содержимое таблицы блокировки.

**convkey***KeyPath*
Отображает хэш-ключи для пути к ключу.

**postblocklist**
Отображает список потоков, на которых размещены postblocks.

**список уведомлений**
Отображает список блоков уведомлений в системе.

**ixlock***LockAddress*
Отображает владение блокировкой намерения. *LockAddress* указывает адрес блокировки.

**dumppool** [**s**|**r**]
Отображает выделенный реестром пул страниц. Если **задано значение s** , список страниц реестра сохраняется во временном файле. Если **задано значение r** , список страниц реестра восстанавливается из ранее сохраненного временного файла.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о реестре и его компонентах *см. в разделе* "Внутренние компоненты Microsoft Windows" марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Ниже приведен пример. Сначала используйте **!reg hivelist** , чтобы получить список адресов hive.

dbgcmd

```
00: kd> !reg hivelist
##

## |     HiveAddr     |Stable Length|    Stable Map    |Volatile Length|
Volatile Map    |MappedViews|PinnedViews|U(Cnt)|     BaseBlock     |
FileName

| fffff8a000014010 |       1000  | fffff8a0000140b0 |       1000    |
fffff8a000014328  |     0| fffff8a00001e000  | <NONAME>
| fffff8a000028010 |     a15000  | fffff8a00002e000 |      1a000    |
fffff8a000028328  |     0| fffff8a000029000  | SYSTEM
| fffff8a00004f010 |      14000  | fffff8a00004f0b0 |       c000    |
fffff8a00004f328  |     0| fffff8a000050000  | <NONAME>
| fffff8a000329010 |       6000  | fffff8a0003290b0 |          0    |
0000000000000000  |     0| fffff8a00032f000  |
Device\HarddiskVolume1\Boot\BCD
| fffff8a0002f2010 |    4255000  | fffff8a0006fa000 |       6000    |
```

```
fffff8a0002f2328  |     0| fffff8a00036c000  |
emRoot\System32\Config\SOFTWARE
| fffff8a000df0010 |      f7000  | fffff8a000df00b0 |       1000    |
fffff8a000df0328  |     0| fffff8a000df1000  |
temRoot\System32\Config\DEFAULT
| fffff8a0010f8010 |       9000  | fffff8a0010f80b0 |       1000    |
fffff8a0010f8328  |     0| fffff8a0010f9000  |
emRoot\System32\Config\SECURITY
| fffff8a001158010 |       7000  | fffff8a0011580b0 |          0    |
0000000000000000  |     0| fffff8a001159000  |
\SystemRoot\System32\Config\SAM
| fffff8a00124b010 |      24000  | fffff8a00124b0b0 |          0    |
0000000000000000  |     0| fffff8a00124c000  |
files\NetworkService\NTUSER.DAT
| fffff8a0012df220 |      b7000  | fffff8a0012df2c0 |          0    |
0000000000000000  |     0| fffff8a0012e6000  |
\SystemRoot\System32\Config\BBI
| fffff8a001312220 |      26000  | fffff8a0013122c0 |          0    |
0000000000000000  |     0| fffff8a00117e000  |
rofiles\LocalService\NTUSER.DAT
| fffff8a001928010 |      64000  | fffff8a0019280b0 |       3000    |
fffff8a001928328  |     0| fffff8a00192b000  |
User.MYTESTCOMPUTER2\ntuser.dat
| fffff8a001b9b010 |     203000  | fffff8a001bc4000 |          0    |
0000000000000000  |     0| fffff8a001b9c000  |
\Microsoft\Windows\UsrClass.dat
| fffff8a001dc0010 |      30000  | fffff8a001dc00b0 |          0    |
0000000000000000  |     0| fffff8a001dc2000  | Volume
Information\Syscache.hve
## | fffff8a0022dc010 |     175000  | fffff8a0022dc0b0 |          0    |
0000000000000000  |     0| fffff8a0022dd000  |
\AppCompat\Programs\Amcache.hve
```

Используйте третий адрес hive в предыдущих выходных данных (fff8a00004f010) в качестве аргумента **для !reg openkeys**.

dbgcmd

```
0: kd> !reg openkeys fffff8a00004f010

# Hive: \REGISTRY\MACHINE\HARDWARE

Index e9:    3069276d kcb=fffff8a00007eb98 cell=00000220 f=00200000
\REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM
Index 101:   292eea1f kcb=fffff8a00007ecc0 cell=000003b8 f=00200000
\REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM\MULTIFUNCTIONADAPTER
Index 140:   d927b0d4 kcb=fffff8a00007ea70 cell=000001a8 f=00200000
\REGISTRY\MACHINE\HARDWARE\DESCRIPTION
Index 160:   96d26a30 kcb=fffff8a00007e6f8 cell=00000020 f=002c0000
\REGISTRY\MACHINE\HARDWARE

# 0x4 keys found
```

Используйте первый полный путь к ключу в предыдущих выходных данных (\REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM) в качестве аргумента в **!reg querykey**.

dbgcmd

```
0: kd> !reg querykey \REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM

Found KCB = fffff8a00007eb98 ::
\REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM

Hive         fffff8a00004f010
KeyNode      fffff8a000054224

[SubKeyAddr]         [SubKeyName]
fffff8a000060244     CentralProcessor
fffff8a00006042c     FloatingPointProcessor
fffff8a0000543bc     MultifunctionAdapter

[SubKeyAddr]         [VolatileSubKeyName]
fffff8a000338d8c     BIOS
fffff8a0002a2e4c     VideoAdapterBusses

 Use '!reg keyinfo fffff8a00004f010 <SubKeyAddr>' to dump the subkey details

[ValueType]         [ValueName]                    [ValueData]
REG_BINARY          Component Information          0x542AC - 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00
REG_SZ              Identifier                     AT/AT COMPATIBLE
REG_FULL_RESOURCE_DESCRIPTORConfiguration Data            ff ff ff ff ff ff
ff ff 00 00 00 00 02 00 00 00 05 00 00 00 18 00 00 00 00 00 00 00 00 00
00 00 00 00 00 80 00 ff 03 00 00 3f 00 fe 00 02 00 81 00 fe 03 00 00 3f 00
fe 00 02 00 05 00 00 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 0c 00 00 00 04 00
REG_SZ              SystemBiosDate                 07/18/07
REG_MULTI_SZ        SystemBiosVersion              HPQOEM - 20070718\0\0
REG_SZ              VideoBiosDate                  03/23/20
REG_MULTI_SZ        VideoBiosVersion               Hardware Version 0.0\0\0
```

Вот еще один пример:

dbgcmd

```
kd> !reg hivelist
##

## | HiveAddr |Stable Length|Stable Map|Volatile Length|Volatile
Map|MappedViews|PinnedViews|U(Cnt)| BaseBlock | FileName

| e16e7428 |       2000  | e16e7484 |          0    |  00000000  |        1
|        0  |     0| e101f000  | \Microsoft\Windows\UsrClass.dat
| e1705a78 |      77000  | e1705ad4 |       1000    |  e1705bb0  |       30
```

```
|        0  |     0| e101c000  | ttings\Administrator\ntuser.dat
| e13d4b88 |     814000  | e146a000 |       1000    |  e13d4cc0  |      255
|        0  |     0| e1460000  | emRoot\System32\Config\SOFTWARE
| e13ad008 |      23000  | e13ad064 |       1000    |  e13ad140  |        9
|        0  |     0| e145e000  | temRoot\System32\Config\DEFAULT
| e13b3b88 |       a000  | e13b3be4 |       1000    |  e13b3cc0  |        3
|        0  |     0| e145d000  | emRoot\System32\Config\SECURITY
| e142d008 |       5000  | e142d064 |          0    |  00000000  |        2
|        0  |     0| e145f000  | <UNKNOWN>
| e11e3628 |       4000  | e11e3684 |       3000    |  e11e3760  |        0
|        0  |     0| e11e4000  | <NONAME>
| e10168a8 |     1c1000  | e1016904 |      15000    |  e10169e0  |       66
|        0  |     0| e1017000  | SYSTEM
## | e10072c8 |       1000  | e1007324 |          0    |  00000000  |
0  |        0  |     0| e1010000  | <NONAME>


kd> !reg hashindex e16e7428

CmpCacheTable = e100a000

Hash Index[e16e7428] : 5ac
Hash Entry[e16e7428] : e100b6b0

kd> !reg openkeys e16e7428

Index 68:  7bab7683 kcb=e13314f8 cell=00000740 f=00200004 \REGISTRY\USER\S-
1-5-21-1715567821-413027322-527237240-500_Classes\CLSID
Index 7a1:  48a30288 kcb=e13a3738 cell=00000020 f=002c0004 \REGISTRY\USER\S-
1-5-21-1715567821-413027322-527237240-500_Classes
```

Чтобы отобразить отформатированные сведения о разделе реестра, используйте **вместо этого расширение !dreg** .

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !regkcb

Статья • 05.04.2024

Расширение **!regkcb** отображает блок управления разделом реестра.

```
dbgcmd

!regkcb Address
```

## Параметры

*Адрес*
Задает адрес блока управления ключом.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Kdextx86.dll |
| **Windows XP и более поздние версии** | Рекомендации недоступны |

## Дополнительная информация

Дополнительные сведения о реестре и его компонентах *см. в разделе* "Внутренние компоненты Microsoft Windows", по Марку Руссиновичу и Дэвиду Соломону.

## Замечания

В Windows 2000 **!regkcb** отображает определенный блок управления разделом реестра.

В Windows XP и более поздних версиях Windows **вместо этого следует использовать команду расширения !reg** .

Каждый раздел реестра имеет блок управления, содержащий свойства, такие как его разрешения.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rellist

Статья • 05.04.2024

Расширение **!rellist** отображает список отношений самонастраивающийся.

```
dbgcmd
```

```
!rellist Address [Flags]
```

## Параметры

*Адрес*
Указывает адрес списка реляционных отношений.

*Flags*
Указывает дополнительные сведения для включения в дисплей. Это может быть любое сочетание следующих битов (значение по умолчанию равно нулю):

Бит 1 (0x2)
Приводит к включению CM_RESOURCE_LIST дисплея. Отображение также включает список ресурсов загрузки, если он доступен.

Бит 2 (0x4)
Приводит к включению списка требований к ресурсам (IO_RESOURCE_LIST).

Бит 3 (0x8)
Приводит к включению переведенного CM_RESOURCE_LIST.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

См. самонастраивающийся отладку приложений этой команды расширения. Сведения об этих структурах списков см. в документации по комплекту драйверов Windows (WDK).

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ruleinfo

Статья • 05.04.2024

Команда **!ruleinfo** отображает сведения о правиле проверки драйверов.

```
dbgcmd
```

```
!ruleinfo RuleId [RuleState [SubState]]
```

## Параметры

*RuleId*
Идентификатор правила проверяющего средства. Это первый аргумент **ошибки** DRIVER_VERIFIER_DETECTED_VIOLATION проверка.

*RuleState*
Дополнительные сведения о нарушении. Это третий аргумент **ошибки** **DRIVER_VERIFIER_DETECTED_VIOLATION** проверка.

*SubState*
Сведения о нарушении. Это четвертый аргумент **ошибки** **DRIVER_VERIFIER_DETECTED_VIOLATION проверка**.

## DLL-библиотеки

ext.dll

## Замечания

Эта команда применяется только к правилам в расширении средства проверки драйверов; То есть правила, имеющие идентификатор больше или равно 0x10000.

В следующем примере показаны четыре аргумента **ошибки** DRIVER_VERIFIER_DETECTED_VIOLATION проверка.

```
dbgcmd
```

```
DRIVER_VERIFIER_DETECTED_VIOLATION (c4)
...
Arguments:
Arg1: 0000000000091001, ID of the 'NdisOidComplete' rule that was violated.
```

```
Arg2: fffff800002d49d0, A pointer to the string describing the violated rule
condition.
Arg3: ffffe000027b8370, Address of internal rule state (second argument to
!ruleinfo).
Arg4: ffffe000027b83f8, Address of supplemental states (third argument to
!ruleinfo).

## Debugging Details:


DV_VIOLATED_CONDITION:  This OID should only be completed with
NDIS_STATUS_NOT_ACCEPTED,
                        NDIS_STATUS_SUCCESS, or NDIS_STATUS_PENDING.

DV_MSDN_LINK: https://go.microsoft.com/fwlink/p/?linkid=278802

DRIVER_OBJECT: ffffe0000277a2b0
...

STACK_TEXT:
ffffd000`2118ff58 fffff803`4c83afa2 : 00000000`000000c4 00000000`00000001
...
ffffd000`2118ff60 fffff803`4c83a8c0 : 00000000`00000003 00000000`00091001
...
...

STACK_COMMAND:  kb

FOLLOWUP_NAME:  Xxxx

FAILURE_BUCKET_ID:  Xxxx
...
```

В предыдущих выходных данных идентификатор правила (0x91001) отображается как Arg1. Arg3 и Arg4 — это адреса сведений о состоянии правила и вложенных данных. Вы можете передать идентификатор правила, состояние правила и подстат **в !ruleinfo** , чтобы получить описание правила и ссылку на подробную документацию правила.

dbgcmd

```
3: kd> !ruleinfo 0x91001 0xffffe000027b8370 0xffffe000027b83f8

RULE_ID: 0x91001

RULE_NAME: NdisOidComplete

RULE_DESCRIPTION: ""
This rule verifies if an NDIS miniport driver completes an OID correctly.
Check RULE_STATE for Oid ( use !ndiskd.oid ), which can be one of the
following:
1) NULL,
```

```
2) Pending OID, or
3) Previous OID if no OID is pending.

MSDN_LINK: https://learn.microsoft.com/windows-
hardware/drivers/devtest/ndis-ndisoidcomplete

CONTEXT: Miniport 0xFFFFE0000283F1A0

CURRENT_TIME (Timed Rules): 142 seconds

RULE_STATE: 0xFFFFE000027B83F8
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Запущена

Статья • 05.04.2024

Расширение **!running** отображает список запущенных потоков на всех процессорах целевого компьютера.

```
dbgcmd
```

```
!running [-i] [-t]
```

## Параметры

**-i**
Приводит к тому, что дисплей также включает в себя неактивные процессоры.

**-t**
Приводит к отображению трассировки стека для каждого процессора.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения об отладке многопроцессорных компьютеров см. в разделе "Синтаксис многопроцессора".

## Замечания

Без параметров **!running** отображает сходство всех активных процессоров и всех бездействующих процессоров. Для всех активных процессоров также будут отображаться текущие и следующие поля потоков из блока управления процессами (PRCB) и состояние встроенных блокировок спина в очереди 16.

Ниже приведен пример из многопроцессорной системы Itanium:

```
dbgcmd
```

```
0: kd> !running
```

```
System Processors 3 (affinity mask)
 Idle Processors 0

     Prcb              Current           Next
  0  e0000000818f8000  e0000000818f9e50  e0000000866f12f0  ................
 1  e000000086f16010  e00000008620ebe0  e000000086eddbc0  .O..............
```

16 символов в конце каждой строки указывают встроенные блоки спина в очереди (записи LockQueue в PRCB). Период (.) указывает, что блокировка не используется, O **означает, что блокировка принадлежит этому обработчику, а W означает,** что процессор помещается в очередь для блокировки. Чтобы просмотреть дополнительные сведения о очереди блокировки спина, используйте **!qlocks**.

Ниже приведен пример, показывающий активные и неактивные процессоры, а также трассировки стека:

dbgcmd

```
0: kd> !running -it

System Processors f (affinity mask)
  Idle Processors f
All processors idle.

     Prcb      Current   Next
  0  ffdff120  805495a0            ................

ChildEBP RetAddr
8053e3f0 805329c2 nt!RtlpBreakWithStatusInstruction
8053e3f0 80533464 nt!_KeUpdateSystemTime+0x126
ffdff980 ffdff980 nt!KiIdleLoop+0x14

 1  f87e0120  f87e2e60            ................

ChildEBP RetAddr
f87e0980 f87e0980 nt!KiIdleLoop+0x14

 2  f87f0120  f87f2e60            ................

ChildEBP RetAddr
f87f0980 f87f0980 nt!KiIdleLoop+0x14

  3  f8800120  f8802e60            ................

ChildEBP RetAddr
f8800980 f8800980 nt!KiIdleLoop+0x14
```

---

# Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Scm

Статья • 05.04.2024

Расширение **!scm** отображает указанную карту общего кэша.

В Windows XP и более поздних версиях Windows используйте **команду dt nt!_SHARED_CACHE_MAP Address** вместо **!scm**.

dbgcmd

```
!scm Address
```

## Параметры

*Адрес*
Указывает адрес карты общего кэша.

## DLL-библиотеки

Рекомендации недоступны

## Дополнительная информация

Сведения об управлении кэшем см. в документации по microsoft Windows SDK и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

Дополнительные сведения о других расширениях управления кэшем см. в расширении !cchelp.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Поиск

Статья • 05.04.2024

Расширение **!search** выполняет поиск страниц в физической памяти для данных размера указателя, которые соответствуют указанным критериям.

Синтаксис

dbgcmd

```
!search [-s] [-p] Data [ Delta [ StartPFN [ EndPFN ]]]
!search -?
```

# Параметры

**-s**
Вызывает проверка ошибки символов, которые будут игнорироваться во время поиска. Это полезно, если вы получаете слишком много неправильных символов для ошибок ядра.

**-p**
Вызывает интерпретацию значения *Данных* как 32-разрядное значение, предотвращая любое расширение знака.

*Данные*
Указывает данные для поиска. *Данные* должны быть размером указателя в целевой системе (32 бита или 64 бита). Всегда отображается точное соответствие значению *данных* . Другие совпадения отображаются также в зависимости от значения *Delta*; дополнительные сведения см. в разделе "Примечания" ниже.

*Разностная версия*
Указывает допустимое различие между значением в памяти и значением *Data*. Дополнительные сведения см. в разделе "Примечания" ниже.

*StartPFN*
Указывает номер кадра страницы (PFN) начала диапазона для поиска. Если это опущено, поиск начинается на самой низкой физической странице.

*EndPFN*
Указывает номер кадра страницы (PFN) конца диапазона для поиска. Если это опущено, поиск заканчивается на самой высокой физической странице.

**-?**
Отображает справку по этому расширению в окне команды отладчика.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Дополнительные способы отображения и поиска физической памяти см. в статье "Чтение и запись памяти".

# Замечания

Если *заданы startPFN* и *EndPFN* , они используются в качестве номеров кадров страницы начала и конца диапазона в физической памяти для поиска. Описание номеров кадров страницы см. в разделе "Преобразование виртуальных адресов в физические адреса". Если *опущены startPFN* и *EndPFN* , выполняется поиск по всей физической памяти.

Отображаются все хиты.

**Расширение !search** будет выполнять поиск по всей памяти в указанном диапазоне страниц и проверять каждое ULONG_PTR выровненное значение. Отображаются значения, удовлетворяющие по крайней мере одному из следующих критериев:

- Значение точно соответствует *данным* .
- Если значение delta равно 0 или опущено: значение отличается от *значения Data* по одному биту.
- Если delta ненулевое значение: значение отличается от *данных* по крайней мере *delta*. Другими словами, значение находится в диапазоне [Data - Delta, Data + Delta].
- Если delta ненулево: значение отличается от наименьшего числа в диапазоне (Data - Delta) на один бит.

В большинстве случаев *данные* указывают нужный адрес, но можно указать любой ULONG_PTR размер данных.

Поскольку структуры поисковой системы отладчика находятся в памяти на целевом компьютере, при поиске всей памяти (или любого диапазона, содержащего эти структуры), вы увидите совпадения в области, в которой находятся сами структуры. Если необходимо устранить эти совпадения, выполните поиск случайного значения; Это указывает, где находятся структуры поиска отладчика.

Ниже приведено несколько примеров. Ниже приведен поиск на странице памяти с 0x237D PFN для значений между 0x80001230 и 0x80001238 включительно:

dbgcmd

```
kd> !search 80001234 4 237d 237d
```

Ниже будут искать страницы памяти, начиная от PFN 0x2370 до 0x237F для значений, которые находятся в пределах одного бита 0x0F100F0F. Точные совпадения указываются полужирным шрифтом; Остальные выключаются по одному биту:

dbgcmd

```
kd> !search 0f100f0f 0 2370 237f
Searching PFNs in range 00002370 - 0000237F for [0F100F0F - 0F100F0F]

Pfn      Offset   Hit      Va       Pte
- - - - - - - - - - - - - - - - - -
0000237B 00000368 0F000F0F 01003368 C0004014
0000237C 00000100 0F100F0F 01004100 C0004014
0000237D 000003A8 0F100F0F 010053A8 C0004014
0000237D 000003C8 0F100F8F 010053C8 C0004014
0000237D 000003E8 0F100F0F 010053E8 C0004014
0000237D 00000408 0F100F0F 01005408 C0004014
0000237D 00000428 0F100F8F 01005428 C0004014
Search done.
```

Столбцы в отображении приведены следующим образом: **Pfn** — это номер кадра страницы (PFN) страницы; **Смещение — это смещение** на этой странице; **Хит** — это значение по указанному адресу; **Va** — это виртуальный адрес, сопоставленный с этим физическим адресом (если это существует и может быть определено); **Pte** — это запись таблицы страницы (PTE).

Чтобы вычислить физический адрес, переместите PFN влево три шестнадцатеричные цифры (12 битов) и добавьте смещение. Например, последняя строка таблицы — это виртуальный адрес 0x0237D000 + 0x428 = 0x0237D428.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !searchpte

Статья • 05.04.2024

Расширение **!searchpte** выполняет поиск физической памяти для указанного номера кадров страницы (PFN).

dbgcmd

```
!searchpte PFN
!searchpte -?
```

## Параметры

*PFN*
Указывает PFN в шестнадцатеричном формате.

**-?**
Отображает справку по этому расширению в окне команды отладчика.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о таблицах страниц и каталогах *страниц см. в разделе* "Внутренние компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

## Замечания

Чтобы остановить выполнение в любое время, нажмите клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !Sel

Статья • 05.04.2024

Команда **расширения !sel** устарела. **Используйте вместо этого команду dg (селектор** отображения).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Сессии

Статья • 05.04.2024

**Расширение !session** управляет контекстом сеанса. Он также может отобразить список всех сеансов пользователей.

Синтаксис

dbgcmd

```
!session
!session -s DefaultSession
!session -?
```

## Параметры

**-s** **** *DefaultSession*
Задает контекст сеанса заданному значению. Если *значение DefaultSession* равно -1, контекст сеанса устанавливается в текущий сеанс.

**-?**
Отображает справку по этому расширению в окне команды отладчика.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения о сеансах пользователей и диспетчере сеансов *(smss.exe) см. в разделе* "Внутренние компоненты Microsoft Windows", по Марку Руссиновичу и Дэвиду Соломону.

## Замечания

**Расширение !session** используется для управления контекстом сеанса. Использование **!session** без параметров будет отображать список активных сеансов на целевом компьютере. Использование **!s***DefaultSession* изменит контекст сеанса на новое значение по умолчанию.

При настройке контекста сеанса контекст процесса автоматически изменяется на активный процесс для этого сеанса, а **параметр .cache forcedecodeptes** включен таким образом, чтобы адреса сеанса были переведены правильно.

Дополнительные сведения и список всех расширений, связанных с сеансом, затронутых контекстом сеанса, см. в разделе "Изменение контекстов".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Smt

Статья • 05.04.2024

Расширение **!smt** отображает сводку по данным одновременного многопоточного процессора.

dbgcmd

```
!smt
```

## DLL-библиотеки

Kdexts.dll

## Замечания

Рассмотрим пример:

dbgcmd

```
lkd> !smt
SMT Summary:
------------
   KeActiveProcessors: **------------------------------ (00000003)
        KiIdleSummary: ------------------------------ (00000000)
 No PRCB     Set Master SMT Set                                    IAID
  0 820f4820 Master     **------------------------------ (00000003)  00
  1 87a4d120 820f4820   **------------------------------ (00000003)  01

Maximum cores per physical processor:   2
Maximum logical processors per core:    1
```

Столбец **No** указывает количество процессора.

Столбец **PRCB** указывает адрес блока управления процессором для процессора. Каждый логический процессор указан отдельно.

Каждый физический процессор имеет ровно один логический процессор, указанный как **главный** в столбце **Set Master** .

Столбец **SMT Set** содержит сведения о одновременном многопоточных наборах процессоров процессора.

В столбце **IAID** отображается начальный идентификатор контроллера контроллера прерываний расширенной программы (ИДЕНТИФИКАТОР APIC). На истинном компьютере x64 каждый процессор начинается с жестко закодированного исходного идентификатора APIC. Это значение идентификатора можно получить с помощью инструкции CPUID. На некоторых других компьютерах исходный идентификатор APIC не обязательно является уникальным для всех процессоров, поэтому идентификатор APIC, доступный через пространство, сопоставленное с памятью и выводом (MMIO) APIC, может быть изменено. Этот метод позволяет программному обеспечению выделять уникальные идентификаторы APIC для всех процессоров на компьютере. В зависимости от процессоров **целевого компьютера столбец IAID** может отображать этот идентификатор или может быть пустым.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !sprocess

Статья • 05.04.2024

Расширение **!sprocess** отображает сведения о указанном процессе сеанса или обо всех процессах в указанном сеансе.

dbgcmd

```
!sprocess Session [Flags [ImageName]]
!sprocess -?
```

# Параметры

*Согласованность сеанса*
Указывает сеанс, принадлежащий требуемому процессу. *Сеанс* всегда интерпретируется как десятичное число.

*Сеанс* может иметь следующие значения:

Развернуть таблицу

| | |
|---|---|
| -1 | Используйте текущий сеанс. Это значение по умолчанию. |
| -2 | Используйте контекст сеанса. |
| -4 | Отображение всех процессов по сеансу. |

*Flags*
Указывает уровень детализации в отображении. *Флаги* могут быть любым сочетанием следующих битов. По умолчанию установлено значение 0.

Развернуть таблицу

| | |
|---|---|
| 0x0 | Отображение минимальной информации. |
| Bit 0 (0x1) | Отображение статистики времени и приоритета. |
| Бит 1 (0x2) | Добавляется в список потоков и событий, связанных с процессом, и состояния ожидания потоков. |

| | |
|---|---|
| Бит 2 (0x4) | Добавляет в отображение списка потоков, связанных с процессом. Если этот бит используется без бита 1 (0x2), каждый поток отображается в одной строке. Если это включено с битом 1, каждый поток отображается с трассировкой стека. |
| Бит 3 (0x8) | Добавляет к отображению каждой функции возвращаемый адрес, указатель стека и, в системах на основе Itanium, **значение регистра bsp** . Он подавляет отображение аргументов функции. |
| Бит 4 (0x10) | Отображение только возвращаемого адреса каждой функции. Подавление аргументов и указателей стека. |

*ImageName*
Указывает имя отображаемого процесса. Все процессы, имена исполняемых изображений которых соответствуют *ImageName* . Имя изображения должно совпадать с именем в блоке EPROCESS. Как правило, это исполняемое имя, которое было вызвано для запуска процесса, включая расширение файла (обычно .exe), и усечено после пятнадцатого символа. Невозможно указать имя изображения, содержащее пробел.

**-?**
Отображает справку по этому расширению в окне команды отладчика. Этот текст справки имеет некоторые упущения.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о сеансах и процессах в режиме ядра см. в разделе "Изменение контекстов". Дополнительные сведения об анализе процессов и потоков см. в разделе "*Внутренние* компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

# Замечания

Выходные данные этого расширения похожи на **!process**, за исключением того, что отображаются адреса _MM_SESSION_SPACE и _MMSESSION.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Srb

Статья • 05.04.2024

Расширение !srb **отображает сведения о блоке** запросов SCSI (SRB).

```
dbgcmd
```

```
!srb Address
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес SRB на целевом компьютере.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения о SRB см. в документации по комплекту драйверов Windows (WDK).

## Замечания

SRB — это системная структура, используемая для передачи запросов ввода-вывода от драйвера класса SCSI к драйверу порта SCSI.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Стеки

Статья • 05.04.2024

Расширение **!stacks** отображает сведения о стеках ядра.

Синтаксис

```
dbgcmd

!stacks [Detail [FilterString]]
```

## Параметры

*Detail*
Указывает уровень детализации, используемый в отображении. В следующей таблице перечислены допустимые значения для *сведений*.

Развернуть таблицу

| | |
|---|---|
| 0 | Отображает сводку по текущим стекам ядра. Это значение по умолчанию. |
| 1 | Отображает стеки, которые в настоящее время выводятся на страницу, а также текущие стеки ядра. |
| 2 | Отображает полные параметры для всех стеков, а также стеки, которые в настоящее время выводятся и текущие стеки ядра. |

*FilterString*
Отображает только потоки, содержащие указанную подстроку в символе.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о стеках *ядра см. в разделе* "Внутренние компоненты Microsoft Windows", Марк Руссинович и Дэвид Соломон.

# Замечания

Расширение **!stacks** содержит краткую сводку о состоянии каждого потока. Это расширение можно использовать вместо **расширения !process** , чтобы получить краткий обзор системы, особенно при отладке многопоточных проблем, таких как конфликты ресурсов или взаимоблокировки.

Расширение **пользовательского режима !findstack** также отображает сведения о конкретных стеках.

Ниже приведен пример простого **отображения !stacks** :

dbgcmd

```
kd> !stacks 0
Proc.Thread  .Thread  ThreadState  Blocker
                                   [System]
   4.000050  827eea10  Blocked    +0xfe0343a5

                                   [smss.exe]

                                   [csrss.exe]
  b0.0000a8  82723b70  Blocked    ntoskrnl!_KiSystemService+0xc4
  b0.0000c8  82719620  Blocked    ntoskrnl!_KiSystemService+0xc4
  b0.0000d0  827d5d50  Blocked    ntoskrnl!_KiSystemService+0xc4
.....
```

Первый столбец показывает идентификатор процесса и идентификатор потока (разделенный на период).

Второй столбец — текущий адрес блока ETHREAD потока.

Третий столбец показывает состояние потока (инициализировано, готово, выполняется, резервная, завершенная, переходная или заблокированная).

Четвертый столбец показывает верхний адрес в стеке потока.

Ниже приведены примеры более подробных **выходных данных !stacks** :

dbgcmd

```
kd> !stacks 1
Proc.Thread  .Thread  ThreadState  Blocker
                                   [System]
   4.000008  827d0030  Blocked    ntoskrnl!MmZeroPageThread+0x66
   4.000010  827d0430  Blocked    ntoskrnl!ExpWorkerThread+0x189
   4.000014  827cf030  Blocked    Stack paged out
   4.000018  827cfda0  Blocked    Stack paged out
```

```
   4.00001c  827cfb10  Blocked    ntoskrnl!ExpWorkerThread+0x189
.....
                                     [smss.exe]
  9c.000098  82738310  Blocked    Stack paged out
  9c.0000a0  826a5190  Blocked    Stack paged out
  9c.0000a4  82739d30  Blocked    Stack paged out

                                     [csrss.exe]
  b0.0000bc  826d0030  Blocked    Stack paged out
  b0.0000b4  826c9030  Blocked    Stack paged out
  b0.0000a8  82723b70  Blocked    ntoskrnl!_KiSystemService+0xc4
.....

kd> !stacks 2
Proc.Thread  .Thread  ThreadState  Blocker
                                     [System]
   4.000008  827d0030  Blocked    ntoskrnl!KiSwapThread+0xc5
                                  ntoskrnl!KeWaitForMultipleObjects+0x2b4
                                  ntoskrnl!MmZeroPageThread+0x66
                                  ntoskrnl!Phase1Initialization+0xd82
                                  ntoskrnl!PspSystemThreadStartup+0x4d
                                  ntoskrnl!CreateSystemRootLink+0x3d8
                                  +0x3f3f3f3f
   4.000010  827d0430  Blocked    ntoskrnl!KiSwapThread+0xc5
                                  ntoskrnl!KeRemoveQueue+0x191
.....
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Swd

Статья • 05.04.2024

Расширение **!swd** отображает состояния таймера программного обеспечения для указанного процессора, включая отложенный вызов процедуры (DPC) и состояния таймера сторожевой системы для потоков.

dbgcmd

```
!swd [Processor]
```

## Параметры

*Процессор*
Задает процессор. Если *обработчик* опущен, сведения отображаются для всех процессоров на целевом компьютере.

## DLL-библиотеки

Kdexts.dll

## Замечания

Таймер наблюдателя завершает работу или перезапускает Windows, если Windows перестает отвечать. Время отображается в секундах.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !sysinfo

Статья • 08.07.2024

Расширение **!sysinfo** считывает и отображает указанные SMBIOS, Advanced Configuration and Power Interface (ACPI) и данные ЦП из файла дампа или динамической системы.

dbgcmd

```
!sysinfo cpuinfo [-csv [-noheaders]]
!sysinfo cpumicrocode [-csv [-noheaders]]
!sysinfo cpuspeed [-csv [-noheaders]]
!sysinfo gbl [-csv [-noheaders]]
!sysinfo machineid [-csv [-noheaders]]
!sysinfo registers
!sysinfo smbios [-csv [-noheaders]] {-debug | -devices | -memory | -power |
-processor | -system | -v}
!sysinfo -?
```

# Параметры

**cpuinfo**
Отображает сведения о процессоре.

**cpumicrocode**
(только для процессоров GenuineIntel) Отображает начальные и кэшированные версии процессора микрокода.

**cpuspeed**
Отображает максимальную и текущую скорость процессора.

**gbl**
Отображает список BIOS таблиц ACPI.

**machineid**
Отображает сведения об идентификаторе компьютера для SMBIOS, BIOS, встроенного ПО, системы и базовой доски.

**Регистров**
Отображает регистры, относящиеся к компьютеру (MSR).

**smbios**
Отображает таблицу SMBIOS.

**-csv**
Отображает все данные в формате с разделимой запятыми, переменной длины (CSV).

**-noheaders**
Подавляет заголовок формата CSV.

**-отлаживать**
Отображает выходные данные в стандартном формате и формате CSV.

**-приборы**
Отображает записи устройства в таблице SMBIOS.

**-память**
Отображает записи памяти в таблице SMBIOS.

**-сила**
Отображает записи питания в таблице SMBIOS.

**-процессор**
Отображает записи процессора в таблице SMBIOS.

**-система**
Отображает системные записи в таблице SMBIOS.

**-v**
Verbose. Отображает сведения о записях в таблице SMBIOS.

**-?**
Отображает справку по этому расширению в окне команды отладчика.

# DLL-библиотеки

Kdexts.dll

# Замечания

Это расширение полезно только в том случае, если файл дампа является файлом аварийного сбоя системы (.dmp), который не был преобразован в мини-файл из ядра или полного дампа, или динамическая система завершена и находится в сети (например, в запросе входа).

Вы можете использовать любое сочетание **параметров -debug**, -devices**, -memory**, -power**, -processor**, **-system** и **-v** в одной команде расширения.

Следующие параметры поддерживаются только в определенных системах:

- Параметр **gbl** работает только в том случае, если целевой компьютер поддерживает ACPI.
- Параметр **smbios** работает только в том случае, если целевой компьютер поддерживает SMBIOS.

Корпорация Майкрософт предпринимает все усилия по удалению личных сведений (PII) из этих записей. Все личные данные удаляются из файлов дампа. Однако в динамической системе некоторые личные данные пока не будут удалены. В результате поля PII будут сообщаться как 0 или пустые, даже если они фактически содержат информацию.

Чтобы остановить выполнение команд, включающих **cpuinfo**, gbl**, registers** или smbios **параметры в любое время, нажмите клавиши CTRL+BREAK (в WinDbg) или** CTRL+C (в KD).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !sysptes

Статья • 08.07.2024

Расширение **!sysptes** отображает форматное представление записей таблицы системной страницы (PTEs).

dbgcmd

```
!sysptes [Flags]
```

# Параметры

*Флаги*
Указывает уровень детализации для отображения. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию равно нулю:

Bit 0 (0x1)
Отображает сведения о бесплатных PTEs.

Бит 1 (0x2)
(Только Windows 2000) Отображает неиспользуемые страницы в статистике использования страниц.

Отображает сведения о бесплатных PTEs в глобальном специальном пуле.

Бит 2 (0x4)
Отображает подробные сведения о любых системных PTEs, выделенных для сопоставления заблокированных страниц.

Бит 3 (0x8)
(Только Windows 2000 и Windows XP) Отображает непагаченные сведения о расширении пула. Если этот бит задан, другие списки не отображаются. Если заданы оба 0x1 и 0x8, отображаются все непагированные PTES расширения пула. Если задано только 0x8, отображается только общее значение.

Бит 4 (0x10)
(Windows Vista и более поздние версии) Отображает специальный пул бесплатных сведений О PTE для сеанса.

## DLL-библиотеки

# Дополнительная информация

Сведения о таблицах *страниц и PTEs см. в разделе* "Внутренние компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

# Замечания

Чтобы изучить конкретный PTE, используйте **расширение !pte** .

Ниже приведен пример из системы Windows:

dbgcmd

```
kd> !sysptes 1

System PTE Information
  Total System Ptes 571224
     SysPtes list of size 1 has 361 free
     SysPtes list of size 2 has 91 free
     SysPtes list of size 4 has 48 free
     SysPtes list of size 8 has 36 free
     SysPtes list of size 9 has 29 free
     SysPtes list of size 23 has 29 free

    starting PTE: fffffe0059388000
    ending PTE:   fffffe00597e3ab8

      free ptes: fffffe0059388000   number free: 551557.
      free ptes: fffffe00597be558   number free: 104.
      free ptes: fffffe00597d2828   number free: 676.

  free blocks: 3   total free: 552337    largest free block: 551557
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Поток

Статья • 05.04.2024

Расширение **!thread** отображает сводную информацию о потоке в целевой системе, включая блок ETHREAD. Эта команда может использоваться только во время отладки в режиме ядра.

Эта команда расширения не совпадает с командой **.thread (Set Register Context).**

Синтаксис

dbgcmd

```
!thread [-p] [-t] [Address [Flags]]
```

# Параметры

**-p**
Отображает сводные сведения о процессе, который владеет потоком.

**-t**
Если этот параметр включен, *адрес* — это идентификатор потока, а не адрес потока.

*Адрес*
Указывает шестнадцатеричный адрес потока на целевом компьютере. Если *адрес* равен -1 или опущен, он указывает текущий поток.

*Flags*
Указывает уровень детализации для отображения. *Флаги* могут быть любым сочетанием следующих битов. Если *флаги* равно 0, отображается только минимальное количество сведений. Значение по умолчанию — 0x6:

Бит 1 (0x2)
Отображает состояния ожидания потока.

Бит 2 (0x4)
Если этот бит используется без бита 1 (0x2), он не действует. Если этот бит используется с битом 1, поток отображается с трассировкой стека.

Бит 3 (0x8)
Добавляет возвращаемый адрес, указатель стека и (в системах Itanium) **значение**

**регистра bsp** в сведения, отображаемые для каждой функции, и подавляет отображение аргументов функции.

Бит 4 (0x10)
Задает контекст процесса, равный процессу, которому принадлежит указанный поток в течение этой команды. Это приводит к более точному отображению стека потоков.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о потоках в режиме ядра см. в разделе "Изменение контекстов " и "Управление процессами и потоками". Дополнительные сведения об анализе процессов и потоков см. в статье *Microsoft Windows Internals*, Марк Руссинович, Алекс Ионеску и Дэвид Соломон.

# Замечания

Ниже приведен пример использования Windows 10:

dbgcmd

```
0: kd> !thread 0xffffcb088f0a4480
THREAD ffffcb088f0a4480  Cid 0e34.3814  Teb: 0000001a27ca6000 Win32Thread:
0000000000000000 RUNNING on processor 0
Not impersonating
DeviceMap                 ffffb80842016c20
Owning Process            ffffcb08905397c0       Image:         MsMpEng.exe
Attached Process          N/A            Image:         N/A
Wait Start TickCount      182835891      Ticks: 0
Context Switch Count      5989           IdealProcessor: 3
UserTime                  00:00:01.046
KernelTime                00:00:00.296
Win32 Start Address 0x00007ffb3b2fd1b0
Stack Init ffff95818476add0 Current ffff958184769d30
Base ffff95818476b000 Limit ffff958184765000 Call 0000000000000000
Priority 8 BasePriority 8 PriorityDecrement 0 IoPriority 2 PagePriority 5
Child-SP          RetAddr           : Args to Child
: Call Site
fffff802`59858c68 fffff801`b56d24aa : ffffcb08`8fd68010 00000000`00000000
fffff802`58259600 00000000`00000008 : nt!DbgBreakPointWithStatus
[d:\rs2\minkernel\ntos\rtl\amd64\debugstb.asm @ 130]
fffff802`59858c70 ffffcb08`8fd68010 : 00000000`00000000 fffff802`58259600
```

```
00000000`00000008 ffffcb08`8f0a4400 : 0xfffff801`b56d24aa
fffff802`59858c78 00000000`00000000 : fffff802`58259600 00000000`00000008
ffffcb08`8f0a4400 00000000`00000019 : 0xffffcb08`8fd68010
```

Используйте такие команды, как **!process** , чтобы найти адрес или идентификатор потока интересующего вас потока.

Полезные сведения в отображении !thread **описаны в следующей** таблице.

Развернуть таблицу

| Параметр | Значение |
|---|---|
| **Адрес потока** | Шестнадцатеричное число после слова *THREAD* является адресом блока ETHREAD. В предыдущем примере адрес потока 0xffffcb088f0a4480. |
| **Идентификатор потока** | Два шестнадцатеричных числа после слова *Cid* — это идентификатор процесса и идентификатор потока: *идентификатор процесса ID.thread*. В предыдущем примере идентификатор процесса — 0x0e34, а идентификатор потока — 0x3814. |
| **Блок среды потока (TEB)** | Шестнадцатеричное число после слова *Teb* — адрес блока среды потока (TEB). |
| **Win32Thread** | Шестнадцатеричное число после слова *Win32Thread* — это адрес Win32Thread. |
| **Состояние потока** | Состояние потока отображается в конце строки, начинающейся с слова *RUNNING*. |
| **Процесс владения** | Шестнадцатеричное число после слова "*Процесс владения* " — это адрес EPROCESS для процесса, который владеет этим потоком. |
| **Начальный адрес** | Шестнадцатеричное число после слов "*Начальный адрес* " — это адрес начала потока. Это может отображаться в символьной форме. |
| **Функция пользовательского потока** | Шестнадцатеричное число после слов *Win32 Start Address* — это адрес функции потока пользователя. |
| **Приоритет** | Сведения о приоритете для потока соответствуют слову *Priority*. |

| Параметр | Значение |
| --- | --- |
| **Трассировка стека** | Трассировка стека для потока отображается в конце этого дисплея. |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !threadfields

Статья • 05.04.2024

Команда **расширения !threadfields** недоступна. Вместо этого используйте dt (тип отображения)..

```
dbgcmd
```

```
!threadfields
```

## DLL-библиотеки

Недоступно (см. раздел "Примечания")

## Дополнительная информация

Сведения о блоке ETHREAD см. в разделе "*Внутренние* компоненты Microsoft Windows", Марк Руссинович и Дэвид Соломон.

## Замечания

Эта команда расширения недоступна. Вместо этого используйте **команду dt (display Type),** чтобы отобразить структуру ETHREAD напрямую:

```
dbgcmd
```

```
kd> dt nt!_ETHREAD
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Время

Статья • 05.04.2024

Команда **расширения !time** отображает сведения о системном времени и времени прерывания и может работать с явными метками времени и текущим временем.

## Среда

Развернуть таблицу

| Позиция | Description |
|---|---|
| Режимы | Режим пользователя, режим ядра |
| Целевые объекты | Динамический, аварийный дампа |
| Платформы | Все |

## Замечания

Ниже приведен пример в пользовательском режиме:

dbgcmd

```
0:000> !time
CURRENT TIME:
System:               01d75f1c`8588b6ee (2021 Jun 11 23:50:13.477)
Interrupt:            0000020e`22112c46 (2 days, 14:46:12.434)
```

## См. также

.time (отображаемое системное время)

.echotime (показать текущее время)

.echotimestamps (показать метки времени)

!Беглых

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Таймер

Статья • 05.04.2024

Расширение **!timer** отображает подробный список всех системных таймеров.

dbgcmd

```
!timer
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о объектах таймера см. в документации по комплекту драйверов Windows (WDK).

## Замечания

Расширение **!timer** отображает дерево таймера, которое хранит все объекты таймера в системе.

Рассмотрим пример:

dbgcmd

```
kd> !timer
Dump system timers

Interrupt time: 9f760774 00000000 [12/ 8/2000 10:59:22.685 (Pacific Standard
Time)]

List Timer    Interrupt Low/High     Fire Time              DPC/thread
 0 8016aea0 P 9fbd8e00 00000000 [12/ 8/2000 10:59:23.154]
ntoskrnl!PopScanIdleList
  1 8257f118   e4e4225a 00000000 [12/ 8/2000 11:01:19.170]  thread 8257f030
 3 80165fc0   286be1c9 0000594a [ 4/ 1/2001 01:59:59.215]
ntoskrnl!ExpTimeZoneDpcRoutine
    80165f40   2a7bf8d9 006f105e [12/31/2099 23:59:59.216]
ntoskrnl!ExpCenturyDpcRoutine
  5 825a0bf8   a952e1c2 00000000 [12/ 8/2000 10:59:39.232]  thread 825a0b10
 10 8251c7a8   41f54d84 00000001 [12/ 8/2000 11:03:55.310]  thread 8251c6c0
    8249fe88   41f54d84 00000001 [12/ 8/2000 11:03:55.310]  thread 8249fda0
```

```
 11 8250e7e8   bc73ffde 00000000 [12/ 8/2000 11:00:11.326]  thread 8250e700

.....

237 82757070   9f904152 00000000 [12/ 8/2000 10:59:22.857]  +f7a56f2e
    82676348   9f904152 00000000 [12/ 8/2000 10:59:22.857]  +fe516352
    82728b78   9f904152 00000000 [12/ 8/2000 10:59:22.857]  +fe516352
238 fe4b5d78   9f92a3ac 00000000 [12/ 8/2000 10:59:22.873]  thread 827ceb10
    801658f0   9f92a3ac 00000000 [12/ 8/2000 10:59:22.873]
ntoskrnl!CcScanDpc
239 8259ad40   765a6f19 00000bba [12/23/2000 09:07:22.900]  thread 825d3670
250 826d42f0   1486bed8 80000000 [         NEVER         ]  thread 825fa030

Total Timers: 193, Maximum List: 7
Current Hand: 226, Maximum Search: 0

Wakeable timers:
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !tokenfields

Статья • 05.04.2024

**Расширение !tokenfields** устарело. Вместо этого используйте команду dt (тип отображения).

```
dbgcmd
```

```
!tokenfields
```

## DLL-библиотеки

Недоступно (см. раздел "Примечания")

## Дополнительная информация

Сведения о структуре TOKEN см. в разделе "*Внутренние* компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон. Эта книга может быть недоступна на некоторых языках и странах. (Структуры маркеров в режиме пользователя, описанные в документации по пакету SDK для Microsoft Windows, немного отличаются.)

## Замечания

Эта команда расширения недоступна в Windows XP или более поздних версиях Windows. Вместо этого используйте **команду dt (тип отображения),** чтобы отобразить структуру TOKEN напрямую:

```
dbgcmd
```

```
kd> dt nt!_TOKEN
```

Чтобы просмотреть конкретный экземпляр структуры TOKEN, используйте **расширение !token** .

Ниже приведен пример **!tokenfields** из системы Windows 2000:

```
dbgcmd
```

```
kd> !tokenfields
 TOKEN structure offsets:
    TokenSource:           0x0
    AuthenticationId:      0x18
    ExpirationTime:        0x28
    ModifiedId:            0x30
    UserAndGroupCount:     0x3c
    PrivilegeCount:        0x44
    VariableLength:        0x48
    DynamicCharged:        0x4c
    DynamicAvailable:      0x50
    DefaultOwnerIndex:     0x54
    DefaultDacl:           0x6c
    TokenType:             0x70
    ImpersonationLevel:    0x74
    TokenFlags:            0x78
    TokenInUse:            0x79
 ProxyData:             0x7c
    AuditData:             0x80
    VariablePart:          0x84
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ловушку

Статья • 01.04.2024

Команда **расширения !trap** устарела. **Вместо этого используйте команду .trap (Display Trap Frame).**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Tss

Статья • 05.04.2024

Команда **расширения !tss** устарела. **Вместо этого используйте команду TSS (сегмент состояния** задачи отображения).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Tz

Статья • 05.04.2024

Расширение **!tz** отображает указанную структуру тепловой зоны питания.

```
dbgcmd

!tz [Address]
```

## Параметры

*Адрес*
Адрес тепловой зоны питания, которую вы хотите отобразить. Если этот параметр опущен, дисплей включает все тепловые зоны на целевом компьютере.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Чтобы просмотреть возможности питания системы, используйте **команду расширения !pocaps** . Чтобы просмотреть политику питания системы, используйте **команду расширения !popolicy** . Сведения о возможностях питания и политике питания см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, марку Руссиновичу и Дэвиду Соломону.

## Замечания

Чтобы остановить выполнение в любое время, нажмите клавиши CTRL+BREAK (в WinDbg) или CTRL+C (в KD).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !tzinfo

Статья • 05.04.2024

Расширение **!tzinfo** отображает содержимое указанной информационной структуры тепловой зоны.

dbgcmd

```
!tzinfo Address
```

## Параметры

*Адрес*
Адрес отображаемой структуры сведений о тепловой зоне.

### DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Чтобы просмотреть возможности питания системы, используйте **команду расширения !pocaps** . Чтобы просмотреть политику питания системы, используйте **команду расширения !popolicy** . Сведения о возможностях питания и политике питания см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, марку Руссиновичу и Дэвиду Соломону.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ubc

Статья • 05.04.2024

**Расширение !ubc** очищает точку останова в пространстве пользователя.

dbgcmd

```
!ubc BreakpointNumber
```

## Параметры

*Число точек останова*
Указывает число точек останова, которые необходимо очистить. Звездочка (*) указывает все точки останова.

### DLL-библиотеки

Kdexts.dll

## Замечания

Это приведет к окончательному удалению набора точек останова с **помощью !ubp**.

## См. также

**!Ubd**

**!Вбо**

**!Убл**

**!Ubp**

Пространство пользователя и системное пространство

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !ubd

Статья • 08.07.2024

Расширение **!ubd** временно отключает точку останова в пространстве пользователя.

```
dbgcmd
```

```
!ubd BreakpointNumber
```

## Параметры

*Число точек останова*
Указывает количество отключаемой точки останова. Звездочка (*) указывает все точки останова.

### DLL-библиотеки

Kdexts.dll

## Замечания

Отключенные точки останова будут игнорироваться. Используйте **!ube** , чтобы повторно включить точку останова.

## См. также

**!ubc**

**!Вбо**

**!ubl**

**!ubp**

Пространство пользователя и системное пространство

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Вбо

Статья • 08.07.2024

Расширение **!ube** повторно включает точку останова в пространстве пользователя.

```
dbgcmd
```

```
!ube BreakpointNumber
```

## Параметры

*Число точек останова*
Указывает число включенных точек останова. Звездочка (*) указывает все точки останова.

### DLL-библиотеки

Kdexts.dll

## Замечания

Используется для повторного включения точки останова, отключенной **!ubd**.

## См. также

**!ubc**

**!ubd**

**!ubl**

**!ubp**

Пространство пользователя и системное пространство

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !Убл

Статья • 05.04.2024

Расширение **!ubl** содержит все точки останова в пространстве пользователя и их текущее состояние.

dbgcmd

```
!ubl
```

## DLL-библиотеки

Kdexts.dll

## Замечания

Ниже приведен пример использования и отображения точек останова пространства пользователя:

dbgcmd

```
kd> !ubp 8014a131
This command is VERY DANGEROUS, and may crash your system!
If you don't know what you are doing, enter "!ubc *" now!

kd> !ubp 801544f4

kd> !ubd 1

kd> !ubl
 0: e ffffffff`8014a131 (ffffffff`82deb000) 1 ffffffff
 1: d ffffffff`801544f4 (ffffffff`82dff000) 0 ffffffff
```

Каждая строка в этом списке содержит номер точки останова, состояние (**включено** или **d** для отключенного), виртуальный адрес, используемый для задания точки останова, физического адреса фактической точки останова, положения байтов и содержимого этого расположения памяти в момент установки точки останова.

## См. также

**!Ubc**

**!Ubd**

**!Вбо**

**!Ubp**

Пространство пользователя и системное пространство

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Ubp

Статья • 05.04.2024

Расширение **!ubp** задает точку останова в пространстве пользователя.

```
dbgcmd

!ubp Address
```

## Параметры

*Адрес*
Указывает шестнадцатеричный виртуальный адрес расположения в пространстве пользователя, в котором должна быть задана точка останова.

### DLL-библиотеки

Kdexts.dll

## Замечания

Расширение **!ubp** задает точку останова в пространстве пользователя. Точка останова устанавливается на фактической физической странице, а не только на виртуальной странице.

Установка физической точки останова будет одновременно изменять каждую виртуальную копию страницы с непредсказуемыми результатами. Одним из возможных последствий является повреждение состояния системы, за которым может следовать ошибка проверка или другой системной аварии. Поэтому эти точки останова следует использовать осторожно, если вообще.

Это расширение нельзя использовать для задания точек останова на страницах, которые были переключены из памяти. Если страница переключается из памяти после установки точки останова, точка останова перестает существовать.

Невозможно задать точку останова внутри таблицы страницы или каталога страницы.

Каждая точка останова назначается номер *точки останова*. Чтобы узнать назначенный номер точки останова, используйте **!ubl**. Точки останова включены

при создании. Чтобы выполнить шаг по точке останова, необходимо сначала отключить ее с помощью **!ubd**. Чтобы очистить точку останова, используйте **!ubc**.

## См. также

**!Ubc**

**!Ubd**

**!Вбо**

**!Убл**

Пространство пользователя и системное пространство

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !urb

Статья • 02.04.2024

Команда **расширения !urb** устарела. **Вместо этого используйте команду dt URB**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Vad

Статья • 05.04.2024

Расширение **!vad** отображает сведения о дескрипторе виртуальных адресов (VAD) или дереве VAD.

- Отображает сведения о дескрипторе одного виртуального адреса (VAD)
- Отображает сведения о дереве VAD.
- Отображает сведения о VAD для определенного модуля пользовательского режима и предоставляет строку, которую можно использовать для загрузки символов для этого модуля.

dbgcmd

```
!vad VAD-Root [Flag]
!vad Address 1
```

# Параметры

*VAD-Root*
Адрес корня дерева VAD, отображаемого.

*Параметр*
Указывает форму, в котором будет приниматься отображение. Возможные значения включают:

0
Отображается все дерево VAD, основанное на *VAD-Root* . (Это значение по умолчанию.)

1
Отображается только VAD-Root, указанный *VAD-Root* . Отображение будет включать более подробный анализ.

*Адрес*
Адрес в диапазоне виртуальных адресов модуля пользовательского режима.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о дескрипторах виртуальных *адресов см. в разделе* "Внутренние компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

# Замечания

Адрес корневого каталога VAD для любого процесса можно найти с помощью **команды !process** .

Команда !vad **может быть полезной**, если необходимо загрузить символы для модуля пользовательского режима, который был выгружаем из памяти. Дополнительные сведения см. в разделе "Сопоставление символов" при выходе реВ.

Ниже приведен пример **расширения !vad** :

dbgcmd

```
kd> !vad 824bc2f8
VAD     level      start      end    commit
82741bf8 ( 1)      78000    78045         8 Mapped  Exe  EXECUTE_WRITECOPY
824ef368 ( 2)      7f6f0    7f7ef         0 Mapped       EXECUTE_READ
824bc2f8 ( 0)      7ffb0    7ffd3         0 Mapped       READONLY
8273e508 ( 2)      7ffde    7ffde         1 Private      EXECUTE_READWRITE
82643fc8 ( 1)      7ffdf    7ffdf         1 Private      EXECUTE_READWRITE

Total VADs:     5  average level:    2  maximum depth: 2

kd> !vad 824bc2f8 1

VAD @ 824bc2f8
  Start VPN:         7ffb0  End VPN:    7ffd3  Control Area:  827f1208
  First ProtoPte: e1008500  Last PTE e100858c  Commit Charge         0 (0.)
  Secured.Flink          0  Blink           0  Banked/Extend:        0
Offset 0
   ViewShare NoChange READONLY

SecNoChange
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !vad_reload

Статья • 08.07.2024

Расширение **!vad_reload** перезагружает модули пользовательского режима для указанного процесса с помощью дескрипторов виртуальных адресов (VAD) этого процесса.

dbgcmd

```
!vad_reload Process
```

## Параметры

*Процесс*
Указывает шестнадцатеричный адрес процесса, для которого будут загружены модули.

## Дополнительная информация

Сведения о VAD см. в разделе "*Внутренние* компоненты Microsoft Windows", Марк Руссинович и Дэвид Соломон.

## Замечания

Расширение !process **можно использовать** для поиска адреса процесса.

Ниже приведен пример поиска адреса и его использования в команде **!vad_reload** .

dbgcmd

```
3: kd> !process 0 0
. . .
PROCESS fffffa8007d54450
    SessionId: 1  Cid: 115c    Peb: 7ffffef6000  ParentCid: 0c58
    DirBase: 111bc3000  ObjectTable: fffff8a006ae0960  HandleCount: 229.
    Image: SearchProtocolHost.exe
. . .
3: kd> !vad_reload fffffa8007d54450
fffffa80`04f5e8b0: VAD maps 00000000`6c230000 - 00000000`6c26bfff, file
cscobj.dll
fffffa80`04e8f890: VAD maps 00000000`6ef90000 - 00000000`6f04afff, file
mssvp.dll
fffffa80`07cbb010: VAD maps 00000000`6f910000 - 00000000`6faf5fff, file
```

```
tquery.dll
fffffa80`08c1f2a0: VAD maps 00000000`6fb80000 - 00000000`6fb9bfff, file
mssprxy.dll
fffffa80`07dce8b0: VAD maps 00000000`6fba0000 - 00000000`6fba7fff, file
msshooks.dll
fffffa80`04fd2e70: VAD maps 00000000`72a50000 - 00000000`72a6cfff, file
userenv.dll
. . .
```

## DLL-библиотеки

Kdexts.dll

## См. также

**!процесс**

**!vad**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !validatelist

Статья • 05.04.2024

Расширение **!validatelist** проверяет допустимость обратных и прямых ссылок в списке, связанном с двойной связью.

dbgcmd

```
!validatelist Address
```

## Параметры

*Адрес*
Адрес списка, связанного с двойной связью.

## DLL-библиотеки

Kdexts.dll

## Замечания

Чтобы остановить выполнение, нажмите клавиши CTRL+Break (в WinDbg) или CTRL+C (в KD).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Проверки

Статья • 05.04.2024

Расширение **!verifier** отображает состояние средства проверки драйверов и его действий.

Средство проверки драйверов входит в Windows. Он работает как на проверка, так и на бесплатных сборках. Дополнительные сведения о средство проверки драйверов см. в разделе "Средство проверки драйверов".

Синтаксис

dbgcmd

```
!verifier [Flags [Image]]
!verifier 4 [Quantity]
!verifier 8 [Quantity]
!verifier 0x40 [Quantity]
!verifier 0x80 [Quantity]
!verifier 0x80 Address
!verifier 0x100 [Quantity]
!verifier 0x100 Address
!verifier 0x200 [Address]
!verifier 0x400 [Address]
!verifier -disable
!verifier ?
```

# Параметры

*Flags*
Указывает, какие сведения отображаются в выходных данных этой команды. Если *флаги* равны значению 4, 8, 0x20, 0x40, 0x80 или 0x100, то остальные аргументы **проверки** интерпретируются на основе конкретных аргументов, связанных с этими значениями. Если *флаги* равны любому другому значению, даже если задано одно или несколько этих битов, разрешены только *аргументы Flags* и *Image* . *Флаги* могут быть любой суммой следующих битов. Значение по умолчанию — 0:

Bit 0 (0x1)
Отображает имена всех проверяемых драйверов. Число байтов, выделенных каждому драйверу из непагрегированного пула и пула страниц, также отображается.

Бит 1 (0x2)
Отображает сведения о пулах (размер пула, заголовки и теги пула) и

невыполненных выделениях памяти, оставленных выгруженными драйверами. Этот флаг не действует, если также не задан бит 0 (0x1).

Бит 2 (0x4)
Отображает сведения о внедрении ошибок. Отображается адрес возврата, имя символа и смещение кода, запрашивающего каждое выделение. Если *флаги* точно 0x4 и *включен параметр* "*Количество* ", можно выбрать количество этих записей. В противном случае отображаются четыре записи.

Бит 3 (0x8)
Отображает последние изменения IRQL, внесенные проверяемыми драйверами. Отображается старая IRQL, новая IRQL, обработчик и метка времени. Если *флаги* точно 0x8 и *включен параметр* "*Количество* ", можно выбрать количество этих записей. В противном случае отображаются четыре записи.

**Предупреждение** в 64-разрядных версиях Windows некоторые функции ядра, которые вызывают или снижают IRQL, реализуются как встроенный код, а не как экспортированные функции. Средство проверки драйверов не сообщает об изменениях IRQL, внесенных встроенным кодом, поэтому журнал перехода IRQL, созданный средством проверки драйверов, может быть неполным. Пример отсутствующих записей перехода IRQL см. в примечаниях.

Бит 6 (0x40)
(Windows Vista и более поздние версии) Отображает сведения из **параметра "Запросы** ожидающих операций ввода-вывода" средства проверки драйвера, включая трассировки из журнала принудительного ожидания irPs.

Параметр *Number* указывает количество отображаемых трассировок. По умолчанию отображается весь журнал.

Бит 7 (0x80)
(Windows Vista и более поздние версии) Отображает сведения из журнала выделения или бесплатного пула ядра.

Параметр *Number* указывает количество отображаемых трассировок. По умолчанию отображается весь журнал.

Если *указан адрес* , отображаются только трассировки, связанные с указанным адресом в пуле ядер " Выделение или бесплатный журнал".

Бит 8 (0x100)
(Windows Vista и более поздние версии) Отображает сведения из журнала вызовов IoAllocateIrp, IoCompleteRequest и IoCancelIrp.

Параметр Number указывает количество отображаемых трассировок. По умолчанию отображается весь журнал.

Если *указан адрес* , отображаются только трассировки, связанные с указанным адресом IRP.

Bit 9 (0x200)
(Windows Vista и более поздние версии) Отображает записи в журнале "Критически важный регион".

Если *указан адрес* , отображаются только записи, связанные с указанным адресом потока.

Бит 10 (0x400)
(Windows Vista и более поздние версии) Отображает отмененные irps, которые в настоящее время отслеживаются проверяющим драйвером.

Если *указан адрес* , отображается только IRP с указанным адресом.

Бит 11 (0x800)
(Windows 8.1 и более поздние версии) Отображение записей из журнала внедрения ошибок, созданного при выборе параметра "Систематическое моделирование низких ресурсов".

*Изображение*
Если *флаги* используются и не равны 4, 8 или 0x10, *изображение* указывает имя драйвера. *Изображение* используется для фильтрации информации, отображаемой значениями *флагов* 0x1 и 0x2: считается только указанный драйвер. Этот драйвер должен быть проверен в настоящее время.

*Количество*
Если *флаги* точно равны 0x4, *количество* указывает количество записей внедрения ошибок для отображения. Если *флаги* точно равны 0x8, *количество* указывает количество отображаемых записей журнала IRQL. Если *флаги* равны точно 0x40, количество *трассировок,* отображаемых в журнале принудительного ожидания irPs. Если флаги точно равны 0x80, количество трассировок, отображаемых в пуле ядра, выделяет или бесплатный журнал. Если флаги равны точно 0x100, количество трассировок, отображаемых в журнале IoAllocateIrp, IoCompleteRequest и IoCancelIrp.

**-Отключить**
Очищает текущие параметры средства проверки драйверов в целевом объекте отладки. Очистка этих параметров не сохраняется при перезагрузке. Если вам нужно отключить параметры средства проверки драйверов для успешной загрузки,

установите точку останова в nt! VerifierInitSystem и используйте **команду !verifier -disable** на этом этапе.

**?**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Дополнительные сведения см. в разделе "Проверка драйверов".

# Замечания

В следующем примере показано, что в 64-разрядных версиях Windows журнал перехода IRQL не всегда завершен. Показанные два элемента являются последовательными записями в журнале обработчика 2. Первая запись показывает, что IRQL собирается от 2 до 0. Вторая запись показывает, что IRQL собирается от 2 до 2. Сведения о том, как IRQL поднялся с 0 до 2 отсутствует.

dbgcmd

```
Thread:             fffffa80068c9400
Old irql:           0000000000000002
New irql:           0000000000000000
Processor:          0000000000000002
Time stamp:         0000000000000857

    fffff8800140f12a ndis!ndisNsiGetInterfaceInformation+0x20a
    fffff88001509478 NETIO!NsiGetParameterEx+0x178
    fffff88005f062f2 nsiproxy!NsippGetParameter+0x24a
    fffff88005f086db nsiproxy!NsippDispatchDeviceControl+0xa3
    fffff88005f087a0 nsiproxy!NsippDispatch+0x48

Thread:             fffffa80068c9400
Old irql:           0000000000000002
New irql:           0000000000000002
Processor:          0000000000000002
Time stamp:         0000000000000857

    fffff8800140d48d ndis!ndisReferenceTopMiniportByNameForNsi+0x1ce
    fffff8800140f072 ndis!ndisNsiGetInterfaceInformation+0x152
    fffff88001509478 NETIO!NsiGetParameterEx+0x178
```

```
    fffff88005f062f2 nsiproxy!NsippGetParameter+0x24a
    fffff88005f086db nsiproxy!NsippDispatchDeviceControl+0xa3
```

Значения 4, 8 и 0x20, 0x40, 0x80 и 0x100 являются специальными значениями для *флагов*. Если эти значения используются, можно использовать специальные аргументы, перечисленные в **разделе "Параметры** ", и отображение будет содержать только сведения, связанные со значением этого флага.

Если используется любое другое значение флагов , *даже если задано одно или несколько этих битов* , разрешены только *аргументы Flags* и *Image* . В этой ситуации, помимо всех других отображаемых сведений, **средство проверки** будет отображать параметры средства проверки драйверов, активные, а также статистику выделения пула, вызывает IRQL, спин-блокировки и обрезки.

Если *флаги* равны 0x20, значения, указанные для *ЗавершенияTime*, *CancelTime* и *ForceCancellation* , используются параметром проверки зависания драйвера средства проверки драйверов. Эти новые значения вступают в силу немедленно и последняя до следующей загрузки. При перезагрузке они отменить изменения значения по умолчанию.

Кроме того, если *флаги* равны 0x20 (с дополнительными параметрами или без нее), журнал проверки зависания драйвера выводится. Дополнительные сведения о интерпретации журнала см. в разделе "Проверка зависания драйверов" документации по проверяющим драйверам в документации по комплекту драйверов Windows (WDK).

Ниже приведен пример **расширения !verifier** на компьютере с Windows 7.

dbgcmd

```
2: kd> !verifier 0xf

Verify Level 9bb ... enabled options are:
    Special pool
    Special irql
    All pool allocations checked on unload
    Io subsystem checking enabled
    Deadlock detection enabled
    DMA checking enabled
    Security checks enabled
    Miscellaneous checks enabled

Summary of All Verifier Statistics

RaiseIrqls                              0x0
AcquireSpinLocks                        0x362
Synch Executions                        0x0
```

```
Trims                                   0xa34a

Pool Allocations Attempted              0x7b058
Pool Allocations Succeeded              0x7b058
Pool Allocations Succeeded SpecialPool 0x7b058
Pool Allocations With NO TAG            0x0
Pool Allocations Failed                 0x0
Resource Allocations Failed Deliberately   0x0

Current paged pool allocations          0x1a for 00000950 bytes
Peak paged pool allocations             0x1b for 00000AC4 bytes
Current nonpaged pool allocations       0xe3 for 00046110 bytes
Peak nonpaged pool allocations          0x10f for 00048E40 bytes

Driver Verification List

Entry     State           NonPagedPool   PagedPool   Module

fffffa8003b6f670 Loaded           000000a0       00000854    videoprt.sys

Current Pool Allocations  00000002    00000013
Current Pool Bytes        000000a0    00000854
Peak Pool Allocations     00000006    00000014
Peak Pool Bytes           000008c0    000009c8

PoolAddress  SizeInBytes    Tag       CallersAddress
fffff9800157efc0     0x0000003c     Vprt      fffff88002c62963
fffff9800146afc0     0x00000034     Vprt      fffff88002c62963
fffff980015bafe0     0x00000018     Vprt      fffff88002c628f7
...

fffffa8003b6f620 Loaded           00046070       000000fc    usbport.sys

Current Pool Allocations  000000e1    00000007
Current Pool Bytes        00046070    000000fc
Peak Pool Allocations     0000010d    0000000a
Peak Pool Bytes           00048da0    00000254

PoolAddress  SizeInBytes    Tag       CallersAddress
fffff98003a38fc0     0x00000038     usbp      fffff88004215e34
fffff98003a2cfc0     0x00000038     usbp      fffff88004215e34
fffff9800415efc0     0x00000038     usbp      fffff88004215e34
...

-----------------------------------------------
Fault injection trace log
-----------------------------------------------

Driver Verifier didn't inject any faults.

-----------------------------------------------
Track irql trace log
-----------------------------------------------

Displaying most recent 0x0000000000000004 entries from the IRQL transition
```

```
log.
There are up to 0x100 entries in the log.

Thread:             fffff80002bf8c40
Old irql:           0000000000000002
New irql:           0000000000000002
Processor:          0000000000000000
Time stamp:         000000000000495e

    fffff8800420f2ca USBPORT!USBPORT_DM_IoTimerDpc+0x9a
    fffff80002a5b5bf nt!IopTimerDispatch+0x132
    fffff80002a7c29e nt!KiProcessTimerDpcTable+0x66
    fffff80002a7bdd6 nt!KiProcessExpiredTimerList+0xc6
    fffff80002a7c4be nt!KiTimerExpiration+0x1be

Thread:             fffff80002bf8c40
Old irql:           0000000000000002
New irql:           0000000000000002
Processor:          0000000000000000
Time stamp:         000000000000495e

    fffff88004205f3a USBPORT!USBPORT_AcquireEpListLock+0x2e
    fffff880042172df USBPORT!USBPORT_Core_TimeoutAllTransfers+0x1f
    fffff8800420f2ca USBPORT!USBPORT_DM_IoTimerDpc+0x9a
    fffff80002a5b5bf nt!IopTimerDispatch+0x132
    fffff80002a7c29e nt!KiProcessTimerDpcTable+0x66

Thread:             fffff80002bf8c40
Old irql:           0000000000000002
New irql:           0000000000000002
Processor:          0000000000000000
Time stamp:         000000000000495e

    fffff88004201694 USBPORT!MPf_CheckController+0x4c
    fffff8800420f26a USBPORT!USBPORT_DM_IoTimerDpc+0x3a
    fffff80002a5b5bf nt!IopTimerDispatch+0x132
    fffff80002a7c29e nt!KiProcessTimerDpcTable+0x66
    fffff80002a7bdd6 nt!KiProcessExpiredTimerList+0xc6

Thread:             fffff80002bf8c40
Old irql:           0000000000000002
New irql:           0000000000000002
Processor:          0000000000000000
Time stamp:         000000000000495e

    fffff8800420167c USBPORT!MPf_CheckController+0x34
    fffff8800420f26a USBPORT!USBPORT_DM_IoTimerDpc+0x3a
    fffff80002a5b5bf nt!IopTimerDispatch+0x132
    fffff80002a7c29e nt!KiProcessTimerDpcTable+0x66
    fffff80002a7bdd6 nt!KiProcessExpiredTimerList+0xc6
```

Ниже приведен пример **расширения !verifier** на компьютере Windows Vista с включенным битом 7 и *указанным адресом* .

dbgcmd

```
0: kd> !verifier 80 a2b1cf20
# Parsing 00004000 array entries, searching for address a2b1cf20.

Pool block a2b1ce98, Size 00000168, Thread a2b1ce98
808f1be6 ndis!ndisFreeToNPagedPool+0x39
808f11c1 ndis!ndisPplFree+0x47
808f100f ndis!NdisFreeNetBufferList+0x3b
8088db41 NETIO!NetioFreeNetBufferAndNetBufferList+0xe
8c588d68 tcpip!UdpEndSendMessages+0xdf
8c588cb5 tcpip!UdpSendMessagesDatagramsComplete+0x22
8088d622 NETIO!NetioDereferenceNetBufferListChain+0xcf
8c5954ea tcpip!FlSendNetBufferListChainComplete+0x1c
809b2370 ndis!ndisMSendCompleteNetBufferListsInternal+0x67
808f1781 ndis!NdisFSendNetBufferListsComplete+0x1a
8c04c68e pacer!PcFilterSendNetBufferListsComplete+0xb2
809b230c ndis!NdisMSendNetBufferListsComplete+0x70
# 8ac4a8ba test1!HandleCompletedTxPacket+0xea

Pool block a2b1ce98, Size 00000164, Thread a2b1ce98
822af87f nt!VerifierExAllocatePoolWithTagPriority+0x5d
808f1c88 ndis!ndisAllocateFromNPagedPool+0x1d
808f11f3 ndis!ndisPplAllocate+0x60
808f1257 ndis!NdisAllocateNetBufferList+0x26
80890933
NETIO!NetioAllocateAndReferenceNetBufferListNetBufferMdlAndData+0x14
8c5889c2 tcpip!UdpSendMessages+0x503
8c05c565 afd!AfdTLSendMessages+0x27
8c07a087 afd!AfdTLFastDgramSend+0x7d
8c079f82 afd!AfdFastDatagramSend+0x5ae
8c06f3ea afd!AfdFastIoDeviceControl+0x3c1
8217474f nt!IopXxxControlFile+0x268
821797a1 nt!NtDeviceIoControlFile+0x2a
8204d16a nt!KiFastCallEntry+0x127
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Vm

Статья • 05.04.2024

Расширение **!vm** отображает сводную информацию об использовании виртуальной памяти статистикой в целевой системе.

dbgcmd

```
!vm [Flags]
```

## Параметры

*Flags*
Указывает, какие сведения будут отображаться в выходных данных этой команды. Это может быть любая сумма следующих битов. Значение по умолчанию равно 0, что приводит к включению статистики виртуальной памяти на уровне системы, а также статистики памяти для каждого процесса.

Bit 0 (0x1)
Приводит к опущению статистики, относящейся к процессу.

Бит 1 (0x2)
Приводит к включению стека потоков управления памятью.

Бит 2 (0x4)
Приводит к включению использования памяти сервера терминала.

Бит 3 (0x8)
Приводит к включению журнала записи файла страницы.

Бит 4 (0x10)
Приводит к включению стека потоков владельца рабочего набора.

Бит 5 (0x20)
(Windows Vista и более поздние версии) Приводит к включению использования виртуального адреса ядра.

## Среда

**Режимы**: только режим ядра

## DLL-библиотеки

Kdexts.dll

# Дополнительная информация

**Команда расширения !memusage** может использоваться для анализа использования физической памяти. Дополнительные сведения об управлении памятью см. в разделе "*Внутренние* компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

# Замечания

Ниже приведен пример коротких выходных данных, создаваемых при *выполнении флагов* 1:

dbgcmd

```
kd> !vm 1

*** Virtual Memory Usage ***
      Physical Memory:     16270   (   65080 Kb)
      Page File: \??\E:\pagefile.sys
         Current:     98304Kb Free Space:     61044Kb
 Minimum:     98304Kb Maximum:       196608Kb
      Available Pages:      5543   (   22172 Kb)
      ResAvail Pages:       6759   (   27036 Kb)
      Locked IO Pages:       112   (     448 Kb)
 Free System PTEs:    45089   (  180356 Kb)
      Free NP PTEs:         5145   (   20580 Kb)
      Free Special NP:       336   (    1344 Kb)
      Modified Pages:        714   (    2856 Kb)
      NonPagedPool Usage:    877   (    3508 Kb)
      NonPagedPool Max:     6252   (   25008 Kb)
      PagedPool 0 Usage:     729   (    2916 Kb)
      PagedPool 1 Usage:     432   (    1728 Kb)
      PagedPool 2 Usage:     436   (    1744 Kb)
      PagedPool Usage:      1597   (    6388 Kb)
      PagedPool Maximum:   13312   (   53248 Kb)
      Shared Commit:        1097   (    4388 Kb)
      Special Pool:          229   (     916 Kb)
      Shared Process:       1956   (    7824 Kb)
      PagedPool Commit:     1597   (    6388 Kb)
      Driver Commit:         828   (    3312 Kb)
      Committed pages:     21949   (   87796 Kb)
      Commit limit:        36256   (  145024 Kb)
```

Все использование памяти отображается на страницах и в килобайтах. Наиболее полезными сведениями в этом отображении являются следующие:

Развернуть таблицу

| Параметр | Значение |
| --- | --- |
| физическая память | Общая физическая память в системе. |
| доступные страницы | Количество страниц памяти, доступных в системе, как виртуальной, так и физической. |
| Использование непагированного пула | Объем страниц, выделенных для непагированного пула. Нераспакованный пул — это память, которая не может быть переключена на файл разбиения на страницы, поэтому она всегда должна занимать физическую память. Если это число слишком велико, это обычно указывает на то, что утечка памяти в системе возникает. |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !vpb

Статья • 05.04.2024

Расширение **!vpb** отображает блок параметров тома (VPB).

```
dbgcmd
```

```
!vpb Address
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес VPB.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о виртуальных машинах см. в документации по комплекту драйверов Windows (WDK) и *внутренних компонентах* Microsoft Windows, марку Руссиновичу и Дэвиду Соломону.

## Замечания

Ниже приведен пример. Во-первых, дерево устройства отображается с расширением **!devnode** :

```
dbgcmd
```

```
kd> !devnode 0 1
Dumping IopRootDeviceNode (= 0x80e203b8)
DevNode 0x80e203b8 for PDO 0x80e204f8
  InstancePath is "HTREE\ROOT\0"
  State = DeviceNodeStarted (0x308)
  Previous State = DeviceNodeEnumerateCompletion (0x30d)
  DevNode 0x80e56dc8 for PDO 0x80e56f18
    InstancePath is "Root\dmio\0000"
    ServiceName is "dmio"
    State = DeviceNodeStarted (0x308)
    Previous State = DeviceNodeEnumerateCompletion (0x30d)
```

```
  DevNode 0x80e56ae8 for PDO 0x80e56c38
    InstancePath is "Root\ftdisk\0000"
    ServiceName is "ftdisk"
    State = DeviceNodeStarted (0x308)
    Previous State = DeviceNodeEnumerateCompletion (0x30d)
 DevNode 0x80e152a0 for PDO 0x80e15cb8
      InstancePath is
"STORAGE\Volume\1&30a96598&0&Signature5C34D70COffset7E00Length60170A00"
      ServiceName is "VolSnap"
      TargetDeviceNotify List - f 0xe1250938  b 0xe14b9198
      State = DeviceNodeStarted (0x308)
      Previous State = DeviceNodeEnumerateCompletion (0x30d)
    .....
```

Последний узел устройства указан в томе. Проверьте объект физического устройства (PDO) с расширением **!devobj** :

dbgcmd

```
kd> !devobj 80e15cb8
Device object (80e15cb8) is for:
 HarddiskVolume1 \Driver\Ftdisk DriverObject 80e4e248
Current Irp 00000000 RefCount 14 Type 00000007 Flags 00001050
Vpb 80e15c30 DevExt 80e15d70 DevObjExt 80e15e40 Dope 80e15bd8 DevNode
80e152a0
ExtensionFlags (0000000000)
AttachedDevice (Upper) 80e14c60 \Driver\VolSnap
Device queue is not busy.
```

Адрес VPB этого устройства включен в этот список. Используйте этот адрес с расширением **!vpb** :

dbgcmd

```
kd> !vpb 80e15c30
Vpb at 0x80e15c30
Flags: 0x1 mounted
DeviceObject: 0x80de5020
RealDevice:   0x80e15cb8
RefCount: 14
Volume Label:           MY-DISK-C
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !vtop

Статья • 05.04.2024

**Расширение !vtop** преобразует виртуальный адрес в соответствующий физический адрес и отображает другие сведения о таблице страниц и каталоге страниц.

Синтаксис

```
dbgcmd
```

```
!vtop PFN VirtualAddress
!vtop 0 VirtualAddress
```

## Параметры

*DirBase*
Указывает базу каталогов для процесса. Каждый процесс имеет собственное виртуальное адресное пространство. **Используйте расширение !process**, чтобы определить базу каталогов для процесса.

*PFN*
Указывает номер кадра страницы (PFN) базы каталога для процесса.

**0**
Заставляет **!vtop** использовать текущий контекст процесса для преобразования адресов.

*VirtualAddress*
Указывает виртуальный адрес, страница которого является требуемой.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Другие методы достижения этих результатов см. в разделе "Преобразование виртуальных адресов в физические адреса". Также см. !ptov**.** Сведения о таблицах страниц и каталогах *страниц см. в разделе* "Внутренние компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

# Замечания

Чтобы использовать эту команду, сначала используйте **расширение !process** для определения базы каталогов процесса. Номер кадра страницы (PFN) этой базы каталогов можно найти, удалив три конечных шестнадцатеричных нуля (другими словами, путем перемещения числа 12 бит).

Рассмотрим пример:

dbgcmd

```
kd> !process 0 0
**** NT ACTIVE PROCESS DUMP ****
....
PROCESS ff779190  SessionId: 0  Cid: 04fc    Peb: 7ffdf000  ParentCid: 0394
 DirBase: 098fd000  ObjectTable: e1646b30  TableSize:   8.
    Image: MyApp.exe
```

Так как база каталогов 0x098FD000, ее PFN 0x098FD.

dbgcmd

```
kd> !vtop 98fd 12f980
Pdi 0 Pti 12f
0012f980 09de9000 pfn(09de9)
```

Обратите внимание, что конечные три нуля являются необязательными. Расширение **!vtop** отображает индекс каталога страницы (PDI), индекс таблицы страницы (PTI), виртуальный адрес, который вы изначально ввели, физический адрес начала физической страницы и номер кадра страницы (PFN) записи таблицы страницы (PTE).

Если вы хотите преобразовать виртуальный адрес 0x0012F980 в физический адрес, необходимо просто взять последние три шестнадцатеричные цифры (0x980) и добавить их в физический адрес начала страницы (0x09DE9000). Это дает физический адрес 0x09DE9980.

Если вы забыли удалить три нуля и передайте полную базу **каталогов на !vtop** вместо PFN, результаты обычно будут правильными. Это связано с тем, что если **!vtop** получает слишком большое число, чтобы быть PFN, он справа сдвигает его двенадцать битов и использует это число вместо этого:

dbgcmd

```
kd> !vtop 98fd 12f980
Pdi 0 Pti 12f
0012f980 09de9000 pfn(09de9)

kd> !vtop 98fd000 12f980
Pdi 0 Pti 12f
0012f980 09de9000 pfn(09de9)
```

Однако лучше всегда использовать PFN, так как некоторые базовые значения каталога не будут преобразованы таким образом.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdmaud

Статья • 05.04.2024

!wdmaud устарел.

```
dbgcmd
```

```
!wdmaud Address Flags
```

## Параметры

*Адрес*
Указывает адрес отображаемой структуры.

*Flags*
Указывает отображаемые сведения. Это должно включать именно один из битов 0x1, 0x2, 0x4 и 0x8. 0x100 бит можно добавить к любому из них.

Bit 0 (0x1)
Отображает список всех операций ввода-вывода, отправленных в wdmaud.sys. При использовании *адрес* должен указать адрес **WdmaIoctlHistoryListHead**. Если задана 0x100 бит, отображение также включает **pContext** , с которым был отправлен каждый IOCTL.

Бит 1 (0x2)
Отображает список всех irps, помеченных WDMAud как ожидающие. При использовании *адрес* должен указать адрес **WdmaPendingIrpListHead**. Если задан 0x100 бит, отображение также включает контекст, для которого был выделен каждый IRP.

Бит 2 (0x4)
Отображает список всех многомерных выражений, выделенных WDMAud. При использовании *адрес* должен указать адрес **WdmaAllocatedMdlListHead**. Если задан 0x100 бит, отображение также включает контекст, для которого был выделен каждый MDL.

Бит 3 (0x8)
Отображает список всех активных контекстов, подключенных к wdmaud.sys. При использовании *адрес* должен указать адрес **WdmaContextListHead**. Если задан 0x100 бит, отображение также включает элементы данных каждой структуры контекста.

Бит 8 (0x100)
Приводит к тому, что отображение содержит подробные сведения.

## DLL-библиотеки

Windows XP и более поздних версий — недоступно

## Дополнительная информация

Сведения об архитектуре звука WDM и звуковых драйверах см. в документации по комплекту драйверов Windows (WDK).

## Замечания

Контексты, подключенные к wdmaud.sys (**pContext**), содержат большую часть данных состояния для каждого устройства. Когда wdmaud.drv загружается в новый процесс, wdmaud.sys уведомляется о его поступлении. При выгрузке wdmaud.drv wdmaud.sys очищает все выделения, сделанные в этом контексте.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !whattime

Статья • 08.07.2024

Расширение **!whattime** преобразует число галок в стандартное значение времени.

```
dbgcmd
```

```
!whattime Ticks
```

## Параметры

*Клещей*
Количество галок.

## DLL-библиотеки

Kdexts.dll

## Замечания

Выходные данные отображаются как *HH:MM:SS.mmm*. Рассмотрим пример:

```
dbgcmd
```

```
kd> !whattime 29857ae4
696613604 Ticks in Standard Time:  15:02:16.040s
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !whatperftime

Статья • 05.04.2024

**Расширение !whatperftime** преобразует значение счетчика производительности высокого разрешения в стандартное значение времени.

```
dbgcmd
```

```
!whatperftime Count
```

## Параметры

*Численность*
Значение счетчика производительности.

## DLL-библиотеки

Kdexts.dll

## Замечания

Вы можете использовать **!whatperftime для преобразования значений, полученных путем вызова QueryPerformanceCounter**. Значения времени счетчика производительности также находятся в трассировках программного обеспечения.

Выходные данные отображаются как *HH:MM:SS.mmm*. Рассмотрим пример:

```
dbgcmd
```

```
kd> !whatperftime 304589
3163529 Performance Counter in Standard Time: .004.313s
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !whea

Статья • 05.04.2024

Расширение **!whea** отображает сведения об архитектуре ошибок оборудования Windows верхнего уровня (WHEA).

dbgcmd

```
!whea
```

## DLL-библиотеки

Windows Vista и более поздние версии — Kdexts.dll

## Дополнительная информация

**Расширения !errrec** и **!errpkt** можно использовать для отображения дополнительных сведений о WHEA. Общие сведения об WHEA см . в документации по архитектуре аппаратных ошибок Windows (WHEA) в комплекте драйверов Windows (WDK).

## Замечания

В следующем примере показаны (усеченные) выходные данные **расширения !whea** :

dbgcmd

```
3: kd> !whea
Error Source Table @ fffff800019ca250
13 Error Sources
Error Source 0 @ fffffa80064132c0
   Notify Type      : Machine Check Exception
   Type             : 0x0 (MCE)
   Error Count      : 8
   Record Count     : 10
   Record Length    : c3e
   Error Records    : wrapper @ fffffa8007cf4000  record @ fffffa8007cf4030
                    : wrapper @ fffffa8007cf4c3e  record @ fffffa8007cf4c6e
                    : wrapper @ fffffa8007cf587c  record @ fffffa8007cf58ac
                    : wrapper @ fffffa8007cf64ba  record @ fffffa8007cf64ea
                    : wrapper @ fffffa8007cf70f8  record @ fffffa8007cf7128
                    : wrapper @ fffffa8007cf7d36  record @ fffffa8007cf7d66
```

```
                    : wrapper @ fffffa8007cf8974  record @ fffffa8007cf89a4
                    : wrapper @ fffffa8007cf95b2  record @ fffffa8007cf95e2
                    : wrapper @ fffffa8007cfa1f0  record @ fffffa8007cfa220
                    : wrapper @ fffffa8007cfae2e  record @ fffffa8007cfae5e
                    : wrapper @ fffffa8007cfba6c  record @ fffffa8007cfba9c
                    : wrapper @ fffffa8007cfc6aa  record @ fffffa8007cfc6da
                    : wrapper @ fffffa8007cfd2e8  record @ fffffa8007cfd318
                    : wrapper @ fffffa8007cfdf26  record @ fffffa8007cfdf56
                    : wrapper @ fffffa8007cfeb64  record @ fffffa8007cfeb94
                    : wrapper @ fffffa8007cff7a2  record @ fffffa8007cff7d2
   Descriptor       : @ fffffa8006413328
      Length                     : 3cc
      Max Raw Data Length        : 392
      Num Records To Preallocate : 10
      Max Sections Per Record    : 3
      Error Source ID            : 0
      Flags                      : 00000000
Error Source 1 @ fffffa8007d00bc0
   Notify Type      : Corrected Machine Check
   Type             : 0x1 (CMC)
   Error Count      : 0
   Record Count     : 10
   Record Length    : c3e
   Error Records    : wrapper @ fffffa8007d01000  record @ fffffa8007d01030
                    : wrapper @ fffffa8007d01c3e  record @ fffffa8007d01c6e
                    : wrapper @ fffffa8007d0287c  record @ fffffa8007d028ac
                    : wrapper @ fffffa8007d034ba  record @ fffffa8007d034ea
                    : wrapper @ fffffa8007d040f8  record @ fffffa8007d04128
                    : wrapper @ fffffa8007d04d36  record @ fffffa8007d04d66
                    : wrapper @ fffffa8007d05974  record @ fffffa8007d059a4
                    : wrapper @ fffffa8007d065b2  record @ fffffa8007d065e2
                    : wrapper @ fffffa8007d071f0  record @ fffffa8007d07220
                    : wrapper @ fffffa8007d07e2e  record @ fffffa8007d07e5e
                    : wrapper @ fffffa8007d08a6c  record @ fffffa8007d08a9c
                    : wrapper @ fffffa8007d096aa  record @ fffffa8007d096da
                    : wrapper @ fffffa8007d0a2e8  record @ fffffa8007d0a318
                    : wrapper @ fffffa8007d0af26  record @ fffffa8007d0af56
                    : wrapper @ fffffa8007d0bb64  record @ fffffa8007d0bb94
                    : wrapper @ fffffa8007d0c7a2  record @ fffffa8007d0c7d2
   Descriptor       : @ fffffa8007d00c28
      Length                     : 3cc
      Max Raw Data Length        : 392
      Num Records To Preallocate : 10
      Max Sections Per Record    : 3
      Error Source ID            : 1
      Flags                      : 00000000
Error Source 2 @ fffffa8007d00770
   Notify Type      : Non-Maskable Interrupt
   Type             : 0x3 (NMI)
   Error Count      : 1
   Record Count     : 1
   Record Length    : 1f8
   Error Records    : wrapper @ fffffa800631b2c0  record @ fffffa800631b2f0
   Descriptor       : @ fffffa8007d007d8
      Length                     : 3cc
```

```
      Max Raw Data Length        : 100
      Num Records To Preallocate : 1
      Max Sections Per Record    : 1
      Error Source ID            : 2
      Flags                      : 00000000
Error Source 3 @ fffffa8007d0dbc0
   Notify Type      : BOOT Error Record
   Type             : 0x7 (BOOT)
   Error Count      : 0
   Record Count     : 1
   Record Length    : 4f8
   Error Records    : wrapper @ 0000000000000000  record @ 0000000000000030
   Descriptor       : @ fffffa8007d0dc28
      Length                     : 3cc
      Max Raw Data Length        : 400
      Num Records To Preallocate : 1
      Max Sections Per Record    : 1
      Error Source ID            : 3
      Flags                      : 00000000

. . . .
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wsle

Статья • 05.04.2024

Расширение **!wsle** отображает все записи списка рабочих наборов (WSLEs).

Синтаксис

dbgcmd

```
!wsle [Flags [Address]]
```

# Параметры

*Flags*
Указывает сведения, которые необходимо включить в дисплей. Это может быть любое сочетание следующих битов. По умолчанию используется значение 0. Если это используется, отображаются только основные сведения о рабочем наборе.

Bit 0 (0x1)
Приводит к включению сведений о каждом адресе WSLE, возрасте, состоянии блокировки и подсчете ссылок. Если в WSLE есть недопустимая запись таблицы страницы (PTE) или запись каталога страницы (PDE), это также отображается.

Бит 1 (0x2)
Вызывает отображение, включаемое общее количество допустимых WSLEs, индекс последнего WSLE и индекс первого свободного WSLE.

Бит 2 (0x4)
Приводит к тому, что отображение включает общее количество свободных WSLEs, а также индекс каждого свободного WSLE. Если бит 1 также задан, то проверка выполняется, чтобы убедиться, что количество бесплатных WSLEs плюс количество допустимых WSLEs фактически равно общему количеству WSLEs.

*Адрес*
Указывает адрес списка рабочих наборов. Если это опущено, используется список рабочих наборов по умолчанию. Указание нуля для *адреса* совпадает с опущением.

# DLL-библиотеки

Kdexts.dll

# Дополнительная информация

Сведения о рабочих наборах *см. в разделе* "Внутренние компоненты Microsoft Windows", Марк Руссинович и Дэвид Соломон.

# Замечания

Это расширение может занять значительное время выполнения.

Ниже приведен пример на целевом компьютере x86 под управлением Windows Server 2003:

dbgcmd

```
kd> !wsle 3

Working Set @ c0503000
    FirstFree:       a7  FirstDynamic:          4
    LastEntry      23d  NextSlot:         4  LastInitialized      259
    NonDirect       65  HashTable:        0  HashTableSize:         0

Reading the WSLE data...

Virtual Address           Age  Locked  ReferenceCount
        c0300203          0        1        1
        c0301203          0        1        1
        c0502203          0        1        1
        c0503203          0        1        1
        c01ff201          0        0        1
        77f74d19          3        0        1
        7ffdfa01          2        0        1
        c0001201          0        0        1

.....

Reading the WSLE data...
Valid WSLE entries = 0xa7
found end @ wsle index 0x259

.....
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Зомби

Статья • 05.04.2024

Расширение **!зомби** отображает все мертвые ("зомби") процессы или потоки.

```
dbgcmd
```

```
!zombies [Flags [RestartAddress]]
```

## Параметры

*Flags*
Указывает, что будет отображаться. Возможные значения включают:

1
Отображает все зомби-процессы. (Это значение по умолчанию.)

2
Отображает все зомби-потоки.

*ПерезапускAddress*
Указывает шестнадцатеричный адрес, с которого начинается поиск. Это полезно, если предыдущий поиск был завершен преждевременно. По умолчанию используется значение 0.

## DLL-библиотеки

Windows XP и более поздних версий — недоступно

## Дополнительная информация

Это расширение устарело.

Чтобы просмотреть список всех процессов и потоков, используйте **расширение !process** .

Общие сведения о процессах и потоках в режиме ядра см. в разделе "Изменение контекстов". Дополнительные сведения об анализе процессов и потоков см. в разделе "*Внутренние* компоненты Microsoft Windows", марк Руссинович и Дэвид Соломон.

## Замечания

Зомби-процессы являются мертвыми процессами, которые еще не были удалены из списка процессов. Зомби-потоки аналогичны.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# User-Mode Extensions

18.07.2025

В этом разделе ссылки описываются команды расширения, которые в основном используются во время отладки в пользовательском режиме.

Отладчик автоматически загружает правильную версию этих команд расширения. Если вы вручную не загружали другую версию, вам не нужно отслеживать используемые версии DLL. Сведения об описании порядка поиска модуля по умолчанию см. в командах расширения отладчика . Сведения о том, как загружать модули расширения, см. в статье о загрузке dll расширений отладчика .

На каждой странице ссылки на команду расширения перечислены библиотеки DLL, предоставляющие следующую команду. Используйте следующие правила, чтобы определить правильный каталог, из которого необходимо загрузить библиотеку DLL этого расширения:

- Если целевое приложение работает в Windows XP или более поздней версии Windows, используйте winxp\Ntsdexts.dll.

Кроме того, расширения в пользовательском режиме, которые не относятся к одной операционной системе, можно найти в winext\Uext.dll.

# !avrf

Статья • 05.04.2024

Расширение **!avrf** управляет параметрами средства проверки приложений и отображает различные выходные данные, созданные средством проверки приложений.

dbgcmd

```
!avrf
!avrf -vs { Length | -a Address }
!avrf -hp { Length | -a Address }
!avrf -cs { Length | -a Address }
!avrf -dlls [ Length ]
!avrf -trm
!avrf -ex [ Length ]
!avrf -threads [ ThreadID ]
!avrf -tp [ ThreadID ]
!avrf -srw  [ Address | Address Length ] [ -stats ]
!avrf -leak  [ -m ModuleName] [ -r ResourceType] [ -a Address ] [ -t ]
!avrf -trace TraceIndex
!avrf -cnt
!avrf -brk [BreakEventType]
!avrf -flt [EventType Probability]
!avrf -flt break EventType
!avrf -flt stacks Length
!avrf -trg [ Start End | dll Module | all ]
!avrf -settings
!avrf -skp [ Start End | dll Module | all | Time ]
```

## Параметры

**-vs** {*Length*| **-a***Address*}
Отображает журнал операций виртуального пространства. *Длина* указывает количество отображаемых записей, начиная с последней. *Адрес* указывает виртуальный адрес. Отображаются записи виртуальных операций, содержащих этот виртуальный адрес.

**-hp** {*Length*| **-a***Address*}
Отображает журнал операций кучи. *Адрес* указывает адрес кучи. Отображаются записи операций кучи, содержащих этот адрес кучи.

**-cs** {*Length*| **-a***Address*}
Отображает журнал удаления критического раздела. *Длина* указывает количество отображаемых записей, начиная с последней. *Адрес* указывает адрес критического

раздела. Записи для определенного критического раздела отображаются при *указании адреса* .

**-dlls** [*длина*]
Отображает журнал загрузки и выгрузки библиотеки DLL. *Длина* указывает количество отображаемых записей, начиная с последней.

**-Trm**
Отображает журнал всех завершенных и приостановленных потоков.

**-ex** [*длина*]
Отображает журнал исключений. Средство проверки приложений отслеживает все исключения в приложении.

**-threads** [*ThreadID*]
Отображает сведения о потоках в целевом процессе. Для дочерних потоков также отображаются размеры стека и **флаги CreateThread** , указанные родительским элементом. Если указать идентификатор потока, отображаются сведения только для этого потока.

**-tp** [ThreadID*]*
Отображает журнал threadpool. Этот журнал содержит трассировки стека для различных операций, таких как изменение маски сходства потоков, изменение приоритета потока, публикация сообщений потоков и инициализация или неинициализация COM из обратного вызова threadpool. Если указать идентификатор потока, отображаются сведения только для этого потока.

**-srw** [*длина **адреса***|] [ **-stats** ]
Отображает журнал средства чтения и записи (SRW). Если указать *адрес*, отображаются записи для блокировки SRW по указанному адресу. Если указать *адрес* и *длину*, отображаются записи для блокировок SRW в этом диапазоне адресов. Если включить **параметр -stats** , отобразится статистика блокировки SRW.

**-leak** [ **-m***ModuleName*] [ **-r***ResourceType*] [ **-a***Address*] [ **-t** ]
Отображает журнал невыполненных ресурсов. Эти ресурсы могут быть утечками в любой момент. Если указать *имя* модуля (включая расширение), отображаются все выдающиеся ресурсы в указанном модуле. При указании *ResourceType* отображаются все выдающиеся ресурсы этого типа ресурса. Если указать *адрес*, отображаются записи невыполненных ресурсов с этим адресом. *ResourceType* может быть одним из следующих вариантов:

Куча: отображение выделения кучи с помощью API-интерфейсов Кучи Win32

Local: отображает локальные или глобальные выделения

CRT: отображает выделения с помощью API CRT

Virtual: Отображает виртуальные резервирования

BSTR: отображает выделения BSTR

Реестр: откроется раздел реестра

Power: отображает объекты power notification

Дескриптор: отображает выделения потоков, файлов и событий

**-trace***TraceIndex* отображает трассировку стека для указанного индекса трассировки трассировки. Некоторые структуры используют этот 16-разрядный номер индекса для идентификации трассировки стека. Этот индекс указывает на расположение в базе данных трассировки стека.

**-cnt** отображает список глобальных счетчиков.

**-brk** [*BreakEventType*] Указывает событие разрыва. *BreakEventType* — это тип события разрыва. Для списка возможных типов и списка текущих параметров события останова введите **!avrf -brk**.

**-flt** [*Вероятность* **eventType**] Указывает внедрение ошибки. *EventType* — это тип события. *Вероятность — это частота* сбоя события. Это может быть любое целое число от 0 до 1000 000 (0xF4240). Если ввести **!avrf -flt** без дополнительных параметров, отображаются текущие параметры внедрения ошибок.

-**Flt break***EventType приводит к тому, что средство проверки приложений прерывается в отладчик каждый раз, когда этот сбой, указанный в EventType*, внедряется.

**-flt stacks***Length Отображает количество* трассировок стека для последних операций, внедренных сбоем.

**-trg** [*начальный*| **модуль *dll*| все**] — указывает целевой диапазон. *Начало* — это начальный адрес целевого диапазона. *Конец* — конечный адрес целевого диапазона. *Модуль* задает имя (включая расширение .exe или .dll, но не включая путь) целевого модуля. Если ввести **-trg all**, все целевые диапазоны сбрасываются. При вводе **-trg** без дополнительных параметров отображаются текущие целевые диапазоны.

**-skp** [начальный| ***модуль dll*| *все* |*Время*] Указывает диапазон исключений. *Начало* — это начальный адрес диапазона исключений. *Конец* — это конечный адрес диапазона исключений. Модуль задает имя модуля, предназначенного или

исключенного. *Модуль* задает имя (включая расширение .exe или .dll, но не включая путь) модуля, который необходимо исключить. При вводе **-skp все** целевые диапазоны или диапазоны исключений сбрасываются. При вводе*значения времени* все ошибки подавляются в *миллисекундах времени* после возобновления выполнения.

## DLL-библиотеки

exts.dll

## Дополнительная информация

Сведения о том, как скачать и установить средство проверки приложений и ее документацию, см. в разделе "Проверка приложений".

## Замечания

**Если расширение !avrf** используется без параметров, оно отображает текущие параметры проверки приложений. **Если включен параметр "Полная куча" или "Быстрая заполнение кучи**", отображаются также сведения о активных кучах страниц. Некоторые примеры см. в разделе "Журналы операций кучи" в остановках средства проверки приложений отладки.

Если произошла остановка средства проверки приложений, **расширение !avrf** без параметров покажет характер остановки и его причины. Примеры см. в разделе "Отладка остановок проверяющего приложения".

Если отсутствуют символы для ntdll.dll и verifier.dll, **расширение !avrf** создает сообщение об ошибке. Сведения о том, как устранить эту проблему, см. в разделе "Установка и настройка отладчика" в остановках средства проверки приложений отладки.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !critsec

Статья • 05.04.2024

Расширение **!critsec** отображает критически важный раздел.

```
dbgsyntax
```

```
!critsec Address
```

## Параметры

*Адрес*

Указывает шестнадцатеричный адрес критического раздела.

### DLL-библиотеки

Ntsdexts.dll

## Дополнительная информация

Другие команды и расширения, которые могут отображать критически важные сведения о разделе, см. в разделе "Отображение критического раздела". Дополнительные сведения о критически важных разделах см. в документации по пакету SDK для Microsoft Windows, документации по комплекту драйверов Windows (WDK) и *внутренним* службам Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Если адрес критического раздела не известен, следует использовать расширение !ntsdexts.locks**.** При этом отображаются все критические разделы, инициализированные путем вызова **RtlInitializeCriticalSection**.

Рассмотрим пример:

```
dbgcmd
```

```
0:000> !critsec 3a8c0e9c
```

```
CritSec +3a8c0e9c at 3A8C0E9C
LockCount          1
RecursionCount     1
OwningThread       99
EntryCount         472
ContentionCount    1
*** Locked
```

## См. также

Отображение критического раздела

Критические истечение времени ожидания разделов (режим пользователя)

!Cs

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !dp (!ntsdexts.dp)

Статья • 08.07.2024

Расширение **!dp** в Ntsdexts.dll отображает процесс CSR.

Эта команда расширения не должна путаться с командой dp (Display Memory) **или с командой расширения !kdext*.dp**.

dbgcmd

```
!dp [v] [ PID | CSR-Process ]
```

## Параметры

**v**
Режим подробного вывода. Приводит к тому, что отображение включает структуру и список потоков.

*PiD*
Указывает идентификатор процесса процесса процесса CSR.

*CSR-Process*
Указывает шестнадцатеричный адрес процесса CSR.

## DLL-библиотеки

Ntsdexts.dll

## Замечания

Это расширение отображает адрес процесса, идентификатор процесса, порядковый номер, флаги и счетчик ссылок. Если выбран подробный режим, отображаются дополнительные сведения, а сведения о потоке отображаются для каждого процесса.

Если процесс не указан, отображаются все процессы.

## См. также

**!dt**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !dreg

Статья • 05.04.2024

Расширение **!dreg** отображает сведения о реестре.

```
dbgcmd
```

```
!dreg [-d|-w] KeyPath[!Value]
!dreg
```

# Параметры

**-d**
Приводит к отображению двоичных значений в виде DWORD.

**-w**
Приводит к отображению двоичных значений как WORD.

*KeyPath*
Указывает путь к разделу реестра. Он может начинаться с любого из следующих сокращений:

**Hklm**
HKEY_LOCAL_MACHINE

**Hkcu**
HKEY_CURRENT_USER

**hkcr**
HKEY_CLASSES_ROOT

**hku**
HKEY_USERS

Если аббревиация не используется, предполагается HKEY_LOCAL_MACHINE.

*Value*
Указывает имя отображаемого значения реестра. Если используется звездочка (*), отображаются все значения. Если *значение* опущено, отображаются все вложенные ключи.

# DLL-библиотеки

# Дополнительная информация

Дополнительные сведения о реестре см. в документации по комплекту драйверов Windows (WDK) и *внутренних элементах* Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном.

# Замечания

**Расширение !dreg** можно использовать для отображения реестра во время отладки в пользовательском режиме.

Он наиболее полезен во время удаленной отладки, так как позволяет просматривать реестр удаленного компьютера. Это также полезно при управлении отладчиком пользовательского режима из отладчика ядра, так как не удается запустить стандартный редактор реестра на целевом компьютере при его заморожении. (Можно использовать **. Команда спящего режима** для этой цели, а также. Дополнительные сведения см. в разделе "Управление отладчиком пользовательского режима" из отладчика ядра.)

Это также полезно при локальной отладке, так как информация представлена в легко удобочитаемом формате.

Если **!dreg** используется во время отладки в режиме ядра, результаты будут отображаться для хост-компьютера, а не целевого компьютера. Чтобы отобразить необработанные сведения о реестре для целевого компьютера, используйте **вместо этого расширение !reg** .

Ниже приведено несколько примеров. Ниже отображаются все подразделы указанного раздела реестра:

```
dbgcmd
```

```
!dreg hkcu\Software\Microsoft
```

Следующие значения будут отображаться в указанном разделе реестра:

```
dbgcmd
```

```
!dreg System\CurrentControlSet\Services\Tcpip!*
```

В следующем разделе реестра отобразится значение Start в указанном разделе реестра:

```
dbgcmd
```

```
!dreg System\CurrentControlSet\Services\Tcpip!Start
```

Ввод **!dreg** без каких-либо аргументов будет отображать краткий текст справки для этого расширения в окне команды отладчика.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Dt

Статья • 05.04.2024

Расширение **!dt** отображает сведения о потоке CSR.

Эта команда расширения не должна путаться с командой **dt (display Type).**

dbgcmd

```
!dt [v] CSR-Thread
```

## Параметры

**v**
Режим подробного вывода.

*CSR-Thread*
Указывает шестнадцатеричный адрес потока CSR.

## DLL-библиотеки

Ntsdexts.dll

## Замечания

Это расширение отображает поток, процесс, идентификатор клиента, флаги и количество ссылок, связанных с потоком CSR. Если выбран подробный режим, отображение также будет включать указатели списка, дескриптор потока и блок ожидания.

## См. также

**!dp (!ntsdexts.dp)**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !findstack

Статья • 05.04.2024

Расширение **!findstack** находит все стеки, содержащие указанный символ или модуль.

dbgcmd

```
!findstack Symbol [DisplayLevel]
!findstack -?
```

## Параметры

*Символ*
Задает символ или модуль.

*DisplayLevel*
Указывает, что должен содержать дисплей. Это может быть любое из следующих значений. Значение по умолчанию равно 1.

**0**
Отображает только идентификатор потока для каждого потока, содержащего *символ*.

**1**
Отображает идентификатор потока и кадр для каждого потока, содержащего *символ*.

**2**
Отображает весь стек потоков для каждого потока, содержащего *символ*.

**-?**
Отображает краткий текст справки для этого расширения в окне команды отладчика.

## DLL-библиотеки

Uext.dll

## Дополнительная информация

Дополнительные сведения о трассировках стека см. в **командах k, kb, kc, kd, kp, kP, kv (Display Stack Backtrace).**

# Замечания

Расширение **режима ядра !stacks** также отображает сведения о стеках, включая краткую сводку о состоянии каждого потока.

Ниже приведены некоторые примеры выходных данных из этого расширения:

dbgcmd

```
0:023> !uext.findstack wininet
Thread 009, 2 frame(s) match
        * 06 03eaffac 771d9263 wininet!ICAsyncThread::SelectThread+0x22a
        * 07 03eaffb4 7c80b50b
wininet!ICAsyncThread::SelectThreadWrapper+0xd

Thread 011, 2 frame(s) match
        * 04 03f6ffb0 771cda1d
wininet!AUTO_PROXY_DLLS::DoThreadProcessing+0xa1
        * 05 03f6ffb4 7c80b50b wininet!AutoProxyThreadFunc+0xb

Thread 020, 6 frame(s) match
        * 18 090dfde8 771db73a wininet!CheckForNoNetOverride+0x9c
        * 19 090dfe18 771c5e4d wininet!InternetAutodialIfNotLocalHost+0x220
        * 20 090dfe8c 771c5d6a wininet!ParseUrlForHttp_Fsm+0x135
        * 21 090dfe98 771bcb2c wininet!CFsm_ParseUrlForHttp::RunSM+0x2b
        * 22 090dfeb0 771d734a wininet!CFsm::Run+0x39
        * 23 090dfee0 77f6ad84 wininet!CFsm::RunWorkItem+0x79

Thread 023, 9 frame(s) match
        * 16 0bd4fe00 771bd256 wininet!ICSocket::Connect_Start+0x17e
        * 17 0bd4fe0c 771bcb2c wininet!CFsm_SocketConnect::RunSM+0x42
        * 18 0bd4fe24 771bcada wininet!CFsm::Run+0x39
        * 19 0bd4fe3c 771bd22b wininet!DoFsm+0x25
        * 20 0bd4fe4c 771bd706 wininet!ICSocket::Connect+0x32
        * 21 0bd4fe8c 771bd4cb
wininet!HTTP_REQUEST_HANDLE_OBJECT::OpenConnection_Fsm+0x391
        * 22 0bd4fe98 771bcb2c wininet!CFsm_OpenConnection::RunSM+0x33
        * 23 0bd4feb0 771d734a wininet!CFsm::Run+0x39
        * 24 0bd4fee0 77f6ad84 wininet!CFsm::RunWorkItem+0x79

0:023> !uext.findstack wininet!CFsm::Run 0
Thread 020, 2 frame(s) match
Thread 023, 3 frame(s) match

0:023> !uext.findstack wininet!CFsm 0
Thread 020, 3 frame(s) match
Thread 023, 5 frame(s) match
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gatom

Статья • 05.04.2024

Расширение **!gatom** отображает глобальную таблицу atom.

```
dbgcmd
```

```
!gatom
```

## DLL-библиотеки

Ntsdexts.dll

## Дополнительная информация

Сведения о глобальной таблице atom см. в документации по пакету SDK для Microsoft Windows.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !igrep

Статья • 05.04.2024

**Расширение !igrep** ищет шаблон в разбиении.

```
dbgcmd
```

```
!igrep [Pattern [StartAddress]]
```

## Параметры

*Шаблон*
Указывает шаблон для поиска. Если опущено, используется последний *шаблон* .

*StartAddress*
Указывает шестнадцатеричный адрес, с которого начинается поиск. Если опущено, используется текущий счетчик программы.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Ntsdexts.dll |
| **Windows XP и более поздние версии** | Рекомендации недоступны |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !locks (!ntsdexts.locks)

Статья • 05.04.2024

Расширение **!locks** в Ntsdexts.dll отображает список критически важных разделов, связанных с текущим процессом.

Эта команда расширения не должна путаться с командой **расширения !kdext*.locks**

.

dbgcmd

```
    !locks [Options]
```

## Параметры

*Параметры*

Указывает объем отображаемых сведений. Можно использовать любое сочетание следующих параметров:

**-v**
Вызывает включение всех критически важных разделов, даже тех, которые в данный момент не принадлежат.

**-o**
Вызывает отображение только потерянных сведений (указатели, которые на самом деле не указывают на допустимые критические разделы).

### DLL-библиотеки

Ntsdexts.dll

## Дополнительная информация

Другие команды и расширения, которые могут отображать критически важные сведения о разделе, см. в разделе "Отображение критического раздела". Сведения о критических разделах см. в документации по пакету SDK Для Microsoft Windows и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

# Замечания

Эта команда расширения показывает все критически важные разделы, которые были инициализированы путем вызова **RtlInitializeCriticalSection**. Если критически важных разделов нет, выходные данные не будут результатом.

Рассмотрим пример:

dbgcmd

```
0:000> !locks

CritSec w3svc!g_pWamDictator+a0 at 68C2C298
LockCount          0
RecursionCount     1
OwningThread       d1
EntryCount         1
ContentionCount    0
*** Locked

CritSec SMTPSVC+66a30 at 67906A30
LockCount          0
RecursionCount     1
OwningThread       d0
EntryCount         1
ContentionCount    0
*** Locked
```

# См. также

Отображение критического раздела

Критические истечение времени ожидания разделов (режим пользователя)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !mapped_file

Статья • 05.04.2024

Расширение **!mapped_file** отображает имя файла, который поддерживает сопоставление файлов, содержащее указанный адрес.

dbgcmd

```
!mapped_file Address
```

## Параметры

*Адрес*
Указывает адрес сопоставления файлов. Если *адрес* не находится в сопоставлении, команда завершается ошибкой.

## DLL-библиотеки

Uext.dll

**Расширение !mapped_file** можно использовать только во время динамической отладки, нереметной отладки.

## Дополнительная информация

Дополнительные сведения о сопоставлении файлов см. в разделе MapViewOfFile в пакете SDK для Windows.

## Замечания

Ниже приведены три примера. Первые два адреса, используемые, сопоставляются из файла, а третий — нет.

dbgcmd

```
0:000> !mapped_file 4121ec
Mapped file name for 004121ec:
'\Device\HarddiskVolume2\CODE\TimeTest\Debug\TimeTest.exe'

0:000> !mapped_file 77150000
Mapped file name for 77150000:
```

```
'\Device\HarddiskVolume2\Windows\System32\kernel32.dll'

0:000> !mapped_file 80310000
No information found for 80310000: error 87
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !runaway

Статья • 05.04.2024

Расширение **!runaway** отображает сведения о времени, потребляемом каждым потоком.

dbgcmd

```
!runaway [Flags]
```

## Параметры

*Flags*
Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x1.

Bit 0 (0x1)
Приводит к тому, что отладчик отображает время пользователя, затраченное каждым потоком.

Бит 1 (0x2)
Приводит к тому, что отладчик отображает время ядра, затраченное каждым потоком.

Бит 2 (0x4)
Приводит к тому, что отладчик отображает время, прошедшее с момента создания каждого потока.

## DLL-библиотеки

Uext.dll

Ntsdexts.dll

## Дополнительная информация

**Расширение !runaway** можно использовать только во время динамической отладки или при отладке файлов аварийного дампа, созданных с помощью **.dump /mt** или **.dump /ma**.

Сведения о потоках в пользовательском режиме см. в разделе "Управление процессами и потоками". Дополнительные сведения об анализе процессов и потоков см. в статье *Microsoft Windows Internals* by Mark Russinovich и Дэвид Соломон.

## Замечания

Это расширение является быстрым способом узнать, какие потоки спинируются из-под контроля или потребляют слишком много времени ЦП.

Отображение определяет каждый поток внутренним нумерированием отладчика и идентификатором потока в шестнадцатеричном формате. Также отображаются идентификаторы отладчика.

Рассмотрим пример:

dbgcmd

```
0:001> !runaway 7

 User Mode Time
 Thread       Time
 0:55c        0:00:00.0093
 1:1a4        0:00:00.0000

 Kernel Mode Time
 Thread       Time
 0:55c        0:00:00.0140
 1:1a4        0:00:00.0000

 Elapsed Time
 Thread       Time
 0:55c        0:00:43.0533
 1:1a4        0:00:25.0876
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !threadtoken

Статья • 05.04.2024

**Расширение расширения !threadtoken** устарело.

```
dbgcmd
```

```
!threadtoken
```

## DLL-библиотеки

Рекомендации недоступны

## Дополнительная информация

Сведения о потоках и олицетворении см. в документации по пакету SDK Для Microsoft Windows и *внутренних компонентах* Microsoft Windows марком Руссиновичем и Дэвидом Соломоном.

## Замечания

**Расширение !threadtoken** устарело в Windows XP и более поздних версиях Windows. Вместо этого используйте **токен !token** .

Если текущий поток олицетворяется, будет отображаться маркер, который использует этот поток.

В противном случае появится сообщение с сообщением "Поток не олицетворение". Затем будет отображаться маркер процесса.

Маркеры будут отображаться в том же формате, что и !handle**, который** используется при отображении дескрипторов маркеров.

Рассмотрим пример:

```
dbgcmd
```

```
0:000> ~
.  0  id: 1d0.55c   Suspend: 1 Teb 7ffde000 Unfrozen
#  1  id: 1d0.1a4   Suspend: 1 Teb 7ffdd000 Unfrozen

0:000> !threadtoken
```

```
***Thread is not impersonating, using process token***
    Auth Id    0 : 0x1c93d
    Type       Primary
    Imp Level  Anonymous
     Token Id  0 : 0x5e8c19
     Mod Id    0 : 0x5e8c12
     Dyn Chg   0x1f4
     Dyn Avail 0x1a4
     Groups    26
     Privs     17
     User      S-1-5-21-2127521184-1604012920-1887927527-74790
     Groups    26
               S-1-5-21-2127521184-1604012920-1887927527-513
               S-1-1-0
               S-1-5-32-544
               S-1-5-32-545
               S-1-5-21-2127521184-1604012920-1887927527-277551
               S-1-5-21-2127521184-1604012920-1887927527-211604
               S-1-5-21-2127521184-1604012920-1887927527-10546
               S-1-5-21-2127521184-1604012920-1887927527-246657
               S-1-5-21-2127521184-1604012920-1887927527-277552
               S-1-5-21-2127521184-1604012920-1887927527-416040
               S-1-5-21-2127521184-1604012920-1887927527-96548
               S-1-5-21-2127521184-1604012920-1887927527-262644
               S-1-5-21-2127521184-1604012920-1887927527-155802
               S-1-5-21-2127521184-1604012920-1887927527-158763
               S-1-5-21-2127521184-1604012920-1887927527-279132
               S-1-5-21-2127521184-1604012920-1887927527-443952
               S-1-5-21-2127521184-1604012920-1887927527-175772
               S-1-5-21-2127521184-1604012920-1887927527-388472
               S-1-5-21-2127521184-1604012920-1887927527-443950
               S-1-5-21-2127521184-1604012920-1887927527-266975
               S-1-5-21-2127521184-1604012920-1887927527-158181
               S-1-5-21-2127521184-1604012920-1887927527-279435
               S-1-5-5-0-116804
               S-1-2-0
               S-1-5-4
               S-1-5-11
     Privileges    17
               SeUndockPrivilege ( Enabled Default )
               SeTakeOwnershipPrivilege ( )
               SeShutdownPrivilege ( )
               SeDebugPrivilege ( )
               SeIncreaseBasePriorityPrivilege ( )
               SeAuditPrivilege ( )
               SeSyncAgentPrivilege ( )
               SeLoadDriverPrivilege ( )
               SeSystemEnvironmentPrivilege ( Enabled )
               SeRemoteShutdownPrivilege ( )
               SeProfileSingleProcessPrivilege ( )
               SeCreatePagefilePrivilege ( )
               SeCreatePermanentPrivilege ( )
               SeSystemProfilePrivilege ( Enabled )
               SeBackupPrivilege ( )
```

```
                SeMachineAccountPrivilege ( )
                SeEnableDelegationPrivilege ( Enabled )
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !uniqstack

Статья • 05.04.2024

Расширение **!uniqstack** отображает все стеки для всех потоков в текущем процессе, за исключением стеков, которые, как представляется, дублируются.

dbgcmd

```
!uniqstack [ -b | -v | -p ] [ -n ]
```

## Параметры

**-b**
Приводит к включению первых трех параметров, передаваемых каждой функции.

**-v**
Приводит к тому, что отображение включает сведения об упущении указателя кадров (FPO). На процессорах на основе x86 также отображаются сведения о соглашении о вызовах.

**-p**
Вызывает отображение, чтобы включить полные параметры для каждой функции, вызываемой в трассировке стека. В этом списке будут указаны тип данных, имя и значение каждого параметра. *Для этого требуются полные сведения о символах.*

**-n**
Приводит к отображению номеров кадров.

## DLL-библиотеки

Uext.dll

## Замечания

Это расширение аналогично команде **k, kb, kc, kd, kp, kP, kv (Display Stack Backtrace),** за исключением того, что она не отображает повторяющиеся стеки.

Например:

dbgcmd

```
0:000> !uniqstack
Processing 14 threads, please wait

.  0  Id: f0f0f0f0.15c Suspend: 1 Teb: 00000000`7fff8000 Unfrozen
      Priority: 0
Child-SP          Child-BSP         RetAddr
00000000`0006e5e0 00000000`00070420 00000000`6b009840
00000000`0006e600 00000000`000703d0 00000000`6b03db00
00000000`0006e950 00000000`000703b0 00000000`6b008520
00000000`0006e950 00000000`00070368 00000000`6b1845e0
00000000`0006e9b0 00000000`00070310 00000000`6b009980
00000000`0006e9d0 00000000`000702d0 00000000`6b009ff0
00000000`0006e9e0 00000000`00070248 00000000`77ea4a10
00000000`0006f290 00000000`000700e0 00000000`77ea5d60
00000000`0006f4c0 00000000`00070028 00000000`77ed6000
00000000`0006f550 00000000`00070000 00000000`77ed9000
00000000`0006f550 00000000`00070000 00000000`77ca78a0
00000000`0006fff0 00000000`00070000 00000000`00000000

.  1  Id: f0f0f0f0.718 Suspend: 1 Teb: 00000000`7fff4000 Unfrozen
      Priority: 0
Child-SP          Child-BSP         RetAddr
00000000`0043eb50 00000000`00440250 00000000`6b008520
00000000`0043eb80 00000000`00440208 00000000`6b1845e0
00000000`0043ebe0 00000000`004401a8 00000000`6b009980
00000000`0043ec00 00000000`00440168 00000000`6b009ff0
00000000`0043ec10 00000000`004400e0 00000000`77ea5f50
00000000`0043f4c0 00000000`00440028 00000000`77ed6000
00000000`0043f550 00000000`00440000 00000000`77ed9000
00000000`0043f550 00000000`00440000 00000000`7de05690
00000000`0043fff0 00000000`00440000 00000000`00000000

. 13  Id: f0f0f0f0.494 Suspend: 1 Teb: 00000000`7ef98000 Unfrozen
      Priority: 0
Child-SP          Child-BSP         RetAddr
00000000`00feffe0 00000000`00ff0040 00000000`77e94f30
00000000`00feffe0 00000000`00ff0040 00000000`00000000

Total threads: 14
Duplicate callstacks: 11 (windbg thread #s follow):
2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !vadump

Статья • 05.04.2024

Расширение **!vadump** отображает все диапазоны виртуальной памяти и соответствующие сведения о защите.

```
dbgcmd
```

```
!vadump [-v]
```

## Параметры

**-v**
Приводит к включению сведений о каждом исходном регионе выделения. Так как отдельные адреса в регионе могут изменить защиту после выделения памяти (например **, VirtualProtect**), исходное состояние защиты для этого большего региона может не совпадать с тем, что каждый диапазон в пределах региона.

## DLL-библиотеки

Uext.dll

## Дополнительная информация

Чтобы просмотреть сведения о защите памяти для одного виртуального адреса, используйте **!vprot**. Сведения о защите памяти см. в разделе "*Внутренние* компоненты Microsoft Windows" Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

Рассмотрим пример:

```
dbgcmd
```

```
0:000> !vadump
BaseAddress:       00000000
RegionSize:        00010000
State:             00010000  MEM_FREE
Protect:           00000001  PAGE_NOACCESS
```

```
BaseAddress:       00010000
RegionSize:        00001000
State:             00001000  MEM_COMMIT
Protect:           00000004  PAGE_READWRITE
Type:              00020000  MEM_PRIVATE
.........
```

В этой строке состояния отображается состояние диапазона памяти, начиная с указанного BaseAddress. Возможные значения состояния : MEM_COMMIT, MEM_FREE и MEM_RESERVE.

В строке "Защита" отображается состояние защиты этого диапазона памяти. Возможные значения защиты: PAGE_NOACCESS, PAGE_READONLY, PAGE_READWRITE, PAGE_EXECUTE, PAGE_EXECUTE_READ, PAGE_EXECUTE_READWRITE, PAGE_WRITECOPY, PAGE_EXECUTE_WRITECOPY и PAGE_GUARD.

В строке "Тип" показан тип памяти. Возможные значения: MEM_IMAGE, MEM_MAPPED и MEM_PRIVATE.

Ниже приведен пример использования **параметра -v** :

dbgcmd

```
0:000> !vadump -v
BaseAddress:       00000000
AllocationBase:    00000000
RegionSize:        00010000
State:             00010000  MEM_FREE
Protect:           00000001  PAGE_NOACCESS

BaseAddress:       00010000
AllocationBase:    00010000
AllocationProtect: 00000004  PAGE_READWRITE
RegionSize:        00001000
State:             00001000  MEM_COMMIT
Protect:           00000004  PAGE_READWRITE
Type:              00020000  MEM_PRIVATE
.........
```

При **использовании -v** строка AllocationProtect показывает защиту по умолчанию, с помощью которую был создан весь регион. В строке "Защита" показана фактическая защита этого конкретного адреса.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !vprot

Статья • 05.04.2024

Расширение **!vprot** отображает сведения о защите виртуальной памяти.

dbgcmd

```
!vprot [Address]
```

## Параметры

*Адрес*
Указывает шестнадцатеричный адрес, состояние защиты памяти которого должно отображаться.

## DLL-библиотеки

Uext.dll

## Дополнительная информация

Чтобы просмотреть сведения о защите памяти для всех диапазонов памяти, принадлежащих целевому процессу, используйте **!vadump**. Сведения о защите памяти см. в разделе "*Внутренние* компоненты Microsoft Windows" Марком Руссиновичем и Дэвидом Соломоном.

## Замечания

**Команда расширения !vprot** может использоваться как для отладки в реальном времени, так и для отладки файлов дампа.

Рассмотрим пример:

dbgcmd

```
0:000> !vprot 30c191c
BaseAddress: 030c1000
AllocationBase: 030c0000
AllocationProtect: 00000080 PAGE_EXECUTE_WRITECOPY
RegionSize: 00011000
State: 00001000 MEM_COMMIT
```

```
Protect: 00000010 PAGE_EXECUTE
Type: 01000000 MEM_IMAGE
```

На этом экране строка AllocationProtect показывает защиту по умолчанию, с помощью которую был создан весь регион. Обратите внимание, что отдельные адреса в этом регионе могут изменить защиту после выделения памяти (например, если **вызывается VirtualProtect** ). В строке "Защита" показана фактическая защита этого конкретного адреса. Возможные значения защиты: PAGE_NOACCESS, PAGE_READONLY, PAGE_READWRITE, PAGE_EXECUTE, PAGE_EXECUTE_READ, PAGE_EXECUTE_READWRITE, PAGE_WRITECOPY, PAGE_EXECUTE_WRITECOPY и PAGE_GUARD.

Строка состояния также применяется к конкретному виртуальному адресу, переданном **в !vprot**. Возможные значения состояния : MEM_COMMIT, MEM_FREE и MEM_RESERVE.

В строке "Тип" показан тип памяти. Возможные значения: MEM_IMAGE, MEM_MAPPED и MEM_PRIVATE.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения отладчика ядра хранилища

18.07.2025

Расширения отладчика ядра хранилища (storagekd) используются для отладки драйверов хранилища в целевых объектах операционной системы Windows 8 и выше.

Extension commands that are useful for debugging storage drivers, via classpnp managed storage class drivers and Storport managed storage miniport drivers, can be found in **Storagekd.dll**.

Ознакомьтесь с расширениями минипорта SCSI (Scsikd.dll и Minipkd.dll) для отладки для windows 7 и более ранней версии целевых объектов ОС.

**Important** You need special symbols to use this extension. Дополнительные сведения см. в статье Средства отладки для Windows.

## Команды расширения отладчика ядра хранилища

Развернуть таблицу

| Command | Description |
|---|---|
| !storagekd.storhelp | Displays help text for **Storagekd.dll** extension commands. |
| !storagekd.storclass | Displays information about the specified *classpnp* device. |
| !storagekd.storadapter | Displays information about the specified *Storport* adapter. |
| !storagekd.storunit | Displays information about the specified *Storport* logical unit. |
| !storagekd.storloglist | Displays the *Storport* adapter's internal log entries. |
| !storagekd.storlogirp | Displays the *Storport's* internal log entries for the adapter filtered for the IRP provided. |
| !storagekd.storlogsrb | Displays the *Storport's* internal log entries for the adapter filtered for the Storage (or SCSI) Request Block (SRB) provided. |
| !storagekd.storsrb | Отображает сведения о указанном блоке |

| Command | Description |
| --- | --- |
| | запросов хранилища (или SCSI). |

# !storagekd.storadapter

Статья • 08.07.2024

Расширение **!storagekd.storadapter** отображает сведения о указанном адаптере Storport.

```
dbgcmd
```

```
!storagekd.storadapter [Address]
```

## Параметры

*Адрес*
Указывает адрес объекта устройства адаптера Storport. Если *адрес* опущен, отображается список всех адаптеров Storport.

## DLL-библиотеки

Storagekd.dll

## Замечания

Ниже приведен пример **отображения !storagekd.storadapter** :

**1: kd> ! storagekd.storadapter**

```
dbgcmd
```

```
# STORPORT adapters:
==================
## Driver                 Object            Extension          State
-----------------------------------------------------------------------
\Driver\vhdmp          fffffa800649a050  fffffa800649a1a0   Working
```

**1: kd> ! storagekd.storadapter fffffa800649a050**

```
dbgcmd
```

```
ADAPTER
   DeviceObj : fffffa800649a050   AdapterExt: fffffa800649a1a0   DriverObj :
fffffa800507fcb0
DeviceState : Working
```

```
   LowerDO fffffa8005f71e10   PhysicalDO fffffa8005f71e10
   SlowLock Free   RemLock -666
   SystemPowerState: Working AdapterPowerState D0   Full Duplex
   Bus 0   Slot 0   DMA 0000000000000000   Interrupt 0000000000000000
   Allocated ResourceList 0000000000000000
Translated ResourceList 0000000000000000
   Gateway: Outstanding 0   Lower 256   High 256
   PortConfigInfo fffffa800649a2d0
   HwInit fffffa80062e8840   HwDeviceExt fffffa8004b84d70   (112 bytes)
   SrbExt 2256 bytes   LUExt 24 bytes

Normal Logical Units:
   Product                 SCSI ID  Object            Extension          Pnd
Out Ct State
   ------------------------------------------------------------------------
---------------
   Msft       Virtual Di   0  0  1  fffffa800658a060  fffffa800658a1b0    0
0  0  Working

   Zombie Logical Units:
   Product                 SCSI ID  Object            Extension          Pnd
Out Ct State
   ------------------------------------------------------------------------
-------------
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !storagekd.storclass

Статья • 08.07.2024

Расширение **!storagekd.storclass** отображает сведения об указанном *устройстве classpnp* .

```
dbgcmd
```

```
!storagekd.storclass [Address [Level]]
```

## Параметры

*Адрес*
Указывает адрес объекта устройства или расширения устройства классarpnp. Если *адрес* опущен, отображается список всех расширений classpnp.

*Уровень*
Указывает объем отображаемых сведений. Этот параметр может иметь значение 0, 1 или 2, а 2 — максимально подробно и 0 меньше. По умолчанию установлено значение 0.

## DLL-библиотеки

Storagekd.dll

## Замечания

Ниже приведен пример **отображения !storagekd.storclass** :

**1: kd> !storagekd.storclass**

```
dbgcmd
```

```
Storage class devices:

* !storclass fffffa80043dc060 [1,2] ST3160812AS Paging Disk
  !storclass fffffa8006581740 [1,2] Msft Virtual Disk Disk

Usage: !storclass <class device> <level [0-2]>
```

**1: kd> !storagekd.storclass fffffa80043dc060 1**

dbgcmd

```
Storage class device fffffa80043dc060 with extension at fffffa80043dc1b0

Classpnp Internal Information at fffffa8003bec360

    Transfer Packet Engine:

     Packet          Status  DL Irp          Opcode  Sector/ListId   UL Irp
    --------         ------ --------         ------ --------------- --------

    Pending Idle Requests: 0x0


    -- dt classpnp!_CLASS_PRIVATE_FDO_DATA fffffa8003bec360 --

Classpnp External Information at fffffa80043dc1b0

    ST3160812AS 3.ADH             9LS20QRL

    Minidriver information at fffffa80043dc670
    Attached device object at fffffa800410a060
    Physical device object at fffffa800410a060

    Media Geometry:

        Bytes in a Sector = 512
        Sectors per Track = 63
        Tracks / Cylinder = 255
        Media Length      = 160000000000 bytes = ~149 GB

    -- dt classpnp!_FUNCTIONAL_DEVICE_EXTENSION fffffa80043dc1b0 --
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !storagekd.storhelp

Статья • 05.04.2024

Расширение **!storagekd.storhelp** отображает текст справки для команд расширения служба хранилища kd.dll.

dbgcmd

```
!storagekd.storhelp
```

# DLL-библиотеки

служба хранилища kd.dll

# Замечания

Ниже приведен пример **отображения !storagekd.storhelp** :

**0: kd> !storagekd.storhelp**

dbgcmd

```
# Storage Debugger Extension
================================================
## General Commands
----------------
!storhelp    - Displays complete help of the commands provided in this KD
extension
!storclass   - Dumps all class devices managed by classpnp
!storadapter - Dumps all adapters managed by Storport
!storunit    - Dumps all disks managed by Storport

## STORPORT specific commands
-------------------------
!storlogirp <args>     - displays internal log entries that reference the
specified IRP.
                         See '!storhelp storlogirp' for details.
!storloglist <args>    - displays internal log entries. See '!storhelp
storloglist' for details.
!storlogsrb <args>     - displays internal log entries that reference the
specified SRB.
                         See '!storhelp storlogsrb' for details.
!storsrb <address>     - display details for the specified SCSI or STORAGE
request block
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !storagekd.storlogirp

Статья • 08.07.2024

Расширение **!storagekd.storlogirp** отображает внутренние записи журнала Storport для адаптера, отфильтрованного для предоставленного IRP.

dbgcmd

```
!storagekd.storlogirp <Address> <irp> [<starting_entry> [<ending_entry>]] [L
<count>]
```

## Параметры

*Адрес*
Указывает адрес расширения устройства storport или объекта устройства.

*irp*
Поиск IRP.

*starting_entry*
Начальная запись в диапазоне для отображения. Если не указано, будут отображаться последние *записи счетчиков* .

*ending_entry*
Конечная запись в диапазоне для отображения. Если запись не указана, *будут отображаться записи счетчиков* , начиная с элемента, указанного *starting_entry*.

*считать*
Количество отображаемых записей. Если значение не указано, используется значение 50.

## DLL-библиотеки

Storagekd.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !storagekd.storloglist

Статья • 08.07.2024

Расширение **!storagekd.storloglist** отображает внутренние записи журнала адаптера Storport.

```
dbgcmd
```

```
!storagekd.storloglist <Address> [<starting_entry> [<ending_entry>]] [L
<count>]
```

## Параметры

*Адрес*
Указывает адрес расширения устройства storport или объекта устройства.

*starting_entry*
Начальная запись в диапазоне для отображения. Если не указано, будут отображаться последние *записи счетчиков* .

*ending_entry*
Конечная запись в диапазоне для отображения. Если запись не указана, *будут отображаться записи счетчиков* , начиная с элемента, указанного *starting_entry*.

*считать*
Количество отображаемых записей. Если значение не указано, используется значение 50.

## DLL-библиотеки

Storagekd.dll

## Замечания

Ниже приведен пример **отображения !storagekd.storloglist** :

**0: kd> !storagekd.storloglist ffffe0010f5e01a0**

```
dbgcmd
```

```
Storport RaidLogList
    Circular buffer location:  0xffffe0010f5e1720
    Total logs written: 8
    Displaying entries 0 through 7
    ------------------------------------------------------------------------
    [0]_[23:04:20.521] ResumeDevice.......... Caller:
storport!RaidUnitPauseTimerDpcRoutine+0x28 (fffff800`fb4d0b28), P/P/T/L:
0/0/0/0, Pause count: 0, Resumed: True
    [1]_[23:04:20.646] ResumeDevice.......... Caller:
storport!RaidUnitPauseTimerDpcRoutine+0x28 (fffff800`fb4d0b28), P/P/T/L:
0/0/0/0, Pause count: 0, Resumed: True
    [2]_[23:04:20.646] SpPauseDevice......... Caller:
iaStorAV!RpiPauseDevice+0x67 (fffff800`fb70554f), P/T/L: 3/0/0, Timeout:
180, Adapter: 0xffffe0010f5e01a0
    [3]_[23:04:20.646] PauseDevice........... Caller:
storport!StorPortPauseDevice+0x2f6 (fffff800`fb4b52d6), P/P/T/L: 0/3/0/0,
Pause count: 1
    [4]_[23:04:20.646] SpResumeDevice........ Caller:
iaStorAV!RpiResumeDevice+0x5f (fffff800`fb7055fb), P/T/L: 3/0/0, Adapter:
0xffffe0010f5e01a0
    [5]_[23:04:20.646] ResumeDevice.......... Caller:
storport!StorPortResumeDevice+0x19 (fffff800`fb4aa23f), P/P/T/L: 0/3/0/0,
Pause count: 0, Resumed: True
    [6]_[23:04:20.646] SpPauseDevice......... Caller:
iaStorAV!RpiPauseDevice+0x67 (fffff800`fb70554f), P/T/L: 3/0/0, Timeout:
180, Adapter: 0xffffe0010f5e01a0
    [7]_[23:04:20.646] PauseDevice........... Caller:
storport!StorPortPauseDevice+0x2f6 (fffff800`fb4b52d6), P/P/T/L: 0/3/0/0,
Pause count: 1
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !storagekd.storlogsrb

Статья • 08.07.2024

Расширение **!storagekd.storlogsrb** отображает внутренние записи журнала Storport для адаптера, отфильтрованного для блока запросов хранилища (или SCSI) (SRB).

dbgcmd

```
!storagekd.storlogsrb <Address> <srb> [<starting_entry> [<ending_entry>]] [L
<count>]
```

## Параметры

*Адрес*
Указывает адрес расширения устройства storport или объекта устройства.

*SRB*
Объект SRB для поиска.

*starting_entry*
Начальная запись в диапазоне для отображения. Если не указано, будут отображаться последние *записи счетчиков* .

*ending_entry*
Конечная запись в диапазоне для отображения. Если запись не указана, *будут отображаться записи счетчиков* , начиная с элемента, указанного *starting_entry*.

*считать*
Количество отображаемых записей. Если значение не указано, используется значение 50.

## DLL-библиотеки

Storagekd.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !storagekd.storsrb

Статья • 08.07.2024

Расширение **!storagekd.storsrb** отображает сведения о указанном блоке запросов хранилища (или SCSI).

dbgcmd

```
!storagekd.storsrb Address
```

## Параметры

*Адрес*
Указывает адрес SRB.

## DLL-библиотеки

Storagekd.dll

## Замечания

Ниже приведен пример **отображения !storagekd.storsrb** :

**0: kd> !storagekd.storsrb ffffe0011fe25b0**

dbgcmd

```
    SRB is a STORAGE request block (SRB_EX)
    SRB EX 0xffffe00111fe25b0 Function 28  Version  1, Signature 53524258,
SrbStatus: 0x02[Aborted], SrbFunction 0x00 [EXECUTE SCSI]
    Address Type is BTL8

       SRB_EX Data Type [SrbExDataTypeScsiCdb16]
    [EXECUTE SCSI] SRB_EX: 0xffffe00111fe2648  OriginalRequest:
0xffffe001125a9010  DataBuffer/Length: 0xffffe00112944000 / 0x00000200
    PTL: (0, 1, 1)  CDB: 28 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
OpCode: SCSI/READ (10)
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !storagekd.storunit

Статья • 08.07.2024

Расширение **!storagekd.storunit** отображает сведения о указанном логическом модуле Storport.

dbgcmd

```
!storagekd.storunit [Address]
```

## Параметры

*Адрес*
Указывает адрес объекта устройства storport. Если *адрес* опущен, отображается список всех единиц Storport.

## DLL-библиотеки

Storagekd.dll

## Замечания

Ниже приведен пример **отображения !storagekd.storunit** :

**0: kd> !storagekd.storunit**

dbgcmd

```
# STORPORT Units:
==================
## Product                 SCSI ID  Object            Extension         Pnd
Out Ct  State
-------------------------------------------------------------------------------
----------
Msft       Virtual Di   0  0  1  fffffa800658a060  fffffa800658a1b0    0   0
0  Working
```

**0: kd> !storagekd.storunit fffffa800658a060**

dbgcmd

```
   DO fffffa800658a060   Ext fffffa800658a1b0   Adapter fffffa800649a1a0
Working
   Vendor: Msft       Product: Virtual Disk       SCSI ID: (0, 0, 1)
   Claimed Enumerated
   SlowLock Free   RemLock 1   PageCount 0
   QueueTagList: fffffa800658a270      Outstanding: Head fffffa800658a398
Tail fffffa800658a398  Timeout -1
   DeviceQueue fffffa800658a2a0   Depth: 250   Status: Not Frozen
PauseCount: 0   BusyCount: 0
   IO Gateway: Busy Count 0   Pause Count 0
   Requests: Outstanding 0   Device 0   ByPass 0


[Device-Queued Requests]

## IRP               SRB Type   SRB               XRB               Command
MDL               SGList            Timeout
------------------------------------------------------------------------------
------------------------------------------------------


[Bypass-Queued Requests]

## IRP               SRB Type   SRB               XRB               Command
MDL               SGList            Timeout
------------------------------------------------------------------------------
------------------------------------------------------


[Outstanding Requests]

## IRP               SRB Type   SRB               XRB               Command
MDL               SGList            Timeout
------------------------------------------------------------------------------
------------------------------------------------------


[Completed Requests]

IRP               SRB Type   SRB               XRB               Command
MDL               SGList            Timeout
------------------------------------------------------------------------------
------------------------------------------------------
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения Bluetooth (Bthkd.dll)

18.07.2025

Расширения отладчика Bluetooth отображают сведения о текущей среде Bluetooth в целевой системе.

**Note** As you work with the Bluetooth debugging extensions, you may come across undocumented behavior or APIs. Настоятельно рекомендуется принимать зависимости от незадокументированного поведения или API, так как они могут изменяться в будущих выпусках.

## В этом разделе

Развернуть таблицу

| Topic | Description |
| --- | --- |
| !bthkd.bthdevinfo | The !bthkd.bthdevinfo command displays the information about a given BTHENUM created device PDO. |
| !bthkd.bthenuminfo | The !bthkd.bthenuminfo command displays information about the BTHENUM FDO. |
| !bthkd.bthinfo | The !bthkd.bthinfo command displays details about the BTHPORT FDO. Эта команда является хорошей отправной точкой для исследований Bluetooth, так как она отображает сведения об адресе, которые можно использовать для доступа ко многим другим командам расширения отладки Bluetooth. |
| !bthkd.bthhelp | The !bthkd.bthhelp command displays help for the Bluetooth debug extension commands. |
| !bthkd.bthtree | The !bthkd.bthtree command displays the complete Bluetooth device tree. |
| !bthkd.bthusbtransfer | The !bthkd.bthusbtransfer command displays the Bluetooth usb transfer context including Irp, Bip and transfer buffer information. |
| !bthkd.dibflags | The !bthkd.dibflags command displays DEVICE_INFO_BLOCK.DibFlags dumps flags set in _DEVICE_INFO_BLOCK.DibFlags. |

| Topic | Description |
| --- | --- |
| !bthkd.hcicmd | The !bthkd.hcicmd command displays a list of the currently pending commands. |
| !bthkd.hciinterface | The !bthkd.hciinterface command displays the bthport!_HCI_INTERFACE structure. |
| !bthkd.l2capinterface | The !bthkd.l2capinterface command displays information about the L2CAP interface. |
| !bthkd.rfcomminfo | The !bthkd.rfcomminfo command displays information about the RFCOMM FDO and the TDI Device Object. |
| !bthkd.rfcommconnection | The !bthkd.rfcommconnection command displays information about a given RFCOMM connection object. |
| !bthkd.rfcommchannel | The !bthkd.rfcommchannel command displays information about a given RFCOMM channel CB. |
| !bthkd.sdpinterface | The !bthkd.sdpinterface command displays information about the SDP interface. |
| !bthkd.scointerface | The !bthkd.scointerface command displays information about the SCO interface. |
| !bthkd.sdpnode | The !bthkd.sdpnode command displays information about a node in an sdp tree. |
| !bthkd.sdpstream | The !bthkd.sdpstream command displays the contents of a SDP stream. |

# !bthkd.bthdevinfo

Статья • 05.04.2024

Команда **!bthkd.bthdevinfo** отображает сведения об заданном созданном BTHENUM устройстве PDO.

dbgsyntax

```
!bthkd.bthdevinfo addr
```

## Параметры

*Addr*
Адрес созданного расширения PDO устройства BTHENUM.

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.bthenuminfo

Статья • 05.04.2024

Команда **!bthkd.bthenuminfo** отображает сведения о FDO BTHENUM.

```
dbgsyntax
```

```
!bthkd.bthenuminfo
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.bthinfo

Статья • 05.04.2024

Команда **!bthkd.bthinfo** отображает сведения о FDO BTHPORT. Эта команда является хорошей отправной точкой для исследований Bluetooth, так как она отображает сведения об адресе, которые можно использовать для доступа ко многим другим командам расширения отладки Bluetooth.

dbgsyntax

```
!bthkd.bthinfo
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.bthhelp

Статья • 05.04.2024

Команда **!bthkd.bthhelp** отображает справку по командам расширения отладки Bluetooth.

```
dbgsyntax
```

```
!bthkd.bthhelp
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.bthtree

Статья • 05.04.2024

Команда **!bthkd.bthtree** отображает полное дерево устройств Bluetooth.

```
dbgsyntax
```

```
!bthkd.bthtree
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.bthusbtransfer

Статья • 05.04.2024

Команда **!bthkd.bthusbtransfer** отображает контекст передачи usb Bluetooth, включая Irp, Bip и данные буфера передачи.

dbgsyntax

```
!bthkd.bthusbtransfer addr
```

## Параметры

*Addr*
Адрес контекста передачи USB Bluetooth.

## Замечания

Для отображения адреса контекста передачи USB можно использовать команду !bthinfo. Он указан в разделе списка передачи.

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.dibflags

Статья • 08.07.2024

Команда **!bthkd.dibflags** отображает DEVICE_INFO_BLOCK. Флаги Дампа DibFlags, заданные в _DEVICE_INFO_BLOCK. DibFlags.

dbgsyntax

```
!bthkd.dibflags flags
```

## Параметры

*Флаги*
Значение _DEVICE_INFO_BLOCK. Флаги дампов DibFlags, заданные в _DEVICE_INFO_BLOCK. DibFlags.

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.hcicmd

Статья • 05.04.2024

Команда **!bthkd.hcicmd** отображает список ожидающих команд.

```
dbgsyntax
```

```
!bthkd.hcicmd
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.hciinterface

Статья • 05.04.2024

Команда **!bthkd.hciinterface** отображает структуру bthport!_HCI_INTERFACE.

```
dbgsyntax
```

```
!bthkd.hciinterface
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.l2capinterface

Статья • 05.04.2024

Команда **!bthkd.l2capinterface** отображает сведения об интерфейсе L2CAP.

```
dbgsyntax
```

```
!bthkd.l2capinterface
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.rfcomminfo

Статья • 05.04.2024

Команда **!bthkd.rfcomminfo** отображает сведения о FDO RFCOMM и объекте устройства TDI.

```
dbgsyntax
```

```
!bthkd.rfcomminfo
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.rfcommconnection

Статья • 08.07.2024

Команда **!bthkd.rfcommconnection** отображает сведения о заданном объекте подключения RFCOMM.

```
dbgsyntax
```

```
!bthkd.rfcommconnection addr
```

## Параметры

*addr*
Адрес структуры rfcomm!_RFCOMM_CONN_OBJ.

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.rfcommchannel

Статья • 05.04.2024

Команда **!bthkd.rfcommchannel** отображает сведения о заданном канале RFCOMM CB.

dbgsyntax

```
!bthkd.rfcommchannel addr
```

## Параметры

*Addr*
Адрес структуры rfcomm!_CHANNEL_CB.

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.sdpinterface

Статья • 05.04.2024

Команда **!bthkd.sdpinterface** отображает сведения об интерфейсе SDP.

dbgsyntax

```
!bthkd.sdpinterface
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.scointerface

Статья • 05.04.2024

Команда **!bthkd.scointerface** отображает сведения об интерфейсе SCO.

```
dbgsyntax
```

```
!bthkd.scointerface
```

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.sdpnode

Статья • 08.07.2024

Команда **!bthkd.sdpnode** отображает сведения о узле в дереве sdp.

```
dbgsyntax
```

```
!bthkd.sdpnode addr [flags]
```

## Параметры

*addr*
Адрес отображаемого узла дерева sdp.

*Флаги*
0x1 — узел recurse

0x2 — подробные сведения

значение по умолчанию — 0x0

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !bthkd.sdpstream

Статья • 05.04.2024

Команда **!bthkd.sdpstream** отображает содержимое потока SDP.

```
dbgsyntax
```

```
!bthkd.sdpstream streamaddr streamlength
```

## Параметры

*streamaddr*
Указатель на начало потока.

*streamlength*
Длина потока SDP в байтах для отображения.

## DLL-библиотеки

Bthkd.dll

## См. также

Расширения Bluetooth (Bthkd.dll)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !acxkd

21.06.2024

Расширение **!acxkd** отображает сведения о драйверах расширения аудиокласса (ACX). Дополнительные сведения о расширениях класса ACX см. в обзоре расширений аудиоклассов ACX.

# Синтаксис

dbgcmd

```
!acxkd.[Options]
```

## DLL-библиотеки

Acxkd.dll

## Версия отладчика

Расширение !acxkd доступно в WinDbg версии 1.2402.24001.0 и более поздних версиях.

## Отладка ACX 1.0

Расширение отладчика !acxkd предлагает только частичные функции в ACX 1.0. Рекомендуется обновить до ACX 1.1.

# Параметры

*Параметры* . Указывает тип отображаемой информации.

Развернуть таблицу

| Вариант | Описание | Параметр |
|---|---|---|
| !Справка | Отображает сведения о доступных командах расширения. | `[<command name>]` |
| !acxcircuit | Дамп объекта ACXCIRCUIT. | `<circuit>` — дескриптор WDF ACXCIRCUIT |

| Вариант | Описание | Параметр |
|---|---|---|
| !acxdataformat | Дамп объекта ACXDATAFORMAT. | `<dataformat>` — дескриптор WDF ACXDATAFORMAT |
| !acxdataformatlist | Дамп объекта ACXDATAFORMATLIST. | `<dataformatlist>` — дескриптор WDF ACXDATAFORMATLIST |
| !acxdevice | Дампа объекта ACXDEVICE. | `<device>` — дескриптор WDF ACXDEVICE |
| !acxelement | Дамп объекта ACXELEMENT. | `<element>` — дескриптор WDF ACXELEMENT |
| !acxevents | Дамп событий объекта ACXOBJECT. | `<events>` — дескриптор WDF ACXOBJECT |
| !acxfactory | Дамп объекта ACXFACTORYCIRCUT. | `<factory>` — дескриптор WDF ACXFACTORYCIRCUIT |
| !acxmanager | Дамп объекта ACXMANAGER. | нет |
| !acxmethods | Методы дампа объекта ACXOBJECT. | `<object>` — дескриптор WDF ACXOBJECT |
| !acxobjbag | Дампа объекта ACXOBJECTBAG. | `<objbag>` — дескриптор WDF ACXOBJECTBAG |
| !acxobject | Дамп объекта ACXOBJECT. | `<object>` — дескриптор WDF ACXOBJECT |
| !acxpin | Дампа объекта ACXPIN. | `<pin>` — дескриптор WDF ACXPIN |
| !acxproperties | Свойства дампа объекта ACXOBJECT. | `<properties>` — дескриптор WDF ACXOBJECT |
| !acxstream | Дамп объекта ACXSTREAM. | `<stream>` — дескриптор ACXSTREAM |
| !acxstreambridge | Дамп объекта ACXSTREAMBRIDGE. | `<bridge>` — дескриптор ACXSTREAMBRIDGE |
| !acxtarget | Дампа объекта ACXTARGET. | `<target>` — дескриптор WDF ACXTARGET |
| !acxtemplate | Дампа объекта ACXTEMPLATE. | `<tmpt>` — дескриптор ACXTEMPLATE |

# Замечания

## !acxkd.help

Чтобы получить список всех доступных команд, используйте команду acxkd `!help` .

dbgcmd

```
0: kd> !acxkd.help
Commands for C:\Debugger\acxkd.dll:
  !acxcircuit        - Dump a ACXCIRCUT object.
  !acxdataformat     - Dump a ACXDATAFORMAT object.
  !acxdataformatlist - Dump a ACXDATAFORMATLIST object.
  !acxdevice         - Dump a ACXDEVICE object.
  !acxelement        - Dump a ACXELEMENT object.
  !acxevents         - Dump events of a ACXOBJECT object.
  !acxfactory        - Dump a ACXFACTORYCIRCUT object.
  !acxmanager        - Dump a ACXMANAGER object.
  !acxmethods        - Dump methods of a ACXOBJECT object.
  !acxobjbag         - Dump a ACXOBJECTBAG object.
  !acxobject         - Dump a ACXOBJECT* object.
  !acxpin            - Dump a ACXPIN object.
  !acxproperties     - Dump properties of a ACXOBJECT object.
  !acxstream         - Dump a ACXSTREAM object.
  !acxstreambridge   - Dump a ACXSTREAMBRIDGE object.
  !acxtarget         - Dump a ACXTARGET* object.
  !acxtemplate       - Dump a ACX*TEMPLATE object.
  !help              - Displays information on available extension commands
!help <cmd> will give more information for a particular command
```

Используйте команду acxkd `!help` , чтобы узнать больше о любой из команд, например о команде `!acxdevice` .

dbgcmd

```
0: kd> !acxkd.help acxdevice
!acxdevice <device>
  <device> - ACXDEVICE handle
Dump a ACXDEVICE object.
```

Используйте команду !acxdevice в качестве отправной точки для проверки драйвера ACX.

dbgcmd

```
3: kd> !acxdevice 0x00007dfadb0a5358
Dumping info for ACXDEVICE 0x00007dfadb0a5358

    In connected standby: FALSE

    State: AfxDeviceStateInitialized
    State history:
        0 : AfxDeviceStateInvalid
        1 : AfxDeviceStateInvalid
        2 : AfxDeviceStateInvalid
```

```
        3 : AfxDeviceStateInvalid
        4 : AfxDeviceStateInvalid
        5 : AfxDeviceStateCreated
        6 : AfxDeviceStateInitializing
        7 : AfxDeviceStateInitialized

    Create dispatch list:
        Create name: eHDMIOutTopo
        Dispatch routine: fffff80393179918
        Dispatch context: ffff82052513b960

    Circuits:
        ---------------------------------

        [Circuit 0]

        Name: eHDMIOutTopo
        Type: AcxCircuitTypeRender
        ComponentId: {BFCA9AD9-4EED-46C2-9323-B5D4400761A5}

        State: AfxCircuitStatePoweredUp

        Interface is enabled
        SymolicLinkName: \??
\HDAUDIO#SUBFUNC_01&VEN_8086&DEV_281F&NID_0001&SUBSYS_00000000&REV_1000#6&4948348&
0&0002&00000025#{2c6bb644-e1ae-47f8-9a2b-1d1fa750f2fa}\eHDMIOutTopo

        !acxproperties 00007dfad9ccccb8
        !acxmethods 00007dfad9ccccb8
        !acxevents 00007dfad9ccccb8

        # Pins: 2
            !acxpin 00007dfadf996dd8
            !acxpin 00007dfad4697238

        # Elements: 1
            !acxelement 00007dfadf997a18

        # Streams: 0

        !acxcircuit 00007dfad9ccccb8

    !wdfqueue 00007dfade9beaf8
    !wdfdevice 00007dfadb0a5358

    !wdfhandle 00007dfadb0a5358
    dt Acx01000!Acx::AfxDevice ffff8205256ab420
```

Щелкните ссылки в выходных данных, чтобы отобразить информацию с помощью команд !acxproperties, !acxmethods и !acxevents.

Сведения о поиске wdfhandle для объекта ACXDEVICE см . в пошаговом руководстве по примеру драйвера ACX в этой статье.

## Команды WDF — !wdfkd.wdfldr

Так как драйверы ACX являются драйверами WDF, используйте любую из команд отладчика ядра WDF. Например, используйте !wdfkd.wdfldr для отображения сведений о версии и привязке ACX с WDF.

dbgcmd

```
0: kd> !wdfldr acx01000
WDF Driver: Acx01000
---------------------------------
CLIENT_MODULE   0xffff82052abdcbc0
  WDF Version   v1.31
  ImageName     Acx01000.sys
  ImageAddress  0xfffff80393150000
  ImageSize     0xb3000
  BindingList   0xffff82052abdcc08

  ImageName        WdfVer Ver   WdfGlobals         BindInfo           ImageAddress
ImageSize
  Wdf01000.sys     v1.33  v1.33 0xffff8205218d8fb0 0xffff8205218d8df0
0xfffff80356a00000 0x000c7000
---------------------------------
CLASS_MODULE    0xffff820525b3dc90
  WDF Version   v1.31
  Version       v1.1
  Service       \Registry\Machine\System\CurrentControlSet\Services\acx01000
  ImageName     Acx01000.sys
  ImageAddress  0xfffff80393150000
  ImageSize     0xb3000
  ClientsList   0xffff820525b3dcf8
  Associated Clients: 1

  ImageName        WdfVer Ver   WdfGlobals         BindInfo           ImageAddress
ImageSize
  AcxHdAudio.sys   v1.25  v1.0  0xffff820527f70ae0 0xfffff803930df3e8
0xfffff803930c0000 0x0008a000
---------------------------------
```

## !acxkd.acxmanager

Используйте команду для отображения сведений об объекте `!acxmanager` ACXMANAGER. Это обеспечивает хорошую отправную точку для изучения драйверов ACX.

В этом примере показана первая часть обширных `!acxmanager` выходных данных, предоставляемых для конфигурации ACX mulitcircuit.

dbgcmd

```
10: kd> !acxmanager
Dumping info for ACXMANAGER 0x000054f94d1d4378

    Delete pending: No

    # singleton composites: 8
        ---------------------------------

        [Composite 0]

        State: AfxCompositeStateActive
        !acxobjbag 000054f94c8e61c8
        !acxtemplate 000054f94c014d28
        !acxobject 000054f94c0141c8

        # circuits: 3
            ---------------------------------
            [Circuit 0 CORE]
…
```

В этом примере выходных данных показан один канал ACX.

dbgcmd

```
0: kd> !acxmanager
Dumping info for ACXMANAGER 0x000049f6c3c769f8

    Delete pending: No

    # singleton composites: 0
    # Composite factories: 0
    !wdfhandle 000049f6c3c769f8
    dt Acx01000!Acx::AfxManager ffffb6093c3896b0
```

## !acxkd.acxobject

В выходных данных для !acxmanager указан адрес для wdfhandle. Используйте адрес wdfhandle с командой `!acxobject` для отображения сведений об ACXMANAGER или любом другом объекте ACX.

dbgcmd

```
0: kd> !acxobject 000049f6c3c769f8
Dumping info for ACXMANAGER 0x000049f6c3c769f8

    Delete pending: No

    # singleton composites: 0
    # Composite factories: 0
```

```
    !wdfhandle 000049f6c3c769f8
    dt Acx01000!Acx::AfxManager ffffb6093c3896b0
```

Щелкните ссылку `dt` в выходных данных, показанных выше, чтобы просмотреть дополнительные сведения о внутренних структурах объектов ACX.

dbgcmd

```
0: kd> dt Acx01000!Acx::AfxManager ffffb6093c3896b0
   +0x000 m_Object         : 0x000049f6`c3c769f8 ACXMANAGER__
   =fffff803`8a478ad8 s_AfxManager     : 0xffffb609`3c3896b0 Acx::AfxManager
   +0x008 m_AcxGlobals     : 0xffffb609`403cae24 _ACX_DRIVER_GLOBALS
   +0x010 m_Flags          : 0
   +0x010 m_Unloading      : 0y0
   +0x018 m_CompositesMutex : _FAST_MUTEX
   +0x050 m_CompositeFactoriesList : _LIST_ENTRY [ 0xffffb609`3c389700 -
0xffffb609`3c389700 ]
   +0x060 m_CompositeFactoriesCount : 0n0
   +0x068 m_CompositesList : _LIST_ENTRY [ 0xffffb609`3c389718 -
0xffffb609`3c389718 ]
   +0x078 m_CompositesCount : 0n0
   +0x080 m_FactoriesList  : _LIST_ENTRY [ 0xffffb609`3c389730 -
0xffffb609`3c389730 ]
   +0x090 m_CircuitsList   : _LIST_ENTRY [ 0xffffb609`40a27068 -
0xffffb609`40a27068 ]
   +0x0a0 m_FiltersMutex   : _FAST_MUTEX
   +0x0d8 m_FactoryFiltersList : _LIST_ENTRY [ 0xffffb609`3c389788 -
0xffffb609`3c389788 ]
   +0x0e8 m_CircuitFiltersList : _LIST_ENTRY [ 0xffffb609`3c389798 -
0xffffb609`3c389798 ]
   +0x0f8 m_FactoriesNotificationHandle : 0xfffffc70c`b2356790 Void
   +0x100 m_CircuitsNotificationHandle : 0xfffffc70c`b2357ec0 Void
   +0x108 m_CommandQueue   : 0xffffb609`3ffc5400 Acx::AfxWorkQueue
```

Обратите внимание, что выходные данные отладчика могут содержать сочетание общедоступных объектов ACX, таких как ACXMANAGER, ACXCIRCFACTORY и ACXCIRCUIT и внутренние структуры, определенные для непрозрачности. Внутренние типы не гарантируют, что они остаются неизменными или доступны в разных выпусках ACX и не должны вызываться или использоваться напрямую.

Так как объекты ACX являются объектами WDF, можно использовать команду !wdfkd.wdfhandle для отображения дополнительных сведений об объекте ACXMANAGER.

dbgcmd

```
0: kd> !wdfhandle 000049f6c3c769f8
Treating handle as a KMDF handle!

Dumping WDFHANDLE 0x000049f6c3c769f8
```

```
============================
Handle type is WDFOBJECT [ACXMANAGER]
Refcount: 2
Contexts:
    context:  dt 0xffffb6093c3896b0 Acx01000!AfxManager (size is 0x110 bytes)
    EvtCleanupCallback fffff8038a453070
Acx01000!Acx::WdfCpp::ObjectContext<ACXMANAGER__
*,Acx::AfxManager>::EvtObjectContextCleanupThunk
    EvtDestroyCallback fffff8038a453000
Acx01000!Acx::WdfCpp::ObjectContext<ACXMANAGER__
*,Acx::AfxManager>::EvtObjectContextDestroyThunk

    context:  dt 0xffffb609404c2280 Acx01000!WdfCustomType_ACXMANAGER (size is
0x10 bytes)
    <no associated attribute callbacks>

Parent: !wdfhandle 0x000049f6d4e2d5c8, type is WDFDEVICE
Owning device: !wdfdevice 0x000049f6d4e2d5c8

!wdfobject 0xffffb6093c389600
```

## !acxkd.acxpin

Команды, отображающие другие сведения ACX, например `!acxpin` требовать дескриптор WDF для объекта. Сведения о поиске дескриптора WDF для объекта ACX см . в пошаговом руководстве по примеру драйвера ACX в этой статье.

dbgcmd

```
0: kd> !acxpin 0x000049f6befeee38
Dumping info for ACXPIN 0x000049f6befeee38

    ID: 0
    Type: AcxPinTypeSink
    Type: AcxPinCommunicationSink

    Category: KSCATEGORY_AUDIO
    Name: {00000000-0000-0000-0000-000000000000}
```

В зависимости от состояния объекта ACX не все сведения могут быть доступны для отображения.

## !acxkd.acxdataformatlist

Как и в !acxpin, `!acxdataformatlist` отображаются сведения о списках данных ACX.

dbgcmd

```
0: kd> !acxdataformatlist 0x000049f6bf8be668
Dumping info for ACXDATAFORMATLIST 0x000049f6bf8be668

    # Scan count: 0
    # Data formats: 6
    Data formats:
        Sample Rate: 48000, #Channels: 2, #Bits: 16, ValidBits: 16, Mask: 0x3
(default)
        Sample Rate: 48000, #Channels: 2, #Bits: 32, ValidBits: 24, Mask: 0x3
        Sample Rate: 44100, #Channels: 2, #Bits: 16, ValidBits: 16, Mask: 0x3
        Sample Rate: 44100, #Channels: 2, #Bits: 32, ValidBits: 24, Mask: 0x3
        Sample Rate: 32000, #Channels: 2, #Bits: 16, ValidBits: 16, Mask: 0x3
        Sample Rate: 32000, #Channels: 2, #Bits: 32, ValidBits: 24, Mask: 0x3

    !wdfhandle 000049f6bf8be668
    dt Acx01000!Acx::AfxDataFormatList ffffb60940444530
```

# Пошаговое руководство по драйверу ACX

В этом разделе представлено пошаговое руководство по отладке драйвера ACX.

## Путь к символам

Используйте команды Symfix и .sympath (Set Symbol Path) для изменения пути символа. Если вы используете локальный код с драйвером, добавьте путь к этому коду. Используйте команду reload (Reload Module) для перезагрузки символов из текущего пути.

dbgcmd

```
.symfix
.sympath+ C:\Windows-driver-samples-develop\audio\Acx\Samples\AudioCodec\Driver
.reload /f
```

## Контекст отладчика для драйверов

Если вы выполняете отладку активного драйвера ACX, установите точку останова. Это помещает отладчик в контекст объектов ACX, чтобы иметь возможность собирать и отображать сведения.

Эти примеры точек останова предназначены для запуска примера драйвера AudioCodec.

dbgcmd

```
bm AudioCodec!DriverEntry
bm AudioCodec!AcxDriverInitialize
```

Эти примеры точек останова предназначены для запуска при выполнении определенных действий, таких как создание пин-кода или канала.

dbgcmd

```
bm AudioCodec!AcxPinCreate
bm AudioCodec!AcxCircuitCreate
bm AudioCodec!Codec_EvtBusDeviceAdd
```

После загрузки драйвера и запуска соответствующей точки останова и допустимого контекста выполнения используйте команды !acxkd для отображения сведений о любом объекте ACX. Используйте команду !acxobject для общих сведений и определенной команды, например !acxcircuit или !acxpin для более детализированной информации.

## Загрузка библиотеки acxkd dll

Используйте команду load (DLL расширения загрузки) для загрузки расширения acxkd.dll.

dbgcmd

```
.load acxkd.dll
```

## Отображение сведений о драйвере ACX

Чтобы собрать сведения о целевом драйвере, используйте команду lm (список загруженных модулей), чтобы просмотреть все загруженные драйверы. Затем используйте параметры Dvm для отображения сведений о интересующем драйвере ACX, как показано здесь.

dbgcmd

```
0: kd> lm Dvm AcxHdAudio
Browse full module list
start             end                 module name
fffff803`8a3c0000 fffff803`8a448000   AcxHdAudio   (private pdb symbols)
C:\ProgramData\Dbg\sym\AcxHdAudio.pdb\6AEA2622909B20C1AD149C57ACBB4A6F1\AcxHdAudio
.pdb
    Loaded symbol image file: AcxHdAudio.sys
    Mapped memory image file:
C:\ProgramData\Dbg\sym\AcxHdAudio.sys\0829423388000\AcxHdAudio.sys
    Image path: \SystemRoot\System32\drivers\AcxHdAudio.sys
```

```
    Image name: AcxHdAudio.sys
    Browse all global symbols  functions  data  Symbol Reload
    Image was built with /Brepro flag.
    Timestamp:        08294233 (This is a reproducible build file hash, not a
timestamp)
    CheckSum:         00087DD6
    ImageSize:        00088000
    Translations:     0000.04b0 0000.04e4 0409.04b0 0409.04e4
    Information from resource tables:
```

Используйте команду x (Проверка символов) и маску подстановочного знака для отображения определенных структур ACX, таких как ACXPIN.

dbgcmd

```
0: kd> x /D AcxHdAudio!acxpin*
 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

fffff803`8a3e3216 AcxHdAudio!AcxPinGetRawDataFormatList =  (inline caller)
AcxHdAudio!HDACodec_EvtFormatChange+66
fffff803`8a3e31e0 AcxHdAudio!AcxPinGetCircuit =  (inline caller)
AcxHdAudio!HDACodec_EvtFormatChange+30
fffff803`8a3e361f AcxHdAudio!AcxPinNotifyDataFormatChange =  (inline caller)
AcxHdAudio!HDACodec_EvtFormatChange+46f
fffff803`8a3e7396 AcxHdAudio!AcxPinCreate =  (inline caller)
AcxHdAudio!HDACodecR_CreateRenderCircuit+962
fffff803`8a3e74a2 AcxHdAudio!AcxPinGetRawDataFormatList =  (inline caller)
AcxHdAudio!HDACodecR_CreateRenderCircuit+a6e
fffff803`8a3e75f8 AcxHdAudio!AcxPinCreate =  (inline caller)
AcxHdAudio!HDACodecR_CreateRenderCircuit+bc4
fffff803`8a3e789a AcxHdAudio!AcxPinAddJacks =  (inline caller)
AcxHdAudio!HDACodecR_CreateRenderCircuit+e66
fffff803`8a3e5ae5 AcxHdAudio!AcxPinGetCircuit =  (inline caller)
AcxHdAudio!HDACodecR_EvtAcxPinRetrieveJackSinkInfo+25
fffff803`8a3e5913 AcxHdAudio!AcxPinGetCircuit =  (inline caller)
AcxHdAudio!HDACodecR_EvtAcxPinRetrieveName+73
fffff803`8a3ea277 AcxHdAudio!AcxPinCreate =  (inline caller)
AcxHdAudio!HDACodecC_CreateCaptureCircuit+b27
fffff803`8a3ea39c AcxHdAudio!AcxPinGetRawDataFormatList =  (inline caller)
AcxHdAudio!HDACodecC_CreateCaptureCircuit+c4c
fffff803`8a3ea4bc AcxHdAudio!AcxPinCreate =  (inline caller)
AcxHdAudio!HDACodecC_CreateCaptureCircuit+d6c
fffff803`8a3ea7b2 AcxHdAudio!AcxPinAddJacks =  (inline caller)
AcxHdAudio!HDACodecC_CreateCaptureCircuit+1062
```

В зависимости от текущего контекста выполнения в отладчике можно использовать dx (выражение объектной модели отладчика) для детализации конкретных структур ACX.

dbgcmd

```
0: kd> dx -r2 AudioCodec!AcxDeviceAddCircuit
AudioCodec!AcxDeviceAddCircuit                 :
AudioCodec!AcxDeviceAddCircuit+0x0 [Type: long __cdecl(WDFDEVICE__ *,ACXCIRCUIT__
*)]
0: kd> u fffff8007ead1120
AudioCodec!AcxDeviceAddCircuit [C:\Program Files (x86)\Windows
Kits\10\Include\10.0.26016.0\km\acx\km\1.1\AcxDevice.h @ 206]:
fffff800`7ead1120 4889542410      mov     qword ptr [rsp+10h],rdx
fffff800`7ead1125 48894c2408      mov     qword ptr [rsp+8],rcx
fffff800`7ead112a 4883ec38        sub     rsp,38h
fffff800`7ead112e b808000000      mov     eax,8
fffff800`7ead1133 486bc046        imul    rax,rax,46h
fffff800`7ead1137 488d0dc2ab0000  lea     rcx,[AudioCodec!AcxFunctions
(fffff800`7eadbd00)]
fffff800`7ead113e 488b0401        mov     rax,qword ptr [rcx+rax]
fffff800`7ead1142 4889442420      mov     qword ptr [rsp+20h],rax
```

## !wdfkd.wdfdriverinfo

Используйте команду !wdfdriverinfo с именем драйвера для сбора сведений WDF, таких как связанные объекты ACX и wdfhandle ACXDEVICE.

dbgcmd

```
0: kd> !wdfdriverinfo AcxHdAudio.sys 1 -v
---------------------------------
Default driver image name: AcxHdAudio
WDF library image name: Wdf01000
 FxDriverGlobals  0xffffb609403e3de0
 WdfBindInfo      0xfffff8038a3dd1b0
   Version        v1.25
 Library module   0xffffb60929cc6050
   ServiceName    \Registry\Machine\System\CurrentControlSet\Services\Wdf01000
   ImageName      Wdf01000
---------------------------------
WDFDRIVER: 0x000049f6bfe9d5d8
    context:  dt 0xffffb60940162bc0 AcxHdAudio!CODEC_DRIVER_CONTEXT (size is 0x1
bytes)
    <no associated attribute callbacks>

    context:  dt 0xffffb6093ccead20 Acx01000!AfxDriver (size is 0x20 bytes)
    EvtCleanupCallback fffff8038a455780
Acx01000!Acx::WdfCpp::ObjectContext<WDFDRIVER__
*,Acx::AfxDriver>::EvtObjectContextCleanupThunk
    EvtDestroyCallback fffff8038a455740
Acx01000!Acx::WdfCpp::ObjectContext<WDFDRIVER__
*,Acx::AfxDriver>::EvtObjectContextDestroyThunk
Object Hierarchy: !wdfhandle 0x000049f6bfe9d5d8 0xff
Driver logs: !wdflogdump AcxHdAudio.sys -d
Framework logs: !wdflogdump AcxHdAudio.sys -f
```

```
    !wdfdevice 0x000049f6bffba488 ff (FDO)
        Pnp/Power State: WdfDevStatePnpStarted, WdfDevStatePowerD0,
WdfDevStatePwrPolStartedWakeCapable
        context:  dt 0xffffb60940045e60 AcxHdAudio!CODEC_DEVICE_CONTEXT (size is
0x110 bytes)
        EvtCleanupCallback fffff8038a3e3c90
AcxHdAudio!HDACodec_EvtDeviceContextCleanup

        context:  dt 0xffffb60933f030f0 Acx01000!AfxDevice (size is 0x150 bytes)
        EvtCleanupCallback fffff8038a451910
Acx01000!Acx::WdfCpp::ObjectContext<WDFDEVICE__
*,Acx::AfxDevice>::EvtObjectContextCleanupThunk
        EvtDestroyCallback fffff8038a451740
Acx01000!Acx::WdfCpp::ObjectContext<WDFDEVICE__
*,Acx::AfxDevice>::EvtObjectContextDestroyThunk

        context:  dt 0xffffb60940a26b90 AcxHdAudio!CODEC_RENDER_DEVICE_CONTEXT
(size is 0x38 bytes)
        <no associated attribute callbacks>

        context:  dt 0xffffb609409f2e00 AcxHdAudio!CODEC_CAPTURE_DEVICE_CONTEXT
(size is 0x8 bytes)
        <no associated attribute callbacks>
        !wdfdevicequeues 0x000049f6bffba488

        !wdfdevice 0x000049f6bef1a848 ff (PDO)
            Pnp/Power State: WdfDevStatePnpStarted,
WdfDevStatePowerD0BusWakeOwner, WdfDevStatePwrPolStartedWakeCapable
            context:  dt 0xffffb609410e5aa0 AcxHdAudio!CODEC_RENDER_DEVICE_CONTEXT
(size is 0x38 bytes)
            EvtCleanupCallback fffff8038a3e6400
AcxHdAudio!HDACodecR_EvtDeviceContextCleanup

            context:  dt 0xffffb60933f090d0 Acx01000!AfxDevice (size is 0x150
bytes)
            EvtCleanupCallback fffff8038a451910
Acx01000!Acx::WdfCpp::ObjectContext<WDFDEVICE__
*,Acx::AfxDevice>::EvtObjectContextCleanupThunk
            EvtDestroyCallback fffff8038a451740
Acx01000!Acx::WdfCpp::ObjectContext<WDFDEVICE__
*,Acx::AfxDevice>::EvtObjectContextDestroyThunk
            !wdfdevicequeues 0x000049f6bef1a848

---------------------------------

WDF Verifier settings for AcxHdAudio.sys is OFF
---------------------------------
```

В выходных данных, показанных выше, ссылка доступна связанному wdfdevice. Щелкните ссылку, чтобы отобразить сведения о связанных объектах устройств WDF.

```
dbgcmd
```

```
0: kd> !wdfdevice 0x000049f6bef1a848 ff
Treating handle as a KMDF handle!

Dumping WDFDEVICE 0x000049f6bef1a848
================================

WDM PDEVICE_OBJECTs:  self ffffb60940ddbdd0

Pnp state:  119 ( WdfDevStatePnpStarted )
Power state:  309 ( WdfDevStatePowerD0BusWakeOwner )
Power Pol state:  531 ( WdfDevStatePwrPolStartedWakeCapable )
```

## !acxkd.acxdevice

Использование того же wdfhandle с `!acxdevice` информацией, ориентированной на ACX.

dbgcmd

```
3: kd> !acxdevice 0x00007dfadb0a5358
Dumping info for ACXDEVICE 0x00007dfadb0a5358

    In connected standby: FALSE

    State: AfxDeviceStateInitialized
    State history:
        0 : AfxDeviceStateInvalid
        1 : AfxDeviceStateInvalid
        2 : AfxDeviceStateInvalid
        3 : AfxDeviceStateInvalid
        4 : AfxDeviceStateInvalid
        5 : AfxDeviceStateCreated
        6 : AfxDeviceStateInitializing
        7 : AfxDeviceStateInitialized

    Create dispatch list:
        Create name: eHDMIOutTopo
        Dispatch routine: fffff80393179918
        Dispatch context: ffff82052513b960

    Circuits:
        ---------------------------------

        [Circuit 0]

        Name: eHDMIOutTopo
        Type: AcxCircuitTypeRender
        ComponentId: {BFCA9AD9-4EED-46C2-9323-B5D4400761A5}

        State: AfxCircuitStatePoweredUp

        Interface is enabled
```

```
        SymolicLinkName: \??
\HDAUDIO#SUBFUNC_01&VEN_8086&DEV_281F&NID_0001&SUBSYS_00000000&REV_1000#6&4948348&
0&0002&00000025#{2c6bb644-e1ae-47f8-9a2b-1d1fa750f2fa}\eHDMIOutTopo

        !acxproperties 00007dfad9ccccb8
        !acxmethods 00007dfad9ccccb8
        !acxevents 00007dfad9ccccb8

        # Pins: 2
            !acxpin 00007dfadf996dd8
            !acxpin 00007dfad4697238

        # Elements: 1
            !acxelement 00007dfadf997a18

        # Streams: 0

        !acxcircuit 00007dfad9ccccb8

    !wdfqueue 00007dfade9beaf8
    !wdfdevice 00007dfadb0a5358

    !wdfhandle 00007dfadb0a5358
    dt Acx01000!Acx::AfxDevice ffff8205256ab420
```

Чтобы отобразить сведения о других объектах ACX, таких как ACXCIRCUIT, используйте ссылку в выходных данных выше, которая вызывается `!acxcircuit` с соответствующим wdfhandle.

dbgcmd

```
3: kd> !acxcircuit 00007dfad9ccccb8
Dumping info for ACXCIRCUIT 0x00007dfad9ccccb8

    Name: eHDMIOutTopo
    Type: AcxCircuitTypeRender
    ComponentId: {BFCA9AD9-4EED-46C2-9323-B5D4400761A5}

    State: AfxCircuitStatePoweredUp
    State history:
        0 : AfxCircuitStateInvalid
        1 : AfxCircuitStateInvalid
        2 : AfxCircuitStateInvalid
        3 : AfxCircuitStateCreated
        4 : AfxCircuitStateInitializing
        5 : AfxCircuitStateInitialized
        6 : AfxCircuitStatePoweredDown
        7 : AfxCircuitStatePoweredUp

    Interface is enabled
    SymolicLinkName: \??
\HDAUDIO#SUBFUNC_01&VEN_8086&DEV_281F&NID_0001&SUBSYS_00000000&REV_1000#6&4948348&
```

```
0&0002&00000025#{2c6bb644-e1ae-47f8-9a2b-1d1fa750f2fa}\eHDMIOutTopo

    # Power references: 0
    # Open handles: 18

    !acxproperties 00007dfad9ccccb8
    !acxmethods 00007dfad9ccccb8
    !acxevents 00007dfad9ccccb8

    # Pins: 2
        !acxpin 00007dfadf996dd8
        !acxpin 00007dfad4697238

    # Elements: 1
        !acxelement 00007dfadf997a18
```

## !acxkd.acxproperties

В выходных ссылках команды с wdfhandle предоставляются для других объектов, таких как свойства ACX, которые можно отобразить.

dbgcmd

```
0: kd> !acxproperties 000049f6bf436b88
Dumping properties info for ACXOBJECT 0x000049f6bf436b88

    # sets: 4

        Set: {8C134960-51AD-11CF-878A-94F801C10000}
            Id: 0, Flags: 0x1
            Id: 1, Flags: 0x1

        Set: {720D4AC0-7533-11D0-A5D6-28DB04C10000}
            Id: 0, Flags: 0x1
            Id: 1, Flags: 0x1
            Id: 2, Flags: 0x1
            Id: 3, Flags: 0x1

        Set: {C034FDB0-FF75-47C8-AA3C-EE46716B50C6}
            Id: 1, Flags: 0x1
            Id: 2, Flags: 0x1
            Id: 3, Flags: 0x1

        Set: {4D12807E-55DB-48B8-A466-F15A510F5817}
            Id: 1, Flags: 0x1
```

## !wdfhandle

Также доступно в выходных `!wdfdriverinfo` данных ссылка на wdfhandle для иерархии объектов WDF, связанной с ACX.

dbgcmd

```
Object Hierarchy: !wdfhandle 0x000049f6bfe9d5d8 0xff
```

При нажатии этой ссылки отображается иерархия объектов WDF для драйвера ACX. Эти выходные данные можно использовать для поиска дескрипторов WDF для других объектов ACX и WDF.

dbgcmd

```
0: kd> !wdfhandle 0x000049f6bfe9d5d8 0xff
Treating handle as a KMDF handle!

Dumping WDFHANDLE 0x000049f6bfe9d5d8
============================
Handle type is WDFDRIVER
Refcount: 1
Contexts:
    context:  dt 0xffffb60940162bc0 AcxHdAudio!CODEC_DRIVER_CONTEXT (size is 0x1
bytes)
    <no associated attribute callbacks>

    context:  dt 0xffffb6093ccead20 Acx01000!AfxDriver (size is 0x20 bytes)
    EvtCleanupCallback fffff8038a455780
Acx01000!Acx::WdfCpp::ObjectContext<WDFDRIVER__
*,Acx::AfxDriver>::EvtObjectContextCleanupThunk
    EvtDestroyCallback fffff8038a455740
Acx01000!Acx::WdfCpp::ObjectContext<WDFDRIVER__
*,Acx::AfxDriver>::EvtObjectContextDestroyThunk

Child WDFHANDLEs of 0x000049f6bfe9d5d8:
    !wdfhandle 0x000049f6bfe9d5d8  dt FxDriver 0xffffb60940162a20 Context
ffffb60940162bc0, Context ffffb6093ccead20 Cleanup fffff8038a455780 Destroy
fffff8038a455740
        !wdfdevice 0x000049f6bffba488  dt FxDevice 0xffffb60940045b70 Context
ffffb60940045e60 Cleanup fffff8038a3e3c90, Context ffffb60933f030f0 Cleanup
fffff8038a451910 Destroy fffff8038a451740, Context ffffb60940a26b90, Context
ffffb609409f2e00
            WDF INTERNAL   dt FxDefaultIrpHandler 0xffffb6093fa54130
            WDF INTERNAL   dt FxPkgGeneral 0xffffb609401f5610
            WDF INTERNAL   dt FxWmiIrpHandler 0xffffb609401f6510
            WDF INTERNAL   dt FxPkgIo 0xffffb6093f6d2400
                !wdfqueue 0x000049f6c00114b8  dt FxIoQueue 0xffffb6093ffeeb40
            WDF INTERNAL   dt FxPkgFdo 0xffffb6093f2ae020
            !wdfhandle 0x000049f6bfe099f8  dt FxCmResList 0xffffb609401f6600
            !wdfhandle 0x000049f6bfe0a358  dt FxCmResList 0xffffb609401f5ca0
            !wdfchildlist 0x000049f6c09adb98  dt FxChildList 0xffffb6093f652460
            !wdfiotarget 0x000049f6c092d1d8  dt FxIoTarget 0xffffb6093f6d2e20
```

```
            !wdfqueue 0x000049f6c04904c8  dt FxIoQueue 0xffffb6093fb6fb30 Context
ffffb6093fa8dcd0 Cleanup fffff8038a45a350 Destroy fffff8038a45a310
            !wdfhandle 0x000049f6c09e7ed8  dt FxWorkItem 0xffffb6093f618120
Context ffffb6093f618220
            WDF INTERNAL   dt FxWmiProvider 0xffffb609403d9b50
                WDF INTERNAL   dt FxWmiInstanceExternal 0xffffb60940a11320
            WDF INTERNAL   dt FxWmiProvider 0xffffb609403d9300
                WDF INTERNAL   dt FxWmiInstanceExternal 0xffffb60940a11720
        !wdfdevice 0x000049f6bef1a848  dt FxDevice 0xffffb609410e57b0 Context
ffffb609410e5aa0 Cleanup fffff8038a3e6400, Context ffffb60933f090d0 Cleanup
fffff8038a451910 Destroy fffff8038a451740
            WDF INTERNAL   dt FxDefaultIrpHandler 0xffffb60940c16450
            WDF INTERNAL   dt FxPkgGeneral 0xffffb60940bc9a10
            WDF INTERNAL   dt FxWmiIrpHandler 0xffffb60940bc9380
            WDF INTERNAL   dt FxPkgIo 0xffffb609400ab5c0
                !wdfqueue 0x000049f6beefd588  dt FxIoQueue 0xffffb60941102a70
            WDF INTERNAL   dt FxPkgPdo 0xffffb609410e6020
            !wdfhandle 0x000049f6bf436e58  dt FxCmResList 0xffffb60940bc91a0
            !wdfhandle 0x000049f6bf436d68  dt FxCmResList 0xffffb60940bc9290
            !wdfqueue 0x000049f6bef185c8  dt FxIoQueue 0xffffb609410e7a30 Context
ffffb60940e72ed0 Cleanup fffff8038a45a350 Destroy fffff8038a45a310
            !wdfhandle 0x000049f6bf436b88 [ACXCIRCUIT]  dt FxUserObject
0xffffb60940bc9470 Context ffffb60940bc9520, Context
...
```

## Команды журнала WDF

!wdflogdump может быть полезно для устранения неполадок драйвера ACX, отображая сведения о журнале WDF.

Отображение журнала для ACX с помощью параметра -d driver.

dbgcmd

```
0: kd> !wdflogdump acx01000 -d
Log dump command                          Log ID                  Size
================                          ======                  ====
 !wdflogdump  Acx01000 -a 0xFFFF820527861000  AFX_Client1             4096
 !wdflogdump  Acx01000 -a 0xFFFF8205215C0000  AFX_Log                 4096
Trace searchpath is:

Trace format prefix is: %7!u!: %!FUNC! -
Trying to extract TMF information from -
C:\ProgramData\Dbg\sym\Acx01000.pdb\B13D39B43205B60C07935803D7CB96981\Acx01000.pdb
--- start of log ---
AFX_Client1 1: Acx::AfxCircuit::Register - INFO:ACXCIRCUIT 00007DFAD9CCCCB8
Registered
AFX_Client1 2: Acx::AfxCircuit::PowerUpNotification - INFO:ACXCIRCUIT
00007DFAD9CCCCB8, EvtAcxCircuitPowerUp callback, STATUS_SUCCESS
AFX_Client1 5: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
```

```
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 7: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 8: Acx::AfxMute::EvtMuteEventEnableCallback - INFO:ACXMUTE
00007DFADF997A18, enabled ACXEVENT 00007DFADF996F98
AFX_Client1 10: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 12: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 13: Acx::AfxPin::EvtJackEventEnableCallback - INFO:ACXPIN
00007DFAD4697238, enabled ACXEVENT 00007DFADF9973F8
AFX_Client1 15: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 17: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
```

Используйте !wdflogdump , чтобы отобразить журнал платформы для определенного драйвера ACX с помощью параметра -f.

dbgcmd

```
0: kd> !wdflogdump AcxHdAudio -f
Trace searchpath is:

Trace format prefix is: %7!u!: %4!s! %!FUNC! -
Trying to extract TMF information from -
C:\ProgramData\Dbg\sym\Wdf01000.pdb\CBDEA3A4F64C17C1752E652A91DD14761\Wdf01000.pdb
Gather log: Please wait, this may take a moment (reading 4024 bytes).
% read so far ... 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
There are 65 log entries
--- start of log ---
16131: 09/03/2023-23:43:07.3233594 imp_WdfRegistryOpenKey - new WDFKEY object open
failed, 0xc0000034(STATUS_OBJECT_NAME_NOT_FOUND)
16132: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleDecrementIo from FxIdleBusy
16133: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleStartTimer from FxIdleDecrementIo
16134: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleTimerRunning from FxIdleStartTimer
16135: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleCancelTimer from FxIdleTimerRunning
16136: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleCheckIoCount from FxIdleCancelTimer
```

```
16137:
...
```

## См. также

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра". Пошаговое руководство по отладке с помощью звукового драйвера WDM см. в разделе "Драйверы отладки " Пошаговые лаборатории (режим ядра Sysvad)". Дополнительные сведения о расширениях класса ACX см. в обзоре расширений аудиоклассов ACX.

# GPIO Extensions

18.07.2025

Команды расширения "Входные и выходные данные общего назначения" (GPIO) отображают состояние программного обеспечения контроллеров GPIO. Эти команды отображают сведения из структур данных, поддерживаемых драйвером расширения платформы GPIO (Msgpioclx.sys). Сведения о расширении платформы GPIO см. вGeneral-Purpose драйверах ввода-вывода (GPIO).

Команды расширения отладчика GPIO реализуются в gpiokd.dll. To load the GPIO commands, enter **.load gpiokd.dll** in the debugger.

Каждый контроллер GPIO имеет набор банков. Каждый банк имеет таблицу пин-кода с массивом пин-кодов. Команды расширения отладчика GPIO отображают сведения о контроллерах GPIO, банках, таблицах закреплении и закреплении.

## Структуры данных, используемые командами GPIO

Команды расширения отладчика GPIO используют эти структуры данных, определенные Msgpioclx.sys.

**msgpioclx!_DEVICE_EXTENSION**
Структура расширения устройства для драйвера расширения платформы GPIO. Эта структура содержит сведения об отдельном контроллере GPIO.

**msgpioclx!_GPIO_BANK_ENTRY**
Эта структура содержит сведения об отдельном банке контроллера GPIO.

**msgpioclx!_GPIO_PIN_INFORMATION_ENTRY**
Эта структура содержит сведения о индивидуальном закреплении в банке контроллера GPIO.

## Начало работы с отладкой GPIO

To start debugging a GPIO issue, enter the **!gpiokd.clientlist** command. The **!gpiokd.clientlist** command displays an overview of all registered GPIO controllers and displays addresses that you can pass to other GPIO debugger commands.

## В этом разделе

Развернуть таблицу

| Topic | Description |
| --- | --- |
| !gpiokd.help | The !gpiokd.help command displays help for the GPIO debugger extension commands. |
| !gpiokd.bankinfo | The !gpiokd.bankinfo command displays information about a GPIO bank. |
| !gpiokd.clientlist | The !gpiokd.clientlist command displays all registered GPIO controllers. |
| !gpiokd.gpioext | The !gpiokd.gpioext command displays information about a GPIO controller. |
| !gpiokd.pininfo | The !gpiokd.pininfo command displays information about a specified GPIO pin. |
| !gpiokd.pinisrvec | The !gpiokd.pinisrvec command displays Interrupt Service Routine (ISR) vector information for a specified pin. |
| !gpiokd.pintable | The !gpiokd.pintable command displays information about an array of GPIO pins. |

# См. также

Специализированные команды расширения

# !gpiokd.help

Статья • 05.04.2024

Расширение **!gpiokd.help** отображает справку по командам расширения отладчика GPIO.

## DLL-библиотеки

Gpiokd.dll

## См. также

Расширения GPIO

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gpiokd.bankinfo

Статья • 08.07.2024

Расширение **!gpiokd.bankinfo** отображает сведения о банке GPIO.

```
dbgcmd
```

```
!gpiokd.bankinfo BankAddress [Flags]
```

## Параметры

*BankAddress*
Адрес структуры _GPIO_BANK_ENTRY, представляющей банк контроллера GPIO.

*Флаги*
Флаги, указывающие, какие сведения отображаются. Этот параметр является битовой или одним или несколькими из следующих флагов.

Развернуть таблицу

| Флаг | Description |
|---|---|
| 0x1 | Отображает таблицу закреплений. |
| 0x2 | Отображает регистры включения и маски. |
| 0x4 | Если установлен бит 0 (0x1), а этот флаг (0x4) установлен, отображается ненастройанные закрепления. |

## DLL-библиотеки

Gpiokd.dll

## См. также

Расширения GPIO

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gpiokd.clientlist

Статья • 08.07.2024

Расширение **!gpiokd.clientlist** отображает все зарегистрированные контроллеры GPIO.

dbgcmd

```
!gpiokd.clientlist [Flags]
```

## Параметры

*Флаги*
Флаги, указывающие, какие сведения отображаются. Этот параметр является битовой или одним или несколькими из следующих флагов.

Развернуть таблицу

| Флаг | Description |
|---|---|
| 0x1 | Для каждого контроллера отображаются подробные сведения, включая все его банки. |
| 0x2 | Если установлен бит 0 (0x1), а этот флаг (0x2) установлен, отображается регистрация включения и маски для каждого банка. |
| 0x4 | Если установлен бит 0 (0x1), а этот флаг (0x4) установлен, отображается ненастройанные закрепления. |

## DLL-библиотеки

Gpiokd.dll

## См. также

Расширения GPIO

---

## Обратная связь

-

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gpiokd.gpioext

Статья • 08.07.2024

Расширение **!gpiokd.gpioext** отображает сведения о контроллере GPIO.

```
dbgcmd
```

```
!gpiokd.gpioext ExtensionAddress [Flags]
```

## Параметры

*ExtensionAddress*
Адрес структуры _DEVICE_EXTENSION, представляющей контроллер GPIO.

*Флаги*
Флаги, указывающие, какие сведения отображаются. Этот параметр является битовой или одним или несколькими из следующих флагов.

Развернуть таблицу

| Флаг | Description |
|---|---|
| 0x1 | Отображает таблицу закреплений для каждого банка. |
| 0x2 | Если установлен бит 0 (0x1), а этот флаг (0x2) установлен, отображается список "Включить" и "Маскировать" для каждого банка. |
| 0x4 | Если установлен бит 0 (0x1), а этот флаг (0x4) установлен, отображается ненастройанные закрепления. |

## DLL-библиотеки

Gpiokd.dll

## См. также

Расширения GPIO

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gpiokd.pininfo

Статья • 05.04.2024

Расширение **!gpiokd.pininfo** отображает сведения о указанном пин-коде GPIO.

```
dbgcmd
```

```
!gpiokd.pininfo PinAddress
```

## Параметры

*PinAddress*
Адрес структуры данных _GPIO_PIN_INFORMATION_ENTRY, представляющей пин-код.

## DLL-библиотеки

Gpiokd.dll

## См. также

Расширения GPIO

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gpiokd.pinisrvec

Статья • 05.04.2024

Расширение **!gpiokd.pinisrvec** отображает векторные сведения о подпрограмме службы прерываний (ISR) для указанного пин-кода.

```
dbgcmd
```

```
!gpiokd.bankinfo PinAddress
```

## Параметры

*PinAddress*

Адрес структуры данных _GPIO_PIN_INFORMATION_ENTRY, представляющей пин-код.

## DLL-библиотеки

Gpiokd.dll

## См. также

Расширения GPIO

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !gpiokd.pintable

Статья • 05.04.2024

Расширение **!gpiokd.pintable** отображает сведения о массиве закреплений GPIO.

```
dbgcmd
```

```
!gpiokd.pintable PinBase PinCount [Flags]
```

## Параметры

*PinBase*
Адрес массива _GPIO_PIN_INFORMATION_ENTRY структур.

*PinCount*
Число отображаемых закреплений.

*Flags*
Флаги, указывающие, какие сведения отображаются. Этот параметр является битовой или одним или несколькими из следующих флагов.

Развернуть таблицу

| Флаг | Description |
|---|---|
| 0x1 | Не используется этой командой. |
| 0x2 | Не используется этой командой. |
| 0x4 | Отображение включает ненастройанные закрепления. |

## DLL-библиотеки

Gpiokd.dll

## См. также

Расширения GPIO

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения USB 3.0

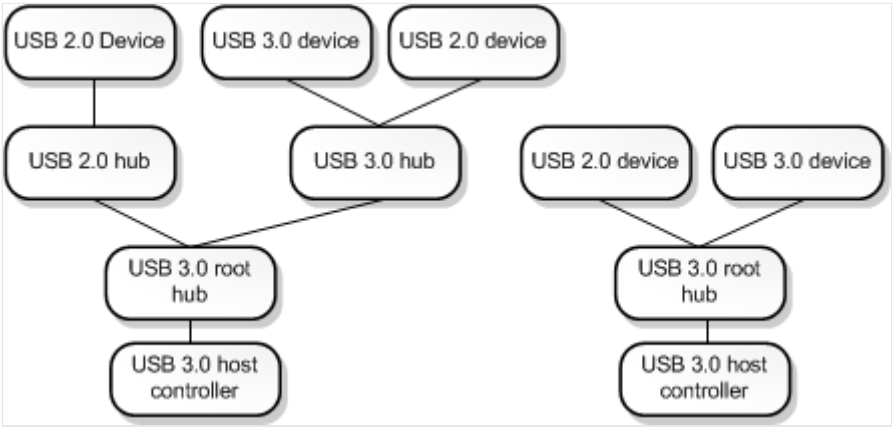
18.07.2025

В этом разделе описываются команды расширения отладчика USB 3.0. Эти команды отображают сведения из структур данных, поддерживаемых тремя драйверами в стеке USB 3.0: драйвер концентратора USB 3.0, драйвер расширения контроллера USB-узла и драйвер контроллера узла USB 3.0. Дополнительные сведения об этих трех драйверах см. в статье о драйверах на стороне USB в Windows. Описание структур данных, используемых драйверами в стеке USB 3.0, см. в разделе "Структуры данных USB 3.0 " и часть 2 инноваций в отладке USB в видео с Windows 8 .

Команды расширения отладчика USB 3.0 реализуются в Usb3kd.dll. To load the Usb3kd commands, enter **.load usb3kd.dll** in the debugger.

# Дерево USB 3.0

Дерево USB 3.0 содержит все контроллеры узла USB 3.0 и все концентраторы и устройства, подключенные к контроллерам узла USB 3.0. На следующей схеме показан пример дерева USB 3.0.




Дерево, показанное на схеме, имеет два контроллера узла USB 3.0. Обратите внимание, что не каждое устройство, показанное на схеме, является устройством USB 3.0. Но все отображаемые устройства (включая концентраторы) являются частью дерева USB 3.0, так как каждое устройство находится в ветви, которая возникает на контроллере узла USB 3.0.

Схему можно рассматривать как два дерева, по одному для каждого контроллера узла. Однако при использовании *дерева USB 3.0* мы ссылаемся на набор всех контроллеров узлов USB 3.0 вместе с подключенными концентраторами и устройствами.

# Начало работы с отладкой USB 3.0

To start debugging a USB 3.0 issue, enter the !usb_tree command. The **!usb_tree** command displays a list of commands and addresses that you can use to investigate host controllers, hubs, ports, devices, endpoints, and other elements of the USB 3.0 tree.

# Hub commands

В следующих командах расширения отображаются сведения о концентраторах, устройствах и портах USB 3.0. Отображаемые сведения основаны на структурах данных, поддерживаемых драйвером концентратора USB 3.0.

- !usb3kd.usb_tree
- !usb3kd.hub_info
- !usb3kd.hub_info_from_fdo
- !usb3kd.device_info
- !usb3kd.device_info_from_pdo
- !usb3kd.port_info

# UCX commands

В следующих командах расширения отображаются сведения о контроллерах узлов USB 3.0, устройствах и портах. Отображаемые сведения основаны на структурах данных, поддерживаемых драйвером расширения контроллера USB-узла.

- !usb3kd.ucx_controller_list
- !usb3kd.ucx_controller
- !usb3kd.ucx_device
- !usb3kd.ucx_endpoint

# Команды контроллера узла

Следующие команды расширения отображают сведения из структур данных, поддерживаемых драйвером контроллера узла USB 3.0.

- !usb3kd.xhci_dumpall
- !usb3kd.xhci_capability
- !usb3kd.xhci_commandring
- !usb3kd.xhci_deviceslots
- !usb3kd.xhci_eventring
- !usb3kd.xhci_registers
- !usb3kd.xhci_resourceusage
- !usb3kd.xhci_trb

- !usb3kd.xhci_transferring
- !usb3kd.xhci_findowner

## Miscellaneous commands

- !usb3kd.usbdstatus
- !usb3kd.urb

## См. также

RCDRKD Extensions

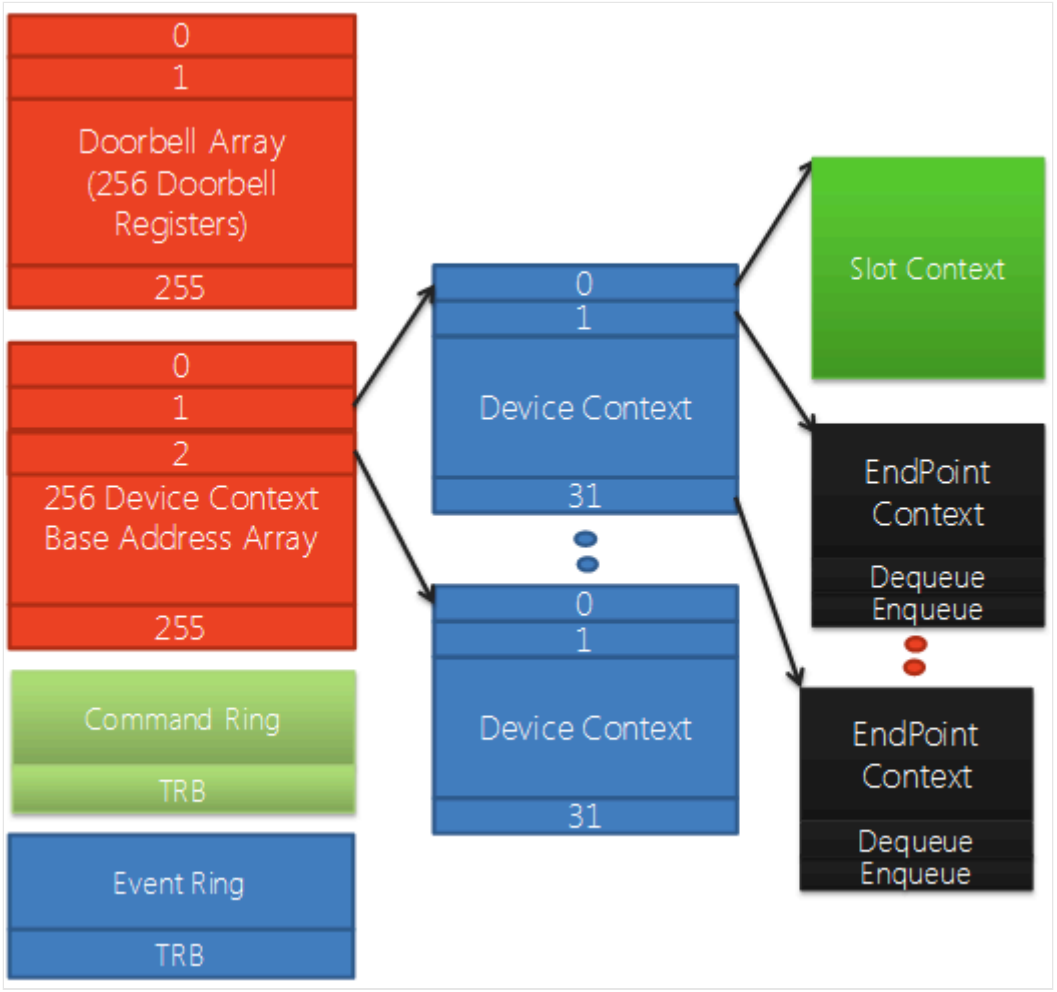
# Структуры данных USB 3.0

18.07.2025

В этом разделе описываются структуры данных, используемые драйвером контроллера узла USB 3.0. Understanding these data structures will help you use the USB 3.0 and RCDRKD debugger extension commands effectively. Структуры данных, представленные здесь, имеют имена, которые соответствуют спецификации USB 3.0. Знакомство со спецификацией USB 3.0 еще больше улучшит возможность использования команд расширения для отладки драйверов USB 3.0.

Драйвер контроллера узла USB 3.0 является частью стека основных драйверов USB 3.0. Дополнительные сведения см. в разделе "Архитектура стека драйверов USB".

Каждый контроллер узла USB 3.0 может содержать до 255 устройств, и каждое устройство может иметь до 31 конечных точек. На следующей схеме показаны некоторые структуры данных, представляющие один контроллер узла и подключенные устройства.




## Базовый массив контекста устройства

Базовый массив контекста устройства — это массив указателей на структуры контекста устройства. Существует одна структура контекста устройства для каждого устройства, подключенного к контроллеру узла. Элементы 1–255 указывают на структуры контекста устройства. Элемент 0 указывает на структуру контекста для контроллера узла.

# Контекст устройства и контекст слота

Структура контекста устройства содержит массив указателей на структуры контекста конечной точки. Существует одна структура контекста конечной точки для каждой конечной точки на устройстве. Элементы 1–31 указывают на структуры контекста конечной точки. Элемент 0 указывает на структуру контекста слота, которая содержит сведения о контексте для устройства.

# Command Ring

Кольцо команд используется программным обеспечением для передачи команд контроллеру узла. Некоторые из этих команд направляются на контроллер узла, и некоторые из них направляются на определенные устройства, подключенные к контроллеру узла.

# Event Ring

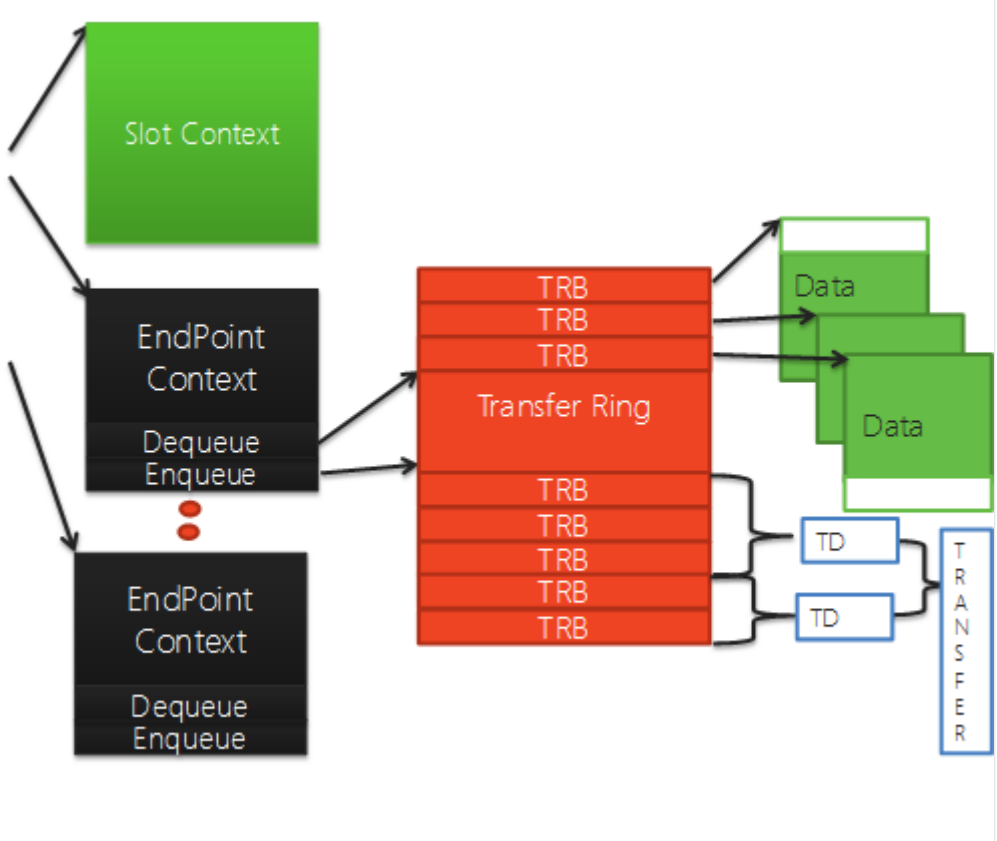
Кольцо событий используется контроллером узла для передачи событий программному обеспечению. То есть кольцо событий — это структура, которую контроллер узла использует для информирования драйверов о завершении действия.

# Массив регистра Doorbell

Массив регистров Doorbell — это массив регистров doorbell, по одному для каждого устройства, подключенного к контроллеру узла. Элементы 1–255 являются регистрами двери. Элемент 0 указывает, существует ли ожидающая команда в кольце команд.

Программное обеспечение уведомляет контроллер узла о том, что у него есть работа, связанная с устройством или конечной точкой, для выполнения путем записи сведений о контексте в регистр двери для устройства.

Следующая схема продолжается справа от предыдущей схемы. В нем показаны дополнительные структуры данных, представляющие одну конечную точку.




# Transfer Ring

Каждая конечная точка имеет один или несколько кругов передачи. Кольцо передачи — это массив блоков запросов передачи (TRB). Каждый TRB указывает на блок смежных данных (до 64 КБ), который будет передаваться между оборудованием и памятью в виде одной единицы.

Когда стек ядра USB 3.0 получает запрос на передачу от usb-клиента, он определяет контекст конечной точки для передачи, а затем разбивает запрос передачи на один или несколько дескрипторов передачи (TD). Каждый TD содержит один или несколько TRB.

# Endpoint Context

Структура контекста конечной точки содержит сведения о контексте для одной конечной точки. It also has **Dequeue** and **Enqueue** members, which are used to track where TRBs are being consumed by the hardware and where TRBs are being added by software.

# See also

Инновации отладки USB в Windows 8

# !usb3kd.help

Статья • 05.04.2024

Команда **!usb3kd.help** отображает справку по командам расширения отладчика USB 3.

## DLL-библиотеки

Usb3kd.dll

## См. также

Расширения USB 3.0

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.device_info

Статья • 05.04.2024

Команда **!usb3kd.device_info** отображает сведения о USB-устройстве в дереве USB 3.0.

dbgcmd

```
!usb3kd.device_info DeviceContext
```

## Параметры

*DeviceContext*
Адрес структуры _DEVICE_CONTEXT, представляющей устройство.

## DLL-библиотеки

Usb3kd.dll

## Замечания

**!device_info и !ucx_device** отображать сведения об устройстве, но отображаемые сведения отличаются. Выходные данные !device_info **с точки зрения драйвера концентратора USB 3.0, а выходные данные !ucx_device** с точки зрения драйвера расширения контроллера USB-узла. Например, выходные **данные !device_info** содержат сведения о дескрипторах конфигурации и интерфейсах, а **выходные данные !ucx_device** содержат сведения о конечных точках.

## Примеры

Вы можете получить адрес структуры контекста устройства, просмотрев выходные данные **команды !usb_tree** . В следующем примере адрес структуры контекста устройства 0xffffa8005abd0c0.

dbgcmd

```
3: kd> !usb_tree

## Dumping HUB Tree - !drvObj 0xffffa800597f770
-----------------------------------------
```

```
Topology
--------
1)  !xhci_info 0fffffa80051d1940  ... - PCI: VendorId ...
    !hub_info 0fffffa8005ad92d0 (ROOT)
    ...
## Enumerated Device List
----------------------
...

2) !device_info 0fffffa8005abd0c0, !devstack fffffa80059c3800
    Current Device State: ConfiguredInD0
    Desc: ... USB Flash Drive
    ...
```

Теперь вы можете передать адрес контекста устройства команде **!device_info** .

dbgcmd

```
3: kd> !device_info 0fffffa8005abd0c0

## Dumping Device Information fffffa8005abd0c0
-----------------------------------------
dt USBHUB3!_DEVICE_CONTEXT 0fffffa8005abd0c0
dt USBHUB3!_HUB_PDO_CONTEXT 0fffffa8005b118d0
!idle_info 0fffffa8005b11920 (dt USBHUB3!_ISM_CONTEXT 0fffffa8005b11920)
Parent !hub_info 0fffffa8005ad92d0 (dt USBHUB3!_HUB_FDO_CONTEXT
0fffffa8005ad92d0)
!port_info 0fffffa8005abe0c0 (dt USBHUB3!_PORT_CONTEXT 0fffffa8005abe0c0)
!ucx_device 0fffffa8005992840 !xhci_deviceslots 0fffffa80051d1940 1

LPMState: U1IsEnabledForUpStreamPort U2IsEnabledForUpStreamPort U1Timeout:
38, U2Timeout: 3
DeviceFlags: HasContainerId NoBOSContainerId Removable HasSerialNumber
MsOsDescriptorNotSupported NoWakeUpSupport DeviceIsSuperSpeedCapable
DeviceHackFlags: WillDisableOnSoftRemove SkipSetIsochDelay
WillResetOnResumeS0 DisableOnSoftRemove

Descriptors:
dt _USB_CONFIGURATION_DESCRIPTOR 0fffffa80053f9250
dt _USB_INTERFACE_DESCRIPTOR 0fffffa80053f9259
ProductId: ... Flash Drive
DeviceDescriptor: VID ... PID ... REV 0a00,  Class: (0)(0) BcdUsb: 0300

UcxRequest: !wdfrequest 0x57ffa662948,
ControlRequest: !wdfrequest 0x57ffa667798, !irp 0fffffa8005997650 !urb
0fffffa8005abd1c0, NumberOfBytes 0
Device working at SuperSpeed
Current Device State: ConfiguredInD0

Device State History: <Event> NewState (<Operation>(),..) :

    [16] <Yes> ConfiguredInD0
```

```
    ...
Device Event History:
    [10] TransferSuccess
    ...
```

## См. также

Расширения USB 3.0

**!usb3kd.device_info_from_pdo**

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.device_info_from_pdo

Статья • 05.04.2024

Команда **!usb3kd.device_info_from_pdo** отображает сведения о USB-устройстве в дереве USB 3.0.

```
dbgcmd
```

```
!usb3kd.device_info_from_pdo DeviceObject
```

## Параметры

*DeviceObject*
Адрес объекта физического устройства (PDO) USB-устройства или концентратора.

## DLL-библиотеки

Usb3kd.dll

## Замечания

**!device_info_from_pdo** и **!ucx_device** отображают сведения об устройстве, но отображаемые сведения отличаются. Выходные данные !device_info_from_pdo с точки зрения драйвера концентратора USB 3.0, а выходные данные **!ucx_device** с точки зрения драйвера расширения контроллера USB-узла. Например, выходные **данные !device_info_from_pdo** содержат сведения о дескрипторах конфигурации и интерфейсах, а **выходные данные !ucx_device** включают сведения о конечных точках.

## Примеры

Адрес PDO можно получить из выходных данных **!usb_tree** или из различных других команд отладчика. Например, **команда !devnode** отображает адреса PDOS. В следующем примере узел устройства USBSTOR является прямым дочерним элементом узла USBHUB3. Адрес PDO для узла USBSTOR 0xfffffa80059c3800.

```
dbgcmd
```

```
3: kd> !devnode 0 1 usbhub3

Dumping IopRootDeviceNode (= 0xfffffa8003609cc0)
DevNode 0xfffffa8005981730 for PDO 0xfffffa8004ffc550
  InstancePath is "USB\ROOT_HUB30\5&11db9684&0&0"
  ServiceName is "USBHUB3"
  State = DeviceNodeStarted (0x308)
  Previous State = DeviceNodeEnumerateCompletion (0x30d)

  DevNode 0xfffffa8005a546a0 for PDO 0xfffffa80059c3800
    InstancePath is "USB\VID_125F&PID_312A\0902100000000000342192873"
    ServiceName is "USBSTOR"
    State = DeviceNodeStarted (0x308)
    Previous State = DeviceNodeEnumerateCompletion (0x30d)

    DevNode 0xfffffa8005a09730 for PDO 0xfffffa8005b3a650
      InstancePath is "USBSTOR\Disk&Ven ...
      ServiceName is "disk"
      State = DeviceNodeStarted (0x308)
      Previous State = DeviceNodeEnumerateCompletion (0x30d)
```

Теперь вы можете передать адрес PDO **в команду !usb3kd.device_info_from_pdo** .

dbgcmd

```
3: kd> !device_info_from_pdo 0xfffffa80059c3800

## Dumping Device Information fffffa80059c3800
-----------------------------------------
!devobj 0xfffffa8004ffc550 (Root HUB)
!device_info 0xfffffa8005abd0c0 (dt usbhub3!_DEVICE_CONTEXT
0xfffffa8005abd0c0)
dt USBHUB3!_DEVICE_CONTEXT 0xfffffa8005abd0c0
dt USBHUB3!_HUB_PDO_CONTEXT 0xfffffa8005b118d0
!idle_info 0xfffffa8005b11920 (dt USBHUB3!_ISM_CONTEXT 0xfffffa8005b11920)
Parent !hub_info 0xfffffa8005ad92d0 (dt USBHUB3!_HUB_FDO_CONTEXT
0xfffffa8005ad92d0)
!port_info 0xfffffa8005abe0c0 (dt USBHUB3!_PORT_CONTEXT 0xfffffa8005abe0c0)
!ucx_device 0xfffffa8005992840 !xhci_deviceslots 0xfffffa80051d1940 1

LPMState: U1IsEnabledForUpStreamPort U2IsEnabledForUpStreamPort U1Timeout:
38, U2Timeout: 3
DeviceFlags: HasContainerId NoBOSContainerId Removable HasSerialNumber
MsOsDescriptorNotSupported NoWakeUpSupport DeviceIsSuperSpeedCapable
DeviceHackFlags: WillDisableOnSoftRemove SkipSetIsochDelay
WillResetOnResumeS0 DisableOnSoftRemove

Descriptors:
dt _USB_CONFIGURATION_DESCRIPTOR 0xfffffa80053f9250
dt _USB_INTERFACE_DESCRIPTOR 0xfffffa80053f9259
ProductId: ...
DeviceDescriptor: VID ...
```

```
UcxRequest: !wdfrequest 0x57ffa662948,
ControlRequest: !wdfrequest 0x57ffa667798, !irp 0xfffffa8005997650 !urb
0xfffffa8005abd1c0, NumberOfBytes 0
Device working at SuperSpeed
Current Device State: ConfiguredInD0

Device State History: <Event> NewState (<Operation>(),..) :

    [16] <Yes> ConfiguredInD0
    ...
Device Event History:
    [10] TransferSuccess
    ...
```

В следующем примере показаны некоторые выходные данные **команды !usb_tree** . Адрес PDO одного из дочерних узлов устройства отображается в качестве аргумента **команды !devstack** . (**!devstack fffa80059c3800**)

dbgcmd

```
3: kd> !usb_tree

## Dumping HUB Tree - !drvObj 0xfffffa800597f770
-----------------------------------------

Topology
--------
1)  !xhci_info 0xfffffa80051d1940  ... - PCI: VendorId ...
    !hub_info 0xfffffa8005ad92d0 (ROOT)
        !port_info 0xfffffa8005a5ca80 !device_info 0xfffffa8005b410c0 Desc:
<none> Speed: High
        ...
## Enumerated Device List
----------------------
...

2) !device_info 0xfffffa8005abd0c0, !devstack fffffa80059c3800
    Current Device State: ConfiguredInD0
    Desc: ... Flash Drive
    USB\VID_...
    !ucx_device 0xfffffa8005992840 !xhci_deviceslots 0xfffffa80051d1940 1
```

# См. также

Расширения USB 3.0

**!usb3kd.device_info**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.dsf

Статья • 05.04.2024

**Расширение !usb3kd.dsf** — это команда переключателя, которая задает контекст отладчика для отладки драйвера узла DSF.

dbgcmd

```
!usb3kd.dsf
```

## DLL-библиотеки

Usb3kd.dll

## См. также

Расширения USB 3.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.hub_info

Статья • 05.04.2024

Команда **!usb3kd.device_info** отображает сведения о концентраторе в дереве USB 3.0.

dbgcmd

```
!usb3kd.hub_info DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства концентратора (FDO).

## DLL-библиотеки

Usb3kd.dll

## Примеры

Чтобы получить адрес расширения устройства, просмотрите выходные данные **команды !usb_tree** . В следующем примере адрес расширения устройства для корневого концентратора 0xfffffa8005ad92d0.

dbgcmd

```
3: kd> !usb_tree

## Dumping HUB Tree - !drvObj 0xfffffa800597f770
-----------------------------------------
Topology
--------
1)  !xhci_info 0xfffffa80051d1940  ... - PCI: VendorId 0x1033 ...
    !hub_info 0xfffffa8005ad92d0 (ROOT)
        !port_info 0xfffffa8005a5ca80 !device_info 0xfffffa8005b410c0 Desc:
<none> Speed: High
        ...
```

Теперь вы можете передать адрес расширения устройства команде **!hub_info** .

dbgcmd

```
3: kd> !hub_info fffffa8005ad92d0

## Dumping HUB Information fffffa8005ad92d0
---------------------------------------
dt USBHUB3!_HUB_FDO_CONTEXT 0xfffffa8005ad92d0
!rcdrlogdump usbhub3 -a 0xfffffa8005ad8010
Capabilities: Initialized, Configured, HasOverCurrentProtection, ...

Total number of ports: 4, HUB depth is 0
Number of 3.0 ports: 2 (Range 1 - 2)
Number of 2.0 ports: 2 (Range 3 - 4)

Descriptors:
    dt _USB_HUB_DESCRIPTOR 0xfffffa8005ad9728
    dt _USB_30_HUB_DESCRIPTOR 0xfffffa8005ad9728
    dt _USB_DEVICE_DESCRIPTOR 0xfffffa8005ad92d0
    dt _USB_CONFIGURATION_DESCRIPTOR 0xfffffa8005ad9770

List of PortContext: 4
    [1] !port_info 0xfffffa8005a5ca80    !device_info 0xfffffa8005b410c0
            Last Port Status(2.0): Connected Enabled Powered
HighSpeedDeviceAttached
            Last Port Change: <none>
            Current Port(2.0) State:
ConnectedEnabled.WaitingForPortChangeEvent
            Current Device State: ConfiguredInD0
    ...

Current Hub State: ConfiguredWithIntTransfer

Hub State History: <Event> NewState (<Operation>(),..) :

    [43] <OperationSuccess> ConfiguredWithIntTransfer
    ...
Hub Event History:
    [ 0] PortEnableInterruptTransfer
    ...
```

# См. также

Расширения USB 3.0

**!usb3kd.hub_info_from_fdo**

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

-

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.hub_info_from_fdo

Статья • 05.04.2024

Команда **!usb3kd.hub_info_from_fdo** отображает сведения о концентраторе в дереве USB 3.0.

dbgcmd

```
!usb3kd.hub_info_from_fdo DeviceObject
```

## Параметры

*DeviceObject*
Адрес объекта функционального устройства (FDO), представляющего концентратор.

## DLL-библиотеки

Usb3kd.dll

## Примеры

Адрес FDO можно получить из выходных данных **!usb_tree** или из различных других команд отладчика. Например, **команда !devstack** отображает адрес FDO. В следующем примере адрес FDO — fffffa800597a660.

dbgcmd

```
3: kd> !devnode 0 1 usbhub3
Dumping IopRootDeviceNode (= 0xfffffa8003609cc0)
DevNode 0xfffffa8005981730 for PDO 0xfffffa8004ffc550
...
3: kd> !devstack 0xfffffa8004ffc550
  !DevObj   !DrvObj            !DevExt   ObjectName
  fffffa800597a660  \Driver\USBHUB3    fffffa8005ad92d0
> fffffa8004ffc550  \Driver\USBXHCI    fffffa8005251d00  USBPDO-0
...
```

Теперь можно передать адрес FDO **в команду !usb3kd.hub_info_from_fdo** .

dbgcmd

```
 3: kd> !hub_info_from_fdo fffffa800597a660

## Dumping HUB Information fffffa800597a660
---------------------------------------
dt USBHUB3!_HUB_FDO_CONTEXT 0xfffffa8005ad92d0
!rcdrlogdump usbhub3 -a 0xfffffa8005ad8010
Capabilities: Initialized, Configured, HasOverCurrentProtection,
SelectiveSuspendSupportedByParentStack, CannotWakeOnConnect, NotArmedForWake

Total number of ports: 4, HUB depth is 0
Number of 3.0 ports: 2 (Range 1 - 2)
Number of 2.0 ports: 2 (Range 3 - 4)

Descriptors:
    dt _USB_HUB_DESCRIPTOR 0xfffffa8005ad9728
    dt _USB_30_HUB_DESCRIPTOR 0xfffffa8005ad9728
    dt _USB_DEVICE_DESCRIPTOR 0xfffffa8005ad92d0
    dt _USB_CONFIGURATION_DESCRIPTOR 0xfffffa8005ad9770

List of PortContext: 4
    [1] !port_info 0xfffffa8005a5ca80    !device_info 0xfffffa8005b410c0
            Last Port Status(2.0): Connected Enabled Powered
HighSpeedDeviceAttached
            Last Port Change: <none>
            Current Port(2.0) State:
ConnectedEnabled.WaitingForPortChangeEvent
            Current Device State: ConfiguredInD0
         ...

Current Hub State: ConfiguredWithIntTransfer

Hub State History: <Event> NewState (<Operation>(),..) :

    [43] <OperationSuccess> ConfiguredWithIntTransfer
    ...

Hub Event History:
    [ 0] PortEnableInterruptTransfer
    ...
```

# См. также

Расширения USB 3.0

**!usb3kd.hub_info**

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.port_info

Статья • 05.04.2024

Команда **!usb3kd.port_info** отображает сведения о USB-порту в дереве USB 3.0.

```
dbgcmd
```

```
!usb3kd.port_info PortContext
```

## Параметры

*PortContext*
Адрес структуры _PORT_CONTEXT.

## DLL-библиотеки

Usb3kd.dll

## Примеры

Чтобы получить адрес контекста порта, просмотрите выходные данные **команды !usb_tree** . В следующем примере адрес контекста порта 0xfffffa8005abe0c0.

```
dbgcmd
```

```
3: kd> !usb_tree

## Dumping HUB Tree - !drvObj 0xfffffa800597f770
-----------------------------------------

Topology
--------
1)  !xhci_info 0xfffffa80051d1940  ... - PCI: VendorId ...
    !hub_info 0xfffffa8005ad92d0 (ROOT)
        ...
        !port_info 0xfffffa8005abe0c0 !device_info 0xfffffa8005abd0c0 Desc:
... USB Flash Drive Speed: Super
```

Теперь можно передать адрес контекста порта команде **!port_info** .

```
dbgcmd
```

```
3: kd> !port_info 0xfffffa8005abe0c0

## Dumping Port Context 0xfffffa8005abe0c0
--------------------------------------
dt USBHUB3!_PORT_CONTEXT 0xfffffa8005abe0c0
!hub_info 0xfffffa8005ad92d0 (dt _HUB_FDO_CONTEXT 0xfffffa8005ad92d0)
!device_info 0xfffffa8005abd0c0 (dt _DEVICE_CONTEXT 0xfffffa8005abd0c0)
!rcdrlogdump usbhub3 -a 0xfffffa8005abf6b0

PortNumber: 2
Last Port Status(3.0): Connected Enabled Powered
Last Port Change: <none>

CurrentPortEvent: PsmEventPortConnectChange
Current Port(3.0) State: ConnectedEnabled.WaitingForPortChangeEvent

Port(3.0) State History: <Event> NewState (<Operation>(),..) :

    [34] <Push> WaitingForPortChangeEvent
    ...
Port Event History:
    [ 8] TransferSuccess
    ...
```

## См. также

Расширения USB 3.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.ucx_device

Статья • 05.04.2024

Расширение **!usb3kd.ucx_device** отображает сведения о USB-устройстве в дереве USB 3.0. Отображение основано на структурах данных, поддерживаемых драйвером расширения контроллера USB (версия *Ucx*.sys).

dbgcmd

```
!usb3kd.ucx_device UcxUsbDevicePrivContext
```

## Параметры

*UcxUsbDevicePrivContext*
Адрес структуры _UCXUSBDEVICE_PRIVCONTEXT, представляющей устройство.

## DLL-библиотеки

Usb3kd.dll

## Замечания

Драйвер расширения контроллера USB (версия *Ucx*.sys) предоставляет уровень абстракции между драйвером концентратора USB 3.0 и драйвером контроллера узла USB 3.0. Драйвер расширения имеет собственное представление контроллеров узлов, устройств и конечных точек. Выходные данные **команды !ucx_device** основаны на структурах данных, поддерживаемых драйвером расширения. Дополнительные сведения о драйвере расширения контроллера USB-узла и драйвере контроллера узла USB 3.0 см. в разделе "Архитектура стека драйверов USB".

**!ucx_device и !device_info** отображать сведения об устройстве, но отображаемые сведения отличаются. Выходные данные !ucx_device с точки зрения драйвера расширения контроллера USB-узла, а выходные данные **!device_info** с точки зрения драйвера концентратора USB 3.0. Например, выходные **данные !ucx_device включают сведения о конечных точках, а выходные данные !device_info** включают сведения о дескрипторах конфигурации и интерфейса.

# Примеры

Чтобы получить адрес частного контекста USB-устройства UCX, просмотрите выходные данные **команды !ucx_controller_list** . В следующем примере адрес частного контекста для второго устройства 0xfffffa8005bd9680.

dbgcmd

```
3: 3: kd> !ucx_controller_list

## Dumping List of UCX controller objects
-------------------------------------
[1] !ucx_controller 0xfffffa80052da050 (dt
ucx01000!_UCXCONTROLLER_PRIVCONTEXT fffffa80052da050)
    !ucx_device 0xfffffa8005a41840
        .!ucx_endpoint 0xfffffa800533f3d0 [Blk In ], UcxEndpointStateEnabled
        ...
    !ucx_device 0xfffffa8005bd9680
        .!ucx_endpoint 0xfffffa8003694860 [Blk Out], UcxEndpointStateEnabled
        ...
```

Теперь вы можете передать адрес частного контекста USB UCX в **команду !ucx_device** .

dbgcmd

```
3: kd> !ucx_device 0xfffffa8005bd9680

## Dumping Ucx USB Device Information fffffa8005bd9680
-----------------------------------------------------
dt ucx01000!_UCXUSBDEVICE_PRIVCONTEXT 0xfffffa8005bd9680
!ucx_controller 0xfffffa80052da050
ParentHub: !wdfhandle 0x57ffacbce78
DefaultEndpoint: !ucx_endpoint 0xfffffa8005be0550
ListOfEndpionts:
    .!ucx_endpoint 0xfffffa8003694860 [Blk Out], UcxEndpointStateEnabled
    .!ucx_endpoint 0xfffffa8003686820 [Blk In ], UcxEndpointStateEnabled
    .!ucx_endpoint 0xfffffa8005be0550 [Control], UcxEndpointStateEnabled
    .!ucx_endpoint 0xfffffa8003695580 [Blk In ], UcxEndpointStateStale
    .!ucx_endpoint 0xfffffa80036a20c0 [Blk Out], UcxEndpointStateStale

EventCallbacks:
    EvtUsbDeviceEndpointsConfigure: (0xfffff880044d1164)
USBXHCI!UsbDevice_UcxEvtEndpointsConfigure
    EvtUsbDeviceEnable: (0xfffff880044cffac) USBXHCI!UsbDevice_UcxEvtEnable
    EvtUsbDeviceDisable: (0xfffff880044d1cbc)
USBXHCI!UsbDevice_UcxEvtDisable
    EvtUsbDeviceReset: (0xfffff880044d2178) USBXHCI!UsbDevice_UcxEvtReset
    EvtUsbDeviceAddress: (0xfffff880044d0934)
USBXHCI!UsbDevice_UcxEvtAddress
    EvtUsbDeviceUpdate: (0xfffff880044d0c80) USBXHCI!UsbDevice_UcxEvtUpdate
```

```
    EvtUsbDeviceDefaultEndpointAdd: (0xffff880044ede1c)
USBXHCI!Endpoint_UcxEvtUsbDeviceDefaultEndpointAdd
    EvtUsbDeviceEndpointAdd: (0xffff880044edfc8)
USBXHCI!Endpoint_UcxEvtUsbDeviceEndpointAdd
```

## См. также

Расширения USB 3.0

**!usb3kd.ucx_controller_list**

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.ucx_endpoint

Статья • 05.04.2024

Команда **!usb3kd.ucx_endpoint** отображает сведения о конечной точке на USB-устройстве в дереве USB 3.0. Отображение основано на структурах данных, поддерживаемых драйвером расширения контроллера USB (версия *Ucx*.sys).

dbgcmd

```
!usb3kd.ucx_endpoint UcxEndpointPrivContext
```

## Параметры

*UcxEndpointPrivContext*
Адрес структуры _UCXENDPOINT_PRIVCONTEXT, представляющей конечную точку.

## DLL-библиотеки

Usb3kd.dll

## Замечания

Драйвер расширения контроллера USB (версия *Ucx*.sys) предоставляет уровень абстракции между драйвером концентратора USB 3.0 и драйвером контроллера узла USB 3.0. Драйвер расширения имеет собственное представление контроллеров узлов, устройств и конечных точек. Выходные данные **команды !ucx_endpoint** основаны на структурах данных, поддерживаемых драйвером расширения. Дополнительные сведения о драйвере расширения контроллера USB-узла и драйвере контроллера узла USB 3.0 см. в разделе "Архитектура стека драйверов USB".

## Примеры

Чтобы получить адрес частного контекста конечной точки UCX, просмотрите выходные данные **команды !ucx_controller_list** . В следующем примере адрес частного контекста для первой конечной точки на втором устройстве 0xfffffa8003694860.

dbgcmd

```
3: kd> !ucx_controller_list

## Dumping List of UCX controller objects
-------------------------------------
[1] !ucx_controller 0xfffffa80052da050 (dt
ucx01000!_UCXCONTROLLER_PRIVCONTEXT fffffa80052da050)
    !ucx_device 0xfffffa8005a41840
        .!ucx_endpoint 0xfffffa800533f3d0 [Blk In ], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa80053405d0 [Blk Out], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8005a3f710 [Control], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8005bbe4e0 [Blk Out], UcxEndpointStateStale
        .!ucx_endpoint 0xfffffa8005ac4810 [Blk In ], UcxEndpointStateStale
    !ucx_device 0xfffffa8005bd9680
        .!ucx_endpoint 0xfffffa8003694860 [Blk Out], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8003686820 [Blk In ], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8005be0550 [Control], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8003695580 [Blk In ], UcxEndpointStateStale
        .!ucx_endpoint 0xfffffa80036a20c0 [Blk Out], UcxEndpointStateStale
```

Теперь можно передать адрес частного контекста конечной точки UCX команде **!ucx_endpoint** .

dbgcmd

```
3: kd> !ucx_endpoint 0xfffffa8003694860

## Dumping Ucx USB Endpoint Information fffffa8003694860
-----------------------------------------------------
dt ucx01000!_UCXENDPOINT_PRIVCONTEXT 0xfffffa8003694860
[Blk Out], UcxEndpointStateEnabled, MaxTransferSize: 4194304
Endpoint Address: 0x02
Endpoint Queue: !wdfqueue 0x57ffc969888

UcxEndpoint State History: <Event> NewState
    [ 3] <UcxEndpointEventOperationSuccess> UcxEndpointStateEnabled
    [ 2] <UcxEndpointEventYes> UcxEndpointStateCompletingPendingOperation1
    [ 1] <UcxEndpointEventEnableComplete> UcxEndpointStateIsAbleToStart2
    [ 0] <SmEngineEventStart> UcxEndpointStateCreated

UcxEndpoint Event History:
    [ 1] UcxEndpointEventEnableComplete
    [ 0] SmEngineEventStart

EventCallbacks:
    EvtEndpointPurge: (0xfffff880044ba6e8)
USBXHCI!Endpoint_UcxEvtEndpointPurge
    EvtEndpointAbort: (0xfffff880044ba94c)
USBXHCI!Endpoint_UcxEvtEndpointAbort
```

```
    EvtEndpointReset: (0xffff880044bb854)
USBXHCI!Endpoint_UcxEvtEndpointReset
```

## См. также

Расширения USB 3.0

**!usb3kd.ucx_controller_list**

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.ucx_controller

Статья • 05.04.2024

Команда **!usb3kd.ucx_controller** отображает сведения о контроллере узла USB 3.0. Отображение основано на структурах данных, поддерживаемых драйвером расширения контроллера USB (версия *Ucx*.sys).

```
dbgcmd
```

```
!usb3kd.ucx_controller UcxControllerPrivContext
```

## Параметры

*UcxControllerPrivContext*
Адрес структуры _UCXCONTROLLER_PRIVCONTEXT, представляющей контроллер.

## DLL-библиотеки

Usb3kd.dll

## Замечания

Драйвер расширения контроллера USB (версия *Ucx*.sys) предоставляет уровень абстракции между драйвером концентратора USB 3.0 и драйвером контроллера узла USB 3.0. Драйвер расширения имеет собственное представление контроллеров узлов, устройств и конечных точек. Выходные данные **команды !ucx_controller** основаны на структурах данных, поддерживаемых драйвером расширения. Дополнительные сведения о драйвере расширения контроллера USB-узла и драйвере контроллера узла USB 3.0 см. в разделе "Архитектура стека драйверов USB".

## Примеры

Чтобы получить адрес частного контекста контроллера UCX, просмотрите выходные данные **команды !ucx_controller_list** . В следующем примере адрес частного контекста 0xfffffa80052da050.

```
dbgcmd
```

```
3: kd> !ucx_controller_list

## Dumping List of UCX controller objects
-------------------------------------
[1] !ucx_controller 0xfffffa80052da050 (dt
ucx01000!_UCXCONTROLLER_PRIVCONTEXT fffffa80052da050)
    !ucx_device 0xfffffa8005a41840
        .!ucx_endpoint 0xfffffa800533f3d0 [Blk In ], UcxEndpointStateEnabled
        ...
    !ucx_device 0xfffffa8005bd9680
        .!ucx_endpoint 0xfffffa8003694860 [Blk Out], UcxEndpointStateEnabled
        ...
```

Теперь вы можете передать адрес частного контекста контроллера UCX команде **!ucx_controller** .

dbgcmd

```
3: kd> !ucx_controller 0xfffffa80052da050

## Dumping Ucx Controller Information fffffa80052da050
-----------------------------------------------------
dt ucx01000!_UCXCONTROLLER_PRIVCONTEXT 0xfffffa80052da050
Parent Device: !wdfdevice 0x57ffac91fd8
Controller Queues:
    Default               : !wdfqueue 0x57ffacc5fd8
    Address'0'Ownership   : !wdfqueue 0x57ffad5ad88
    DeviceManagement      : !wdfqueue 0x57ffacd6fd8
    ... pend on Ctrl Reset: !wdfqueue 0x57ffad48fd8

Controller Reset State History: <Event> NewState
    [ 2] <ControllerResetEventOperationSuccess>
ControllerResetStateRHPdoInD0
    [ 1] <ControllerResetEventRHPdoEnteredD0>
ControllerResetStateStopBlockingResetComplete1
    [ 0] <SmEngineEventStart> ControllerResetStateRhPdoInDx

Controller Reset Event History:
    [ 1] ControllerResetEventRHPdoEnteredD0
    [ 0] SmEngineEventStart

Root Hub PDO: !wdfdevice 0x57ffaf4daa8
Number of 2.0 Ports: 2
Number of 3.0 Ports: 2
RootHub Control !wdfqueue 0x57ffacb4798
RootHub Interrupt !wdfqueue 0x57ffad033f8, pending !wdfequest 0x57ffa5fe998

Device Tree:
    !ucx_device 0xfffffa8005a41840
        .!ucx_endpoint 0xfffffa800533f3d0 [Blk In ], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa80053405d0 [Blk Out], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8005a3f710 [Control], UcxEndpointStateEnabled
```

```
        .!ucx_endpoint 0xfffffa8005bbe4e0 [Blk Out], UcxEndpointStateStale
        .!ucx_endpoint 0xfffffa8005ac4810 [Blk In ], UcxEndpointStateStale
    !ucx_device 0xfffffa8005bd9680
        .!ucx_endpoint 0xfffffa8003694860 [Blk Out], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8003686820 [Blk In ], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8005be0550 [Control], UcxEndpointStateEnabled
        .!ucx_endpoint 0xfffffa8003695580 [Blk In ], UcxEndpointStateStale
        .!ucx_endpoint 0xfffffa80036a20c0 [Blk Out], UcxEndpointStateStale
```

## См. также

Расширения USB 3.0

**!usb3kd.ucx_controller_list**

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.ucx_controller_list

Статья • 05.04.2024

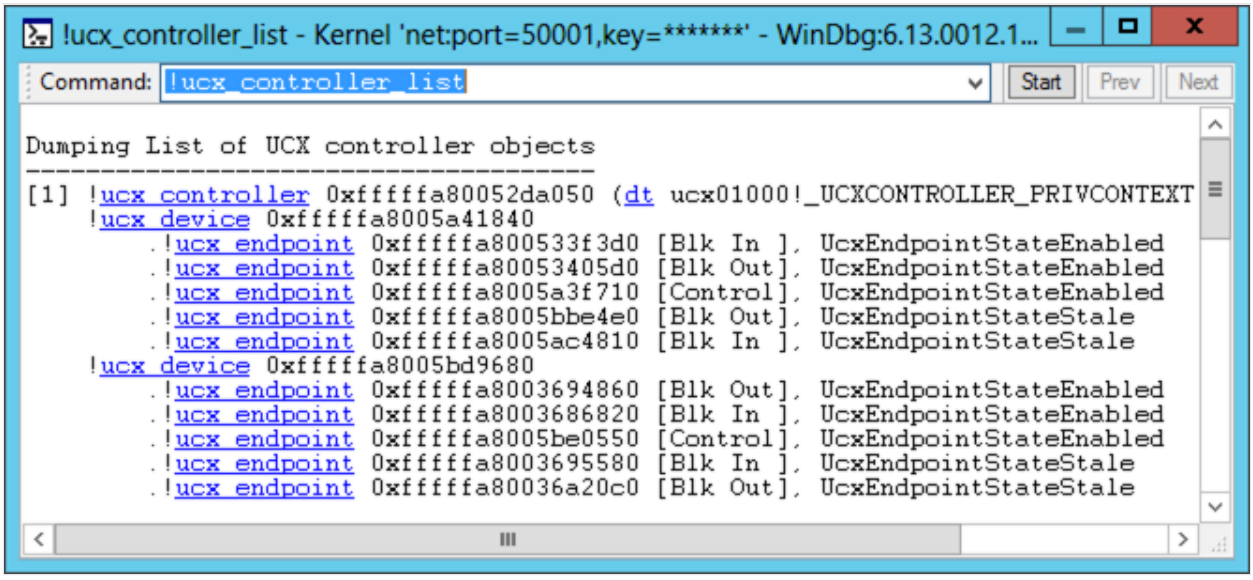
Команда **!usb3kd.ucx_controller_list** отображает сведения обо всех контроллерах узла USB 3.0 на компьютере. Отображение основано на структурах данных, поддерживаемых драйвером расширения контроллера USB (версия *Ucx*.sys).

dbgcmd

```
!usb3kd.ucx_controller_list
```

## Примеры

На следующем снимке экрана показаны выходные данные **команды !ucx_controller_list** .




Выходные данные показывают, что есть один контроллер узла USB 3.0, представленный строкой, начинающейся с !ucx_controller**.** Вы увидите, что два устройства подключены к контроллеру и что каждое устройство имеет четыре конечные точки.

Выходные данные используют язык разметки отладчика (DML) для предоставления ссылок. Ссылки выполняют команды, которые предоставляют подробные сведения об отдельных устройствах или конечных точках. Например, вы можете получить подробные сведения о конечной точке, щелкнув одну из **ссылок !ucx_endpoint** . В качестве альтернативы щелчку ссылки можно ввести команду. Например, чтобы просмотреть сведения о первой конечной точке второго устройства, можно ввести команду **!ucx_endpoint 0xfffffa8003694860**.

**Обратите внимание** , что функция DML доступна в WinDbg, но не в Visual Studio или KD.

## DLL-библиотеки

Usb3kd.dll

## Замечания

**Команда !ucx_controller_list** является родительской командой для этого набора команд.

- **!ucx_controller**
- **!ucx_device**
- **!ucx_endpoint**

Драйвер расширения контроллера USB (версия *Ucx*.sys) предоставляет уровень абстракции между драйвером концентратора USB 3.0 и драйвером контроллера узла USB 3.0. Драйвер расширения имеет собственное представление контроллеров узлов, устройств и конечных точек. Выходные данные команд в **семействе !ucx_controller_list** основаны на структурах данных, поддерживаемых драйвером расширения. Дополнительные сведения о драйвере расширения контроллера USB-узла и драйвере контроллера узла USB 3.0 см. в разделе "Архитектура стека драйверов USB". Описание структур данных, используемых драйверами в стеке USB 3.0, см. в разделе "Часть 2 инноваций в отладке USB" в видео с Windows 8 .

## См. также

Расширения USB 3.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.usbanalyze

Статья • 08.07.2024

Расширение **!usb3kd.usbanalyze** анализирует проверку ошибок USB 3.0.

```
dbgcmd
```

```
!usb3kd.usbanalyze [-v]
```

## Параметры

**-v**
Отображение подробно.

## DLL-библиотеки

Usb3kd.dll

## См. также

Расширения USB 3.0

Драйверы универсальной последовательной шины (USB)

**BUGCODE_USB3_DRIVER**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.usbdstatus

Статья • 05.04.2024

Расширение **!usb3kd.usbdstatus** отображает имя кода состояния USBD.

```
dbgcmd

!usb3kd.ucx_usbdstatus UrbStatus
```

## Параметры

*UsbdStatus*
Числовое значение кода состояния USBD.

## DLL-библиотеки

Usb3kd.dll

## Замечания

Коды состояния USBD определены в USB.h.

## Примеры

Следующий пример передает числовое значение 0x80000200 команде **!usbdstatus** . Команда возвращает имя кода состояния USBD_STATUS_INVALID_URB_FUNCTION.

```
dbgcmd

3: kd> !usbdstatus 0x80000200
USBD_STATUS_INVALID_URB_FUNCTION (0x80000200)
```

## См. также

Расширения USB 3.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.usb_tree

Статья • 05.04.2024

Расширение **!usb3kd.usb_tree** отображает сведения в виде дерева обо всех контроллерах, концентраторах и устройствах USB 3.0 на компьютере.

dbgcmd

```
!usb3kd.usb_tree [1]
```

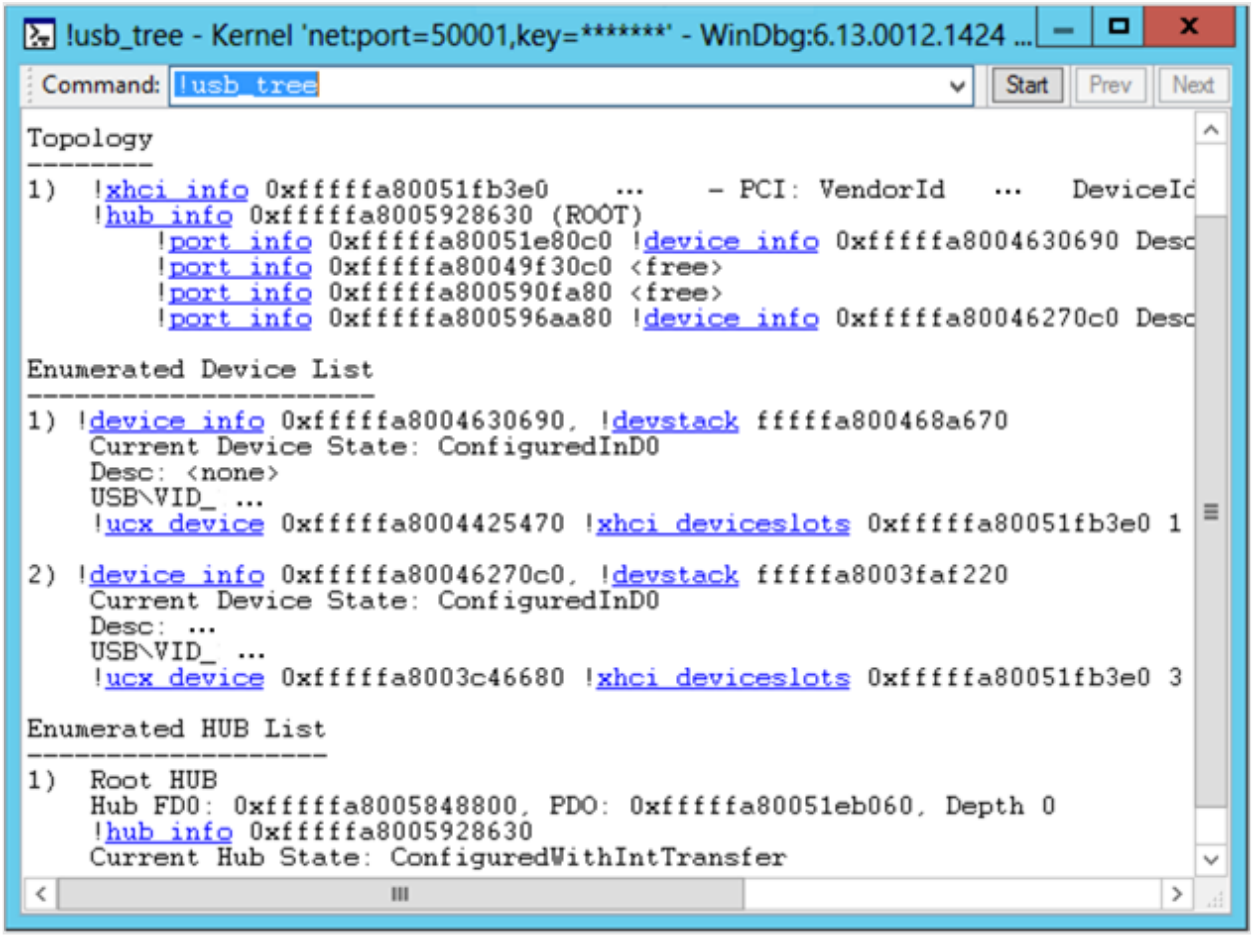
## Параметры

1

Отображение содержит сведения о состоянии для концентраторов и портов.

## Примеры

На следующем снимке экрана показаны выходные данные **команды !usb_tree** .



Выходные данные показывают, что есть один контроллер узла USB 3.0, который представлен строкой, начинающейся с !xhci_info**.** Следующая строка представляет корневой концентратор для контроллера узла. Следующие четыре строки представляют порты, связанные с корневым концентратором. Вы увидите, что два порта подключены к устройствам.

Выходные данные используют язык разметки отладчика (DML) для предоставления ссылок. Ссылки выполняют команды, предоставляющие подробные сведения об отдельных объектах в дереве. Например, можно получить сведения об одном из подключенных устройств, щелкнув одну из **ссылок !device_info** . В качестве альтернативы щелчку ссылки можно ввести команду. Например, чтобы просмотреть сведения о первом подключенном устройстве, можно ввести команду **!device_info 0xfffffa8004630690**.

**Обратите внимание** , что функция DML доступна в WinDbg, но не в Visual Studio или KD.

# DLL-библиотеки

Usb3kd.dll

# Замечания

**Команда !usb_tree** является родительской командой для этого набора команд.

- **!hub_info**
- **!hub_info_from_fdo**
- **!device_info**
- **!device_info_from_pdo**
- **!port_info**

Сведения, отображаемые семейством **команд !usb_tree** , основаны на структурах данных, поддерживаемых драйвером концентратора USB 3.0. Сведения о драйвере концентратора USB 3.0 и других драйверах в стеке USB 3.0 см. в статье "Архитектура стека драйверов USB". Описание структур данных, используемых драйверами в стеке USB 3.0, см. в разделе "Часть 2 инноваций в отладке USB" в видео с Windows 8 .

# См. также

Расширения USB 3.0

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.urb

Статья • 05.04.2024

Расширение **!usb3kd.urb** отображает сведения о блоке USB-запроса (URB).

dbgcmd

```
!usb3kd.urb UrbAddress
```

## Параметры

*UrbAddress*
Адрес URB.

## DLL-библиотеки

Usb3kd.dll

## Примеры

В следующем примере показан адрес URB (0xfffffa8005a2cbe8) в выходных данных **команды !xhci_deviceslots** .

dbgcmd

```
3: kd> !xhci_deviceslots 0fffffa800520d2d0

## Dumping dt _DEVICESLOT_DATA 0fffffa8003612e80
-----------------------------------------------
DeviceContextBase: VA 0fffffa8005a64000 LA 0x116864000 !wdfcommonbuffer
0x57ffa7ca758 Size 4096

## [1] SlotID : dt USBXHCI!_USBDEVICE_DATA 0fffffa80059027d0 dt
_SLOT_CONTEXT32 0fffffa8005a65000
-------------------------------------------------------------------------
--------------------
    USB\VID_...
    SlotEnabled IsDevice NumberOfTTs 0 TTThinkTime 0
    ...
            PendingTransferList:
                [0] dt _TRANSFER_DATA 0fffffa80059727f0 !urb
0xfffffa8005a2cbe8 !wdfrequest 0x57ffa68d998 TransferState_Pending
    ...
```

В следующем примере адрес URB передается команде **!usb3kd.urb** .

dbgcmd

```
3: kd> !urb 0xfffffa8005a2cbe8

## Dumping URB 0xfffffa8005a2cbe8
------------------------------
Function:         URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER (0x9)
UsbdDeviceHandle:
    !ucx_device 0xfffffa8005901840
    !xhci_deviceslots 0xfffffa800520d2d0 1

Status:           USBD_STATUS_PENDING (0x40000000)
UsbdFlags:        (0x0)
dt _URB_BULK_OR_INTERRUPT_TRANSFER 0xfffffa8005a2cbe8
PipeHandle:             0xfffffa800596f720
TransferFlags:          (0x1) USBD_TRANSFER_DIRECTION_IN
TransferBufferLength:   0x0
TransferBuffer:         0xfffffa8005a2cc88
TransferBufferMDL:      0xfffffa8005848930
```

# См. также

Расширения USB 3.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_capability

Статья • 05.04.2024

Расширение **!usb3kd.xhci_capability** отображает возможности контроллера узла USB 3.0.

dbgcmd

```
!usb3kd.xhci_capability DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства контроллера узла (FDO).

## DLL-библиотеки

Usb3kd.dll

## Замечания

Выходные данные **команды !xhci_capability** основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в статье "Архитектура стека драйверов USB".

## Примеры

Чтобы получить адрес расширения устройства, просмотрите выходные данные **команды !xhci_dumpall** . В следующем примере адрес расширения устройства 0xfffffa800536e2d0.

dbgcmd

```
3: kd> !xhci_dumpall

## Dumping all the XHCI controllers - DrvObj 0xfffffa80053072f0
------------------------------------------------------------
1)  ... - PCI: VendorId ... DeviceId ... RevisionId ... Firmware ...
```

```
    dt USBXHCI!_CONTROLLER_DATA 0xfffffa80052f20c0
    !rcdrlogdump USBXHCI -a 0xfffffa8005068520
    !rcdrlogdump USBXHCI -a 0xfffffa8004e8b9a0 (rundown)
    !wdfdevice 0x57ffac91fd8
    !xhci_capability 0xfffffa800536e2d0
    ...
```

Теперь вы можете передать адрес расширения устройства команде **!xhci_capability** .

dbgcmd

```
3: kd> !xhci_capability 0xfffffa800536e2d0

## Controller Capabilities
-----------------------
    dt USBXHCI!_REGISTER_DATA 0xfffffa8005362c00
    dt USBXHCI!_CAPABILITY_REGISTERS 0xfffff880046a8000
    MajorRevision.MinorRevision = 0.96
    Device Slots: 32
    Interrupters: 8
    Ports: 4
    IsochSchedulingThreshold: 1
    EventRingSegmentTableMax: 1 (2^ERST = 2)
    ScratchpadRestore: OFF
    MaxScratchpadBuffers: 0
    U1DeviceExitLatency: 0
    U2DeviceExitLatency: 0
    AddressingCapability: 64 bit
    BwNegotiationCapability: ON
    ContextSize: 32 bytes
    PortPowerControl: ON
    PortIndicators: OFF
    LightHCResetCapability: OFF
    LatencyToleranceMessagingCapability: ON
    NoSecondarySidSupport: TRUE
    MaximumPrimaryStreamArraySize = 4 ( 2^(MaxPSASize+1) = 32 )
    XhciExtendedCapabilities:
        [1] USB_LEGACY_SUPPORT: dt _USBLEGSUP 0xfffff880046a8500
        [2] Supported Protocol 0xfffff880046a8510, Version 3.0, Offset 1,
Count 2, HighSpeedOnly OFF, IntegratedHub OFF, HardwareLPM OFF
        [3] Supported Protocol 0xfffff880046a8520, Version 2.0, Offset 3,
Count 2, HighSpeedOnly OFF, IntegratedHub OFF, HardwareLPM OFF

## Software Supported Capabilities
------------------------------
    DeviceSlots: 32
    Interrupters: 1
    Ports: 4
    MaxEventRingSegments: 2
    U1DeviceExitLatency: 0
    U2DeviceExitLatency: 0
```

```
    DeviceFlags:
        IgnoreBiosHandoffFailure
        SetLinkTrbChainBit
        UseSingleInterrupter
        DisableIdlePowerManagement
```

## См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_commandring

Статья • 05.04.2024

Расширение **!usb3kd.xhci_commandring** отображает сведения о структуре данных командного круга, связанной с контроллером узла USB 3.0.

```
dbgcmd

!usb3kd.xhci_commandring DeviceExtension
```

## Параметры

*DeviceExtension*
AAddress расширения устройства для функционального объекта устройства контроллера узла (FDO).

## DLL-библиотеки

Usb3kd.dll

## Замечания

Выходные данные **команды !xhci_commandring** основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в статье "Архитектура стека драйверов USB".

Кольцо команд — это структура данных, используемая драйвером контроллера узла USB 3.0 для передачи команд контроллеру узла.

## Примеры

Чтобы получить адрес расширения устройства, просмотрите выходные данные **команды !xhci_dumpall** . В следующем примере адрес расширения устройства 0xfffffa800536e2d0.

```
dbgcmd

3: kd> !xhci_dumpall
```

```
## Dumping all the XHCI controllers - DrvObj 0xfffffa80053072f0
-------------------------------------------------------------
1)  ... - PCI: VendorId ... DeviceId ... RevisionId ... Firmware ...

    dt USBXHCI!_CONTROLLER_DATA 0xfffffa80052f20c0
    !rcdrlogdump USBXHCI -a 0xfffffa8005068520
    !rcdrlogdump USBXHCI -a 0xfffffa8004e8b9a0 (rundown)
    !wdfdevice 0x57ffac91fd8
    !xhci_capability 0xfffffa800536e2d0
    !xhci_registers 0xfffffa800536e2d0
    !xhci_commandring 0xfffffa800536e2d0 (No commands are pending)
    ...
```

Теперь вы можете передать адрес расширения устройства команде **!xhci_commandring** .

dbgcmd

```
3: kd> !xhci_commandring 0xfffffa800536e2d0

## Dumping dt _COMMAND_DATA 0xfffffa8005362f70 !rcdrlogdump USBXHCI -a
0xfffffa8005a8f010
-------------------------------------------------------------------------------
---------
Stop: OFF Abort: OFF Running: ON
CommandRingBufferData: VA 0xfffffa8005aeb200 LA 0x1168eb200 !wdfcommonbuffer
0x57ffa65d988 Size 512
DequeueIndex: 24 EnqueueIndex: 24 CycleState: 0

    Command Ring TRBs:
        [  0] Unknown TRB Type 49 0xfffffa8005aeb200

        [  1] ENABLE_SLOT                 0xfffffa8005aeb210 CycleBit 1
        [  2] ADDRESS_DEVICE              0xfffffa8005aeb220 CycleBit 1
SlotId  1 BlockSetAddressRequest 1
        ...

    PendingList:
        Empty List

    WaitingList:
        Empty List
```

# См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_deviceslots

Статья • 08.07.2024

Расширение **!usb3kd.xhci_deviceslots** отображает сведения о устройствах, подключенных к контроллеру узла USB 3.0.

```
dbgcmd
```

```
!usb3kd.xhci_deviceslots DeviceExtension [SlotNumber] [verbose]
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства контроллера узла (FDO).

*SlotNumber*
Номер слота отображаемого устройства. Если этот параметр опущен, отображаются все устройства.

**многословный**
Отображение подробно.

## DLL-библиотеки

Usb3kd.dll

## Замечания

Выходные данные **команды !xhci_deviceslots** основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в статье "Архитектура стека драйверов USB".

Драйвер контроллера узла USB 3.0 поддерживает список структур данных, представляющих устройства, подключенные к контроллеру. Каждая из этих структур данных определяется номером слота.

## Примеры

Чтобы получить адрес расширения устройства, просмотрите выходные данные **команды !xhci_dumpall** . В следующем примере адрес расширения устройства 0xfffffa800536e2d0.

dbgcmd

```
3: kd> !xhci_dumpall

## Dumping all the XHCI controllers - DrvObj 0xfffffa80053072f0
-------------------------------------------------------------
1)  ... - PCI: VendorId ... DeviceId ... RevisionId ... Firmware ...

    dt USBXHCI!_CONTROLLER_DATA 0xfffffa80052f20c0
    !rcdrlogdump USBXHCI -a 0xfffffa8005068520
    !rcdrlogdump USBXHCI -a 0xfffffa8004e8b9a0 (rundown)
    !wdfdevice 0x57ffac91fd8
    !xhci_capability 0xfffffa800536e2d0
    !xhci_registers 0xfffffa800536e2d0
    !xhci_commandring 0xfffffa800536e2d0 (No commands are pending)
    !xhci_deviceslots 0xfffffa800536e2d0
    ...
```

Теперь вы можете передать адрес расширения устройства команде **!usb3kd.xhci_deviceslots** .

dbgcmd

```
3: kd> !xhci_deviceslots 0xfffffa800536e2d0

## Dumping dt _DEVICESLOT_DATA 0xfffffa8005226220
-----------------------------------------------
DeviceContextBase: VA 0xfffffa8005ab9000 LA 0x1168b9000 !wdfcommonbuffer
0x57ffa65c9b8 Size 4096

## [1] SlotID : dt USBXHCI!_USBDEVICE_DATA 0xfffffa8005a427d0 dt
_SLOT_CONTEXT32 0xfffffa8005aba000
-------------------------------------------------------------------------------
--------------------
    USB\VID_125F&PID_312A ADATA Technology Co., Ltd.
    SlotEnabled IsDevice NumberOfTTs 0 TTThinkTime 0
    Speed: Super PortPathDepth: 1 PortPath: [ 2 ] DeviceAddress: 1
    !device_info_from_pdo 0xfffffa8005a36800
    DeviceContextBuffer: VA 0xfffffa8005aba000 LA 0x1168ba000
!wdfcommonbuffer 0x57ffa656948 Size 4096
    InputDeviceContextBuffer: VA 0xfffffa8005b65000 LA 0x116965000
!wdfcommonbuffer 0x57ffa5be958 Size 4096

    [1] DeviceContextIndex : dt USBXHCI!_ENDPOINT_DATA 0xfffffa8005a126f0 dt
_ENDPOINT_CONTEXT32 0xfffffa8005aba020 ES_RUNNING
    -------------------------------------------------------------------------
-------------------------------------
```

```
        EndpointType_Control Address: 0x0 PacketSize: 512 Interval: 0
        !ucx_endpoint 0xfffffa8005a3f710
        RecorderLog: !rcdrlogdump USBXHCI -a 0xfffffa8005b60010

        [0] dt _TRANSFERRING_DATA 0xfffffa8005b64ec0 Events: 0x0
TransferRingState_Idle
     ...

## [2] SlotID : dt USBXHCI!_USBDEVICE_DATA 0xfffffa80052de320 dt
_SLOT_CONTEXT32 0xfffffa8005b8b000
--------------------------------------------------------------------------
--------------------
    USB\VID_18A5&PID_0304 Verbatim Americas LLC
    SlotEnabled IsDevice NumberOfTTs 0 TTThinkTime 0
    Speed: High PortPathDepth: 1 PortPath: [ 3 ] DeviceAddress: 2
    !device_info_from_pdo 0xfffffa80058fb800
    DeviceContextBuffer: VA 0xfffffa8005b8b000 LA 0x11698b000
!wdfcommonbuffer 0x57ffa426b18 Size 4096
    InputDeviceContextBuffer: VA 0xfffffa8005b8c000 LA 0x11698c000
!wdfcommonbuffer 0x57ffadbe3c8 Size 4096

    [1] DeviceContextIndex : dt USBXHCI!_ENDPOINT_DATA 0xfffffa800714b050 dt
_ENDPOINT_CONTEXT32 0xfffffa8005b8b020 ES_RUNNING
    ----------------------------------------------------------------------
-------------------------------------
        EndpointType_Control Address: 0x0 PacketSize: 64 Interval: 0
        !ucx_endpoint 0xfffffa80036a20c0
        RecorderLog: !rcdrlogdump USBXHCI -a 0xfffffa8005bd0b60

        [0] dt _TRANSFERRING_DATA 0xfffffa8004ed8df0 Events: 0x0
TransferRingState_Idle
        ----------------------------------------------------------------------
----------
    ...
```

# См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_dumpall

Статья • 05.04.2024

Команда **!usb3kd.xhci_dumpall** отображает сведения обо всех контроллерах узла USB 3.0 на компьютере. Отображение основано на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys).

dbgcmd

```
!usb3kd.xhci_dumpall [1]
```

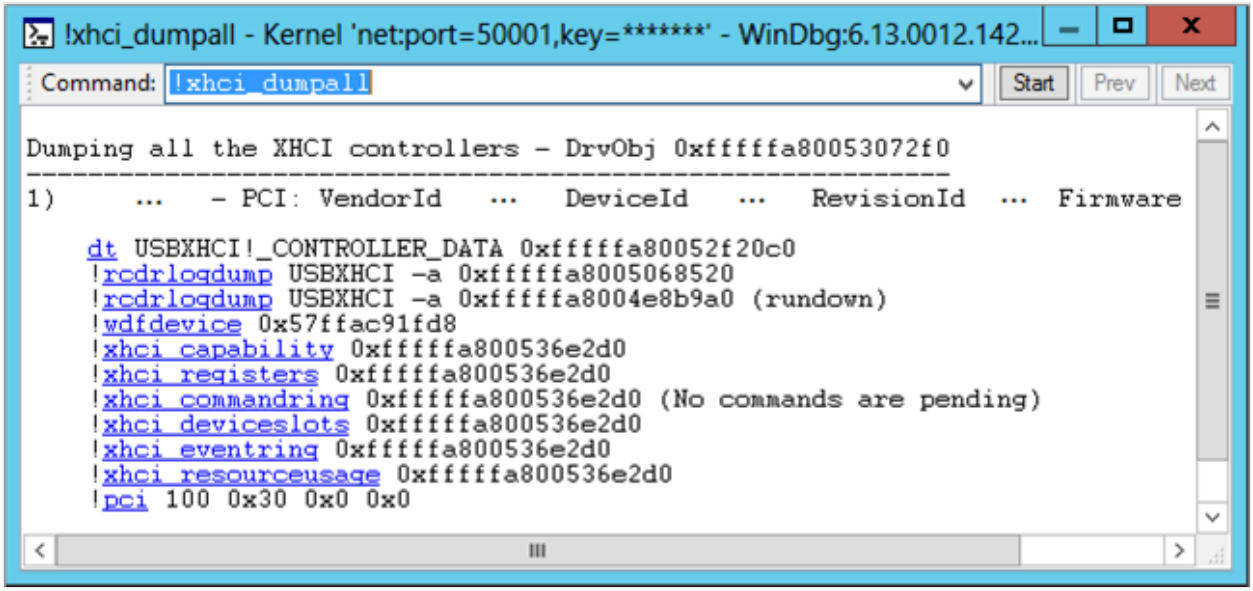
## Параметры

1

Выполняет все команды XHCI и отображает выходные данные каждой команды.

## Примеры

На следующем снимке экрана показаны выходные данные **команды !xhci_dumpall**l.




В выходных данных показано, что есть один контроллер узла USB 3.0.

Выходные данные используют язык разметки отладчика (DML) для предоставления ссылок. Ссылки выполняют команды, которые предоставляют подробные сведения о состоянии контроллера узла, как он поддерживается драйвером контроллера узла USB 3.0. Например, можно получить подробные сведения о возможностях контроллера узла, щелкнув ссылку **!xhci_capability** . В качестве альтернативы

щелчку ссылки можно ввести команду. Например, чтобы просмотреть сведения об использовании ресурсов контроллера узла, можно ввести команду **!xhci_resourceusage 0xfffffa800536e2d0**.

**Обратите внимание** , что функция DML доступна в WinDbg, но не в Visual Studio или KD.

## DLL-библиотеки

Usb3kd.dll

## Замечания

**Команда !xhci_dumpall** является родительской командой для этого набора команд.

- **!xhci_capability**
- **!xhci_info**
- **!xhci_deviceslots**
- **!xhci_commandring**
- **!xhci_eventring**
- **!xhci_transferring**
- **!xhci_trb**
- **!xhci_registers**
- **!xhci_resourceusage**

Сведения, отображаемые семейством **команд !xhci_dumpall** , основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0. Сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB 3.0 см. в статье "Архитектура стека драйверов USB". Описание структур данных, используемых драйверами в стеке USB 3.0, см. в разделе "Часть 2 инноваций в отладке USB" в видео с Windows 8 .

## См. также

Расширения USB 3.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_eventring

Статья • 05.04.2024

Расширение **!usb3kd.xhci_eventring** отображает сведения о структуре данных кольца событий, связанной с контроллером узла USB 3.0.

```
dbgcmd
```

```
!usb3kd.xhci_eventring DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства контроллера узла (FDO).

## DLL-библиотеки

Usb3kd.dll

## Замечания

Команда выходных данных **!xhci_eventring** основана на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в драйверах на стороне USB-узла в Windows.

Кольцо событий — это структура, используемая контроллером узла USB 3.0 для информирования драйверов о завершении действия.

## Примеры

Чтобы получить адрес расширения устройства, просмотрите выходные данные **команды !xhci_dumpall** . В следующем примере адрес расширения устройства 0xfffffa800536e2d0.

```
dbgcmd
```

```
3: kd> !xhci_dumpall
```

```
## Dumping all the XHCI controllers - DrvObj 0fffffa80053072f0
-------------------------------------------------------------
1)  ... - PCI: VendorId ... DeviceId ... RevisionId ... Firmware ...

    dt USBXHCI!_CONTROLLER_DATA 0fffffa80052f20c0
    !rcdrlogdump USBXHCI -a 0fffffa8005068520
    !rcdrlogdump USBXHCI -a 0fffffa8004e8b9a0 (rundown)
    !wdfdevice 0x57ffac91fd8
    !xhci_capability 0fffffa800536e2d0
    !xhci_registers 0fffffa800536e2d0
    !xhci_commandring 0fffffa800536e2d0 (No commands are pending)
    !xhci_deviceslots 0fffffa800536e2d0
    !xhci_eventring 0fffffa800536e2d0
    ...
```

Теперь вы можете передать адрес расширения устройства команде **!xhci_eventring** .

dbgcmd

```
3: kd> !xhci_eventring 0fffffa800536e2d0

## Dumping dt _PRIMARY_INTERRUPTER_DATA ffffa800536b5b0
-------------------------------------------------------

## [0] Interrupter : dt _INTERRUPTER_DATA 0fffffa800536b7d0  !rcdrlogdump
USBXHCI -a 0fffffa8005aeab60
-------------------------------------------------------------------------------
-------------------------
    DequeueSegment: 1 DequeueIndex: 217 TotalEventRingSegments: 2
TRBsPerSegment: 256
    CurrentBufferData : VA 0fffffa8005373000 LA 0x117173000
!wdfcommonbuffer 0x57ffa65b9b8 Size 4096
    EventRingTableBufferData : VA 0fffffa8005aeb000 LA 0x1168eb000
!wdfcommonbuffer 0x57ffa65d988 Size 512

    [0] VA 0fffffa8005370000 LA 0x117170000 !wdfcommonbuffer 0x57ffa6599b8
Size 4096
    [1] VA 0fffffa8005373000 LA 0x117173000 !wdfcommonbuffer 0x57ffa65b9b8
Size 4096

    Event Ring TRBs:
        [207] TRANSFER_EVENT      0fffffa8005373cf0 CycleBit 0 SlotId  2
EndpointID  4 EventData 1 Pointer 0fffffa8005366700 CC_SUCCESS
        [208] TRANSFER_EVENT      0fffffa8005373d00 CycleBit 0 SlotId  2
EndpointID  3 EventData 1 Pointer 0fffffa8005a3d850 CC_SHORT_PACKET
        [209] TRANSFER_EVENT      0fffffa8005373d10 CycleBit 0 SlotId  1
EndpointID  4 EventData 1 Pointer 0fffffa8005a3d850 CC_SUCCESS
        [210] TRANSFER_EVENT      0fffffa8005373d20 CycleBit 0 SlotId  1
EndpointID  3 EventData 1 Pointer 0fffffa8005366700 CC_SUCCESS
        [211] TRANSFER_EVENT      0fffffa8005373d30 CycleBit 0 SlotId  2
EndpointID  4 EventData 1 Pointer 0fffffa8005366700 CC_SUCCESS
        [212] TRANSFER_EVENT      0fffffa8005373d40 CycleBit 0 SlotId  2
EndpointID  3 EventData 1 Pointer 0fffffa8005a3d850 CC_SHORT_PACKET
```

```
        [213] TRANSFER_EVENT      0xfffffa8005373d50 CycleBit 0 SlotId  1
EndpointID  4 EventData 1 Pointer 0xfffffa8005a3d850 CC_SUCCESS
        [214] TRANSFER_EVENT      0xfffffa8005373d60 CycleBit 0 SlotId  1
EndpointID  3 EventData 1 Pointer 0xfffffa8005366700 CC_SUCCESS
        [215] TRANSFER_EVENT      0xfffffa8005373d70 CycleBit 0 SlotId  2
EndpointID  4 EventData 1 Pointer 0xfffffa8005366700 CC_SUCCESS
        [216] TRANSFER_EVENT      0xfffffa8005373d80 CycleBit 0 SlotId  2
EndpointID  3 EventData 1 Pointer 0xfffffa8005a3d850 CC_SHORT_PACKET
```

# См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_findowner

Статья • 05.04.2024

Команда **!usb3kd.xhci_findowner** находит владельца общий буфер.

```
dbgcmd
```

```
!usb3kd.xhci_findowner Address
```

## Параметры

*Адрес*
Виртуальный или физический адрес общего буфера.

## DLL-библиотеки

Usb3kd.dll

## Замечания

Общий буфер — это блок физической непрерывной памяти, которая может решаться оборудованием. Стек драйверов USB 3.0 использует общие буферы для взаимодействия с контроллерами узла USB 3.0. Предположим, что системный сбой, и вы столкнулись с адресом, который вы подозреваете, может быть общей буферной памятью. Если адрес является общей буферной памятью, эта команда сообщает, какой контроллер узла USB 3.0 принадлежит (если у вас есть более одного контроллера USB 3.0) и то, что используется для памяти.

## Примеры

В следующем примере вызывается **!xhci_resourceusage** для перечисления адресов некоторых распространенных буферов.

```
dbgcmd
```

```
0: kd> !usb3kd.xhci_resourceusage 0x867fbff0

## Dumping CommonBuffer Resources
------------------------------
    dt USBXHCI!_COMMON_BUFFER_DATA 0x868d61c0
```

```
    DmaEnabler:!wdfdmaenabler 0x79729fe8

    CommonBuffers Large: Total 8 Available 2 Used 6 TotalBytes 32768
        [ 1] dt _TRACKING_DATA 0x868d73a4 VA 0x868e0000 LA 0xdb2e0000 --
Owner 0x86801690 Tag: Int2 Size 4096
        [ 2] dt _TRACKING_DATA 0x868d6d1c VA 0x868e1000 LA 0xdb2e1000 --
Owner 0x86801690 Tag: Int2 Size 4096
        [ 3] dt _TRACKING_DATA 0x868d6c54 VA 0x868e2000 LA 0xdb2e2000 --
Owner 0x86801690 Tag: Int2 Size 4096
        [ 4] dt _TRACKING_DATA 0x868d6b8c VA 0x868e3000 LA 0xdb2e3000 --
Owner 0x86801690 Tag: Int2 Size 4096
        [ 5] dt _TRACKING_DATA 0x868d67b4 VA 0x868e5000 LA 0xdb2e5000 --
Owner 0x86801548 Tag: Slt1 Size 4096
        [ 6] dt _TRACKING_DATA 0x868d50b4 VA 0x868e6000 LA 0xdb2e6000 --
Owner 0x86801548 Tag: Slt3 Size 4096

    CommonBuffers Small: Total 16 Available 13 Used 3 TotalBytes 8192
        [ 1] dt _TRACKING_DATA 0x868d6974 VA 0x868e4000 LA 0xdb2e4000 --
Owner 0x86801690 Tag: Int1 Size 512
        [ 2] dt _TRACKING_DATA 0x868d69a4 VA 0x868e4200 LA 0xdb2e4200 --
Owner 0x86801548 Tag: Slt2 Size 512
        [ 3] dt _TRACKING_DATA 0x868d69d4 VA 0x868e4400 LA 0xdb2e4400 --
Owner 0x86801488 Tag: Cmd1 Size 512
```

Одним из виртуальных адресов, перечисленных в предыдущем выходных данных, является 0x868e2000. В следующем примере этот адрес передается **в !xhci_findowner**. Одним из физических адресов, перечисленных в предыдущем выходных данных, является 0xdb2e4400. В следующем примере 0xdb2e4440 (смещение 0x40 байтов от 0xdb2e4400) передается **в !xhci_findowner**.

dbgcmd

```
0: kd> !xhci_findowner 0x868e2000

!xhci_info 0x867fbff0  Texas Instruments - PCI: VendorId 0x104c DeviceId
0x8241 RevisionId 0x02

    dt _TRACKING_DATA 0x868d6c54 VA 0x868e2000 LA 0xdb2e2000 -- Owner
0x86801690 Tag: Int2 Size 4096

    dt _INTERRUPTER_DATA 0x86801690
    !xhci_eventring 0x867fbff0  <-- This memory is used for event ring.


0: kd> !xhci_findowner 0xdb2e4440  <-- Note the offset difference.

!xhci_info 0x867fbff0  Texas Instruments - PCI: VendorId 0x104c DeviceId
0x8241 RevisionId 0x02

    dt _TRACKING_DATA 0x868d69d4 VA 0x868e4400 LA 0xdb2e4400 -- Owner
0x86801488 Tag: Cmd1 Size 512
```

```
    dt _COMMAND_DATA 0x86801488
    !xhci_commandring 0x867fbff0  <-- This memory is used for command ring.
```

Команда **!xhci_findowner** особенно полезна, если у вас есть адрес в блоке запроса на передачу (TRB), и вы хотите отслеживать его обратно в слот устройства, к которому он принадлежит. В следующем примере один из адресов, перечисленных в выходных данных **!xhci_transferring** , является 0xda452230, который является физическим адресом TRB. Пример передает этот адрес **в !xhci_findowner**. Выходные данные показывают, что TRB принадлежит слоту устройства 8 (**!xhci_deviceslots 0x8551d370 8**).

dbgcmd

```
0: kd> !usb3kd.xhci_transferring 0x87652200

        [  0] NORMAL       0xda452200 CycleBit 1 IOC 0 BEI 0
InterrupterTarget 2 TransferLength     6 TDSize  0
        [  1] EVENT_DATA   0xda452210 CycleBit 1 IOC 1 BEI 0
InterrupterTarget 2 Data 0x8511375c TotalBytes 6
        [  2] NORMAL       0xda452220 CycleBit 1 IOC 0 BEI 0
InterrupterTarget 2 TransferLength     6 TDSize  0
        [  3] EVENT_DATA   0xda452230 CycleBit 1 IOC 1 BEI 0
InterrupterTarget 2 Data 0x857d076c TotalBytes 6

0: kd> !xhci_findowner 0xda452230

!xhci_info 0x8551d370  Renesas - PCI: VendorId 0x1912 DeviceId 0x0015
RevisionId 0x02 Firmware 0x0020.0006

    dt _TRACKING_DATA 0x8585fd5c VA 0x87652200 LA 0xda452200 -- Owner
0x85894548 Tag: Rng1 Size 512

    !xhci_deviceslots 0x8551d370 8

    [0] dt _TRANSFERRING_DATA 0x85894548 Events: 0x0 TransferRingState_Idle
    ------------------------------------------------------------------------
------
        WdfQueue: !wdfqueue 0x7a76bcb0 (0 waiting)
        CurrentRingBufferData: VA 0x87652200 LA 0xda452200 !wdfcommonbuffer
0x7a7a0370 Size 512
        Current:  !xhci_transferring 0x87652200
        PendingTransferList:
            [0] dt _TRANSFER_DATA 0x851136f0 !urb 0x84e55468 !wdfrequest
0x7aeec9e8 TransferState_Pending
            [1] dt _TRANSFER_DATA 0x857d0700 !urb 0x85733be8 !wdfrequest
0x7a82f9d8 TransferState_Pending
```

# См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_info

Статья • 05.04.2024

Расширение **!usb3kd.xhci_info** отображает все команды XHCI для отдельного контроллера узла USB 3.0.

dbgcmd

```
!usb3kd.xhci_info DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства контроллера узла (FDO).

## DLL-библиотеки

Usb3kd.dll

## Замечания

Выходные данные **команды !usb3kd.xhci_info** основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в драйверах на стороне USB-узла в Windows.

## Примеры

Вы можете получить адрес расширения устройства из **команды !xhci_dumpall** или из различных других команд отладчика. Например, **команда !devstack** отображает адрес расширения устройства. В следующем примере адрес расширения устройства для FDO контроллера узла — fffffa800536e2d0.

dbgcmd

```
3: kd> !devnode 0 1 usbxhci
Dumping IopRootDeviceNode (= 0xfffffa8003609cc0)
DevNode 0xfffffa8003df3010 for PDO 0xfffffa8003dd5870
  InstancePath is "PCI\VEN_...
```

```
  ServiceName is "USBXHCI"
  ...

3: kd> !devstack 0xfffffa8003dd5870
  !DevObj   !DrvObj            !DevExt   ObjectName
  fffffa800534b060  \Driver\USBXHCI    fffffa800536e2d0  USBFDO-3
  fffffa8003db5790  \Driver\ACPI       fffffa8003701cb0
> fffffa8003dd5870  \Driver\pci        fffffa8003dd59c0  NTPNP_PCI0020
...
```

Теперь вы можете передать адрес расширения устройства команде **!xhci_info** .

dbgcmd

```
3: kd> !xhci_info 0fffffa80`0536e2d0

## Dumping XHCI controller commands - DeviceExtension 0fffffa800536e2d0
-------------------------------------------------------------
 ... - PCI: VendorId ... DeviceId ... RevisionId ... Firmware ...

    dt USBXHCI!_CONTROLLER_DATA 0fffffa80052f20c0
    !rcdrlogdump USBXHCI -a 0fffffa8005068520
    !rcdrlogdump USBXHCI -a 0fffffa8004e8b9a0 (rundown)
    !wdfdevice 0x57ffac91fd8
    !xhci_capability 0fffffa800536e2d0
    !xhci_registers 0fffffa800536e2d0
    !xhci_commandring 0fffffa800536e2d0 (No commands are pending)
    !xhci_deviceslots 0fffffa800536e2d0
    !xhci_eventring 0fffffa800536e2d0
    !xhci_resourceusage 0fffffa800536e2d0
    !pci 100 0x30 0x0 0x0
```

# См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_registers

Статья • 05.04.2024

Расширение **!usb3kd.xhci_registers** отображает регистры контроллера узла USB 3.0.

```
dbgcmd

!usb3kd.xhci_registers DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства контроллера узла (FDO).

## DLL-библиотеки

Usb3kd.dll

## Замечания

Выходные данные **команды !xhci_registers** основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в драйверах на стороне USB-узла в Windows.

## Примеры

Чтобы получить адрес расширения устройства, просмотрите выходные данные **команды !xhci_dumpall** . В следующем примере адрес расширения устройства 0xfffffa800536e2d0.

```
dbgcmd

3: kd> !xhci_dumpall

## Dumping all the XHCI controllers - DrvObj 0xfffffa80053072f0
---------------------------------------------------------------
1)  ... - PCI: VendorId ... DeviceId ... RevisionId ... Firmware ...
```

```
    dt USBXHCI!_CONTROLLER_DATA 0xfffffa80052f20c0
    !rcdrlogdump USBXHCI -a 0xfffffa8005068520
    !rcdrlogdump USBXHCI -a 0xfffffa8004e8b9a0 (rundown)
    !wdfdevice 0x57ffac91fd8
    !xhci_capability 0xfffffa800536e2d0
    !xhci_registers 0xfffffa800536e2d0
    ...
```

Теперь вы можете передать адрес расширения устройства команде **!xhci_registers** .

dbgcmd

```
3: kd> !xhci_registers 0xfffffa800536e2d0

## Dumping controller registers
----------------------------

    dt USBXHCI!_OPERATIONAL_REGISTERS 0xfffff880046a8020

    DeviceContextBaseAddressArrayPointer: 00000001168b9000

    Command Registers
    -----------------
        RunStopBit: 1
        HostControllerReset: 0
        ...

    Status Registers
    ----------------
        HcHalted: 0
        HostSystemError: 0
        ...

    commandRingControl Registers
    ----------------------------
        RingCycleState: 0
        CommandStop: 0
        ...
    Runtime Registers
    -----------------
        dt USBXHCI!_RUNTIME_REGISTERS 0xfffff880046a8600
        MicroFrameIndex: 0x3f7a

        dt -ba8 USBXHCI!_INTERRUPTER_REGISTER_SET 0xfffff880046a8620

    RootPort Registers
    ------------------
        dt -a4 -r2 USBXHCI!_PORT_REGISTER_SET 0xfffff880046a8420
```

# См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_resourceusage

Статья • 05.04.2024

Расширение **!usb3kd.xhci_resourceusage** отображает ресурсы, используемые контроллером узла USB 3.0.

dbgcmd

```
!usb3kd.xhci_resourceusage DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства контроллера узла (FDO).

## DLL-библиотеки

Usb3kd.dll

## Замечания

Выходные данные **команды !xhci_resourceusage** основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в драйверах на стороне USB-узла в Windows.

## Примеры

Чтобы получить адрес расширения устройства, просмотрите выходные данные **команды !xhci_dumpall** . В следующем примере адрес расширения устройства 0xfffffa800536e2d0.

dbgcmd

```
3: kd> !xhci_dumpall

## Dumping all the XHCI controllers - DrvObj 0xfffffa80053072f0
------------------------------------------------------------
1)  ... - PCI: VendorId ... DeviceId ... RevisionId ... Firmware ...
```

```
    dt USBXHCI!_CONTROLLER_DATA 0xfffffa80052f20c0
    !rcdrlogdump USBXHCI -a 0xfffffa8005068520
    !rcdrlogdump USBXHCI -a 0xfffffa8004e8b9a0 (rundown)
    !wdfdevice 0x57ffac91fd8
    !xhci_capability 0xfffffa800536e2d0
    !xhci_registers 0xfffffa800536e2d0
    !xhci_commandring 0xfffffa800536e2d0 (No commands are pending)
    !xhci_deviceslots 0xfffffa800536e2d0
    !xhci_eventring 0xfffffa800536e2d0
    !xhci_resourceusage 0xfffffa800536e2d0
    ...
```

Теперь вы можете передать адрес расширения устройства команде **!xhci_resourceusage** .

dbgcmd

```
 3: kd> !xhci_resourceusage 0xfffffa800536e2d0

## Dumping CommonBuffer Resources
------------------------------
    dt USBXHCI!_COMMON_BUFFER_DATA 0xfffffa80059a5920
    DmaEnabler:!wdfdmaenabler 0x57ffa65a9c8

    CommonBuffers Large: Total 9 Available 2 Used 7 TotalBytes 36864
        [ 1] dt _TRACKING_DATA 0xfffffa80059a6768 VA 0xfffffa8005370000 LA
0x117170000 ...
        [ 2] dt _TRACKING_DATA 0xfffffa80059a4768 VA 0xfffffa8005373000 LA
0x117173000 ...
        ...
    CommonBuffers Small: Total 32 Available 8 Used 24 TotalBytes 16384
        [ 1] dt _TRACKING_DATA 0xfffffa80059a2798 VA 0xfffffa8005aeb000 LA
0x1168eb000 ...
        [ 2] dt _TRACKING_DATA 0xfffffa80059a27e8 VA 0xfffffa8005aeb200 LA
0x1168eb200 ...
        ...
```

# См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_trb

Статья • 05.04.2024

Расширение **!usb3kd.xhci_trb** отображает один или несколько блоков запросов передачи (TRBS), используемых контроллером узла USB 3.0.

```
dbgcmd
```

```
!usb3kd.xhci_trb VirtualAddress Count
!usb3kd.xhci_trb PhysicalAddress Count 1
```

## Параметры

*VirtualAddress*
Виртуальный адрес TRB.

*PhysicalAddress*
Физический адрес TRB.

*Численность*
Число последовательных TRB для отображения, начиная с *VirtualAddress* или *PhysicalAddress*.

1
Указывает, что адрес является физическим адресом.

## DLL-библиотеки

Usb3kd.dll

## Замечания

Выходные данные **команды !xhci_trb** основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в драйверах на стороне USB-узла в Windows.

## Примеры

В следующем примере **0x844d7c00** — это виртуальный адрес TRB. **1** — это число, указывающее, сколько последовательных TROB для отображения.

dbgcmd

```
0: kd> !xhci_trb 0x844d7c00 1

        [  0] ISOCH        0x844d7c00 CycleBit 1 IOC 0 CH 1 BEI 0
InterrupterTarget 1 TransferLength  2688 TDSize  0 TBC 0 TLBPC 2 Frame 0x3D2
```

В следующем примере **0x0dced7c00** — это физический адрес TRB. **4** — это число, указывающее количество последовательных TRB для отображения. **1** указывает, что адрес является физическим адресом.

dbgcmd

```
0: kd> !xhci_trb 0x0dced7c00 4 1

        [  0] ISOCH        0xdced7c00 CycleBit 1 IOC 0 CH 1 BEI 0
InterrupterTarget 1 TransferLength  2688 TDSize  0 TBC 0 TLBPC 2 Frame 0x3D2
        [  1] EVENT_DATA   0xdced7c10 CycleBit 1 IOC 1 CH 0 BEI 1
InterrupterTarget 1 Data 0x194c9bcf001b0001 PacketId 27 Frame 0x194c9bcf
TotalBytes 2688
        [  2] ISOCH        0xdced7c20 CycleBit 1 IOC 0 CH 1 BEI 0
InterrupterTarget 1 TransferLength  1352 TDSize  2 TBC 0 TLBPC 2 Frame 0x3D2
        [  3] NORMAL       0xdced7c30 CycleBit 1 IOC 0 CH 1 BEI 0
InterrupterTarget 1 TransferLength  1336 TDSize  0
```

## См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usb3kd.xhci_transferring

Статья • 05.04.2024

Расширение **!usb3kd.xhci_transferring** отображает кольцо передачи (используется контроллером узла USB 3.0), пока он не обнаружит бит циклического изменения.

```
dbgcmd
```

```
!usb3kd.xhci_transferring VirtualAddress
!usb3kd.xhci_transferring PhysicalAddress 1
```

## Параметры

*VirtualAddress*
Виртуальный адрес кольца передачи.

*PhysicalAddress*
Физический адрес кольца передачи.

1
Указывает, что адрес является физическим адресом.

## DLL-библиотеки

Usb3kd.dll

## Замечания

Выходные данные **команды !xhci_transferring** основаны на структурах данных, поддерживаемых драйвером контроллера узла USB 3.0 (UsbXhci.sys). Дополнительные сведения о драйвере контроллера узла USB 3.0 и других драйверах в стеке USB см. в драйверах на стороне USB-узла в Windows.

Кольцо передачи — это структура, используемая драйвером контроллера узла USB 3.0 для поддержания списка блоков запросов на передачу (TRBs). Эта команда принимает виртуальный или физический адрес кольца передачи, но отображает физический адрес TRB. Это делается так, чтобы команда может правильно пройти по TRB LINK.

# Примеры

Чтобы получить адрес кольца передачи, просмотрите выходные данные **команды !xhci_deviceslots** . В следующем примере виртуальный адрес кольца передачи 0xfffffa8005b2fe00.

dbgcmd

```
3: kd> !usb3kd.xhci_deviceslots 0xfffffa800523a2d0

## Dumping dt _DEVICESLOT_DATA 0xfffffa80051a3300
-----------------------------------------
DeviceContextBase: VA 0xfffffa8005a41000 LA 0x116841000 !wdfcommonbuffer
0x57ffa6ff9b8 Size 4096

## [1] SlotID : dt USBXHCI!_USBDEVICE_DATA 0xfffffa800598c7d0 dt
_SLOT_CONTEXT32 0xfffffa8005a42000
-------------------------------------------------------------------------
--------------------
    USB\VID_125F&PID_312A ADATA Technology Co., Ltd.
    SlotEnabled IsDevice NumberOfTTs 0 TTThinkTime 0
    Speed: Super PortPathDepth: 1 PortPath: [ 2 ] DeviceAddress: 1
    !device_info_from_pdo 0xfffffa800597d720
    DeviceContextBuffer: VA 0xfffffa8005a42000 LA 0x116842000
!wdfcommonbuffer 0x57ffa7009b8 Size 4096
    InputDeviceContextBuffer: VA 0xfffffa8005b2d000 LA 0x11692d000
!wdfcommonbuffer 0x57ffa674958 Size 4096
    ...

    [3] DeviceContextIndex : dt USBXHCI!_ENDPOINT_DATA 0xfffffa8005b394e0 dt
_ENDPOINT_CONTEXT32 0xfffffa8005a42060 ES_RUNNING
    -------------------------------------------------------------------------
-----------------------------------
       ...
            CurrentRingBufferData: VA 0xfffffa8005b2fe00 LA 0x11692fe00
!wdfcommonbuffer 0x57ffa67c988 Size 512
            Current:  !xhci_transferring 0xfffffa8005b2fe00
            PendingTransferList:
                [0] dt _TRANSFER_DATA 0xfffffa8005b961b0 !urb
0xfffffa8005b52be8 !wdfrequest 0x57ffa469fd8 TransferState_Pending
```

Теперь вы можете передать адрес кольца передачи команде **!xhci_transferring** .

dbgcmd

```
kd> !xhci_transferring 0xfffffa8005b2fe00

        [  0] NORMAL       0x000000011692fe00 CycleBit 1 IOC 0 BEI 0
InterrupterTarget 0 TransferLength    13 TDSize  0
        [  1] EVENT_DATA   0x000000011692fe10 CycleBit 1 IOC 1 BEI 0
InterrupterTarget 0 Data  0 0xfffffa8005986850 TotalBytes 13
```

```
        [  2] NORMAL       0x000000011692fe20 CycleBit 1 IOC 0 BEI 0
InterrupterTarget 0 TransferLength    13 TDSize  0
        [  3] EVENT_DATA   0x000000011692fe30 CycleBit 1 IOC 1 BEI 0
InterrupterTarget 0 Data  0 0xfffffa8005b96210 TotalBytes 13
        [  4] NORMAL       0x000000011692fe40 CycleBit 1 IOC 0 BEI 0
InterrupterTarget 0 TransferLength    13 TDSize  0
        [  5] EVENT_DATA   0x000000011692fe50 CycleBit 1 IOC 1 BEI 0
InterrupterTarget 0 Data  0 0xfffffa8005b96210 TotalBytes 13
```

## См. также

Расширения USB 3.0

**!xhci_dumpall**

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения USB 2.0

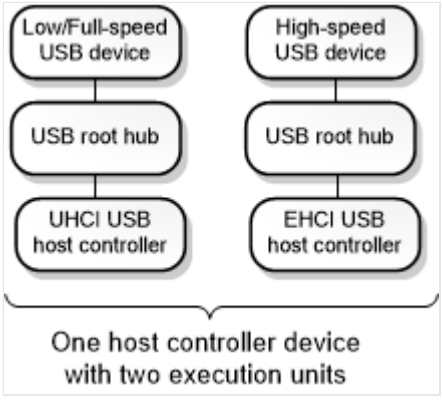
18.07.2025

В этом разделе описываются команды расширения отладчика USB 2.0. Эти команды отображают сведения из структур данных, поддерживаемых драйверами в стеке драйверов USB 2.0. Дополнительные сведения об этих трех драйверах см. в статье о драйверах на стороне USB в Windows.

Команды расширения отладчика USB 2.0 реализуются в Usbkd.dll. To load the Usbkd commands, enter **.load usbkd.dll** in the debugger.

## Дерево USB 2.0

Дерево USB 2.0 содержит узлы устройства, представляющие единицы выполнения на устройствах контроллера узла EHCI, а также дочерние узлы, представляющие концентраторы и подключенные устройства. На этой схеме показан пример дерева USB 2.0.




На схеме показано одно физическое устройство контроллера узла с двумя единицами выполнения. Каждая единица выполнения отображается как узел устройства в дереве устройств Plug and Play. Одна единица выполнения отображается как узел контроллера USB-узла UHCI, а другой блок выполнения отображается как узел контроллера USB-узла EHCI. Каждый из этих узлов имеет дочерний узел, представляющий корневой концентратор USB. Каждый корневой концентратор имеет один дочерний узел, представляющий подключенное USB-устройство.

Обратите внимание, что схема не является деревом в том смысле, что не все узлы спускаются от одного родительского узла. Однако при использовании *дерева USB 2.0* мы называем набор узлов устройств, представляющих единицы выполнения на устройствах контроллера узла EHCI, а также узлы для концентраторов и подключенных устройств.

# Начало работы с отладкой USB 2.0

To start debugging a USB 2.0 issue, enter the !usb2tree command. The **!usb2tree** command displays a list of commands and addresses that you can use to investigate host controllers, hubs, ports, devices, endpoints, and other elements of the USB 2.0 tree.

## В этом разделе

- !usbkd.usbhelp
- !usbkd._ehcidd
- !usbkd._ehciep
- !usbkd._ehciframe
- !usbkd._ehciqh
- !usbkd._ehciregs
- !usbkd._ehcisitd
- !usbkd._ehcistq
- !usbkd._ehcitd
- !usbkd._ehcitfer
- !usbkd._ehciitd
- !usbkd.doesdumphaveusbdata
- !usbkd.isthisdumpasyncissue
- !usbkd.urbfunc
- !usbkd.usb2
- !usbkd.usb2tree
- !usbkd.usbchain
- !usbkd.usbdevobj
- !usbkd.usbdpc
- !usbkd.ehci_info_from_fdo
- !usbkd.usbdevh
- !usbkd.usbep
- !usbkd.usbfaildata
- !usbkd.usbhcdext
- !usbkd.usbdstatus
- !usbkd.usbhcdhccontext
- !usbkd.usbhcdlist
- !usbkd.usbhcdlistlogs
- !usbkd.usbhcdlog
- !usbkd.usbhcdlogex
- !usbkd.usbhcdpnp
- !usbkd.usbhcdpow

- !usbkd.hub2_info_from_fdo
- !usbkd.usbhuberr
- !usbkd.usbhubext
- !usbkd.usbhubinfo
- !usbkd.usbhublog
- !usbkd.usbhubmddevext
- !usbkd.usbhubmdpd
- !usbkd.usbhubpd
- !usbkd.usbhubs
- !usbkd.usblist
- !usbkd.usbpo
- !usbkd.usbpdos
- !usbkd.usbpdoxls
- !usbkd.usbpnp
- !usbkd.usbportisasyncadv
- !usbkd.usbportmdportlog
- !usbkd.usbportmddcontext
- !usbkd.usbportmddevext
- !usbkd.usbtriage
- !usbkd.usbtt
- !usbkd.usbtx
- !usbkd.usbusb2ep
- !usbkd.usbusb2tt
- !usbkd.usbver

## See also

Расширения USB 3.0

RCDRKD Extensions

# !usbkd.usbhelp

Статья • 05.04.2024

Команда **!usbkd.usbhelp** отображает справку по командам расширения отладчика USB 2.0.

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehcidd

Статья • 05.04.2024

Команда **!usbkd._ehcidd** отображает сведения из **структуры usbehci!_DEVICE_DATA** .

dbgcmd

```
!usbkd._ehcidd StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbehci!_DEVICE_DATA**. Чтобы найти адреса **структур usbehci!_DEVICE_DATA**, используйте **!usbhcdext** или **!usbhcdlist.**

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов **получить адрес структуры usbehci!_DEVICE_DATA** . Сначала введите **!usbkd.usbhcdlist**.

dbgcmd

```
0: kd> !usbkd.usbhcdlist

MINIPORT List @ fffff80001e5bbd0

## List of EHCI controllers

!drvobj ffffe00001fd33a0 dt USBPORT!_USBPORT_MINIPORT_DRIVER
ffffe00001f48bd0 Registration Packet ffffe00001f48c08

01. Xxxx Corporation PCI: VendorID Xxxx DeviceID Xxxx RevisionId 0002
    !devobj ffffe0000781a050
    !ehci_info ffffe0000781a1a0
    Operational Registers ffffd00021fb8420
    Device Data ffffe0000781bda0
    ...
```

В предыдущих выходных **данных `ffffe0000781bda0` используется адрес структуры _DEVICE_DATA**.

Теперь передайте адрес **структуры в !_ehcidd**

dbgcmd

```
0: kd> !usbkd._ehcidd ffffe0000781bda0

*USBEHCI DEVICE DATA ffffe0000781bda0
** dt usbehci!_DEVICE_DATA ffffe0000781bda0

get_field_ulong ffffe0000781bda0 usbehci!_DEVICE_DATA Flags
*All Enpoints list:
head @ ffffe0000781bdb0 f_link ffffe0000781bdb0 b_link ffffe0000781bdb0
AsyncQueueHead ffffd00021cf5000 !_ehciqh ffffd00021cf5000
    PhysicalAddress: 0xde79a000
    NextQh: ffffd00021cf5000 Hlink de79a002
    PrevQh: ffffd00021cf5000
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehciep

Статья • 05.04.2024

Команда **!usbkd._ehciep** отображает сведения из **структуры usbehci!_ENDPOINT_DATA** . Используйте эту команду для отображения сведений об асинхронных конечных точках (то есть элементах управления и массовых конечных точках).

dbgcmd

```
!usbkd._ehciep StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbehci!_ENDPOINT_DATA**. Чтобы найти адреса **структур usbehci!_ENDPOINT_DATA**, используйте **!usbhcdext** и **!usblist.**

## DLL-библиотеки

Usbkd.dll

## Примеры

В этом примере показан один из способов **получения адреса структуры usbehci!_ENDPOINT_DATA** . Начните с команды !usb2tree**.**

dbgcmd

```
0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe0000206e1a0 !devobj ffffe0000206e050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000024a61a0 !devstack ffffe000024a6050
        Port 1: !port2_info ffffe000026dd000
        Port 2: !port2_info ffffe000026ddb40
        Port 3: !port2_info ffffe000026de680 !devstack ffffe00001ec3060
            !device2_info ffffe00001ec31b0 (USB Mass Storage Device: Xxx
Corporation)
        Port 4: !port2_info ffffe000026df1c0
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !ehci_info ffffe0000206e1a0. Щелкните команду DML или передайте адрес расширения устройства в !usbhcdext.

dbgcmd

```
0: kd> !usbkd.usbhcdext ffffe0000206e1a0
...
DeviceHandleList: !usblist ffffe0000206f3b8, DL
DeviceHandleDeletedList: !usblist ffffe0000206f3c8, DL [Empty]
GlobalEndpointList: !usblist ffffe0000206f388, EP
...
```

Предыдущие выходные данные отображают команду **!usblist ffffe0000206f388, EP**. Используйте эту команду для отображения списка конечных точек.

dbgcmd

```
0: kd> !usblist ffffe0000206f388, EP
list: ffffe0000206f388 EP
...
dt usbport!_HCD_ENDPOINT ffffe000026dc970  !usbep ffffe000026dc970
common buffer bytes 0x00003000 (12288) @ va 0000000021e6e000 pa
00000000d83c9000
Device Address: 0x01, ep 0x81 Bulk In Flags: 00000041 dt _USB_ENDPOINT_FLAGS
ffffe000026dc990
... usbehci!_ENDPOINT_DATA ffffe000026dcc38
...
```

В предыдущем выходных данных `ffffe000026dcc38` используется адрес **структуры usbehci!_ENDPOINT_DATA** . Передайте этот адрес в **!_ehciep**.

dbgcmd

```
0: kd> !usbkd._ehciep ffffe000026dcc38
*USBEHCI
dt usbehci!_ENDPOINT_DATA ffffe000026dcc38
Flags: 0x00000000
dt usbehci!_HCD_QUEUEHEAD_DESCRIPTOR ffffd00021e6e080
*HwQH ffffd00021e6e080
HwQH
     HwQH.HLink dea2e002
     HwQH.EpChars 02002101
         DeviceAddress: 0x1
         IBit: 0x0
         EndpointNumber: 0x1
         EndpointSpeed: 0x2 HcEPCHAR_HighSpeed
         DataToggleControl: 0x0
         HeadOfReclimationList: 0x0
```

```
          MaximumPacketLength: 0x200 - 512
    ...
current slot 0000000000000000
slot[0] dt usbehci!_ENDPOINT_SLOT ffffe000026dcdb8 - slot_NotBusy
----
     ffffd00021e6e100
     dt usbehci!_HCD_TRANSFER_DESCRIPTOR ffffd00021e6e100
    tdphys: d83c9100'200 txlen 00000000 tx ffffd00000000041 flags 6d4e695d
_BUSY _SLOT_RESET
    Next_qTD: d83c9200'd83c9180 AltNext_qTD 77423c00'41
    NextTD: ffffd00021e6e200 AltNextTD ffffd00021e6e180 SlotNextTd
ffffd00021e6e200 tok 00000c00  Xbytes x0 (0)
.
     ffffd00021e6e200
     dt usbehci!_HCD_TRANSFER_DESCRIPTOR ffffd00021e6e200
    tdphys: d83c9200'5000 txlen 00000000 tx ffffd00000000041 flags 6d4e695d
_BUSY _SLOT_RESET
    Next_qTD: d83c9280'd83c9180 AltNext_qTD 77423c00'41
    NextTD: ffffd00021e6e280 AltNextTD ffffd00021e6e180 SlotNextTd
ffffd00021e6e280 tok 00000c00  Xbytes x0 (0)
...
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehciframe

Статья • 05.04.2024

Команда **!usbkd._ehciframe** отображает EHCI miniport FrameListBaseAddress периодический цепочку входа списка, индексированную номером кадра.

dbgcmd

```
!usbkd._ehciframe StructAddr, FrameNumber
```

## Параметры

*StructAddr*

**Адрес структуры usbehci!_DEVICE_DATA**.

*FrameNumber*

Номер кадра в диапазоне от 0 до 1023.

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehciqh

Статья • 05.04.2024

Команда **!usbkd._ehciqh** отображает сведения из **структуры usbehci!_HCD_QUEUEHEAD_DESCRIPTOR** . Используйте эту команду для отображения сведений об асинхронных конечных точках (то есть элементах управления и массовых конечных точках).

dbgcmd

```
!usbkd._ehciqh StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbehci!_HCD_QUEUEHEAD_DESCRIPTOR.**

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehciregs

Статья • 05.04.2024

Команда **!usbkd._ehciregs** отображает регистры состояния рабочего и корневого порта контроллера узла USB EHCI.

```
dbgcmd
```

```
!usbkd._ehciregs StructAddr[, NumPorts]
```

## Параметры

*StructAddr*
**Адрес структуры usbehci!_HC_OPERATIONAL_REGISTER**. Чтобы найти адрес **структуры usbehci!_HC_OPERATIONAL_REGISTER** , используйте **!usbkd.usbhcdlist**.

*NumPorts*
Число регистров состояния порта корневого концентратора для отображения.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов **получить адрес структуры usbehci!_HC_OPERATIONAL_REGISTER** . Сначала введите **!usbkd.usbhcdlist**.

```
dbgcmd
```

```
0: kd> !usbkd.usbhcdlist
MINIPORT List @ fffff80001e5bbd0

## List of EHCI controllers

!drvobj ffffe00001fd33a0 dt USBPORT!_USBPORT_MINIPORT_DRIVER ...
...
02. Xxxx Corporation PCI: VendorID Xxxx DeviceID Xxxx RevisionId 0002
    !devobj ffffe00001ca1050
    !ehci_info ffffe00001ca11a0
    Operational Registers ffffd000228bf020
```

В предыдущих выходных **данных `ffffd000228bf020` используется адрес структуры _HC_OPERATIONAL_REGISTER**.

Теперь передайте адрес **структуры в !_ehciregs**. В этом примере второй аргумент ограничивает отображение двумя регистрами состояния порта корневого концентратора.

dbgcmd

```
0: kd> !usbkd._ehciregs ffffd000228bf020, 2
*(ehci)HC_OPERATIONAL_REGISTER ffffd000228bf020
    USBCMD 00010001
    .HostControllerRun: 1
    .HostControllerReset: 0
    .FrameListSize: 0
    .PeriodicScheduleEnable: 0
    .AsyncScheduleEnable: 0
    .IntOnAsyncAdvanceDoorbell: 0
    .HostControllerLightReset: 0
    .InterruptThreshold: 1
    .ParkModeEnable: 0
    .ParkModeCount: 0

    USBSTS 00002008
    .UsbInterrupt: 0
    .UsbError: 0
    .PortChangeDetect: 0
    .FrameListRollover: 1
    .HostSystemError: 0
    .IntOnAsyncAdvance: 0
    ----
    .HcHalted: 0
    .Reclimation: 1
    .PeriodicScheduleStatus: 0
    .AsyncScheduleStatus: 0

    USBINTR 0000003f
    .UsbInterrupt: 1
    .UsbError: 1
    .PortChangeDetect: 1
    .FrameListRollover: 1
    .HostSystemError: 1
    .IntOnAsyncAdvance: 1
    PeriodicListBase dec8e000
    AsyncListAddr dec91000
    PortSC[0] 00001000
        PortConnect x0
        PortConnectChange x0
        PortEnable x0
        PortEnableChange x0
        OvercurrentActive x0
        OvercurrentChange x0
        ForcePortResume x0
```

```
        PortSuspend x0
        PortReset x0
        HighSpeedDevice x0
        LineStatus x0
        PortPower x1
        PortOwnedByCC x0
        PortIndicator x0
        PortTestControl x0
        WakeOnConnect x0
        WakeOnDisconnect x0
        WakeOnOvercurrent x0
    PortSC[1] 00001000
        PortConnect x0
        PortConnectChange x0
        PortEnable x0
        PortEnableChange x0
        OvercurrentActive x0
        OvercurrentChange x0
        ForcePortResume x0
        PortSuspend x0
        PortReset x0
        HighSpeedDevice x0
        LineStatus x0
        PortPower x1
        PortOwnedByCC x0
        PortIndicator x0
        PortTestControl x0
        WakeOnConnect x0
        WakeOnDisconnect x0
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehcisitd

Статья • 05.04.2024

Команда **!usbkd._ehcisitd** отображает сведения из **usbehci!_HCD_SI_TRANSFER_DESCRIPTOR**

dbgcmd

```
!usbkd._ehcisitd StructAddr
```

## Параметры

*StructAddr*

**Адрес структуры usbehci!_HCD_SI_TRANSFER_DESCRIPTOR.**

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehcistq

Статья • 05.04.2024

Команда **!usbkd._ehcistq** отображает структуру **usbehci!_HCD_QUEUEHEAD_DESCRIPTOR** .

dbgcmd

```
!usbkd._ehciep StructAddr
```

## Параметры

*StructAddr*

**Адрес структуры usbehci!_HCD_QUEUEHEAD_DESCRIPTOR.**

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehcitd

Статья • 05.04.2024

Команда **!usbkd._ehcitd** отображает сведения из **структуры usbehci!_TRANSFER_CONTEXT** . Используйте эту команду для отображения сведений об асинхронных конечных точках (то есть элементах управления и массовых конечных точках).

dbgcmd

```
!usbkd._ehcitd StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbehci!_TRANSFER_CONTEXT**.

## DLL-библиотеки

Usbkd.dll

## Примеры

В этом примере показан один из способов **получения адреса структуры usbehci!_TRANSFER_CONTEXT** . Используйте !_ehciep для отображения сведений о конечной точке.

dbgcmd

```
0: kd> !_ehciep ffffe000001ab618
*USBEHCI
dt usbehci!_ENDPOINT_DATA ffffe000001ab618
Flags: 0x00000000
dt usbehci!_HCD_QUEUEHEAD_DESCRIPTOR ffffd00021e65080
*HwQH ffffd00021e65080
HwQH
     HwQH.HLink dea2e002
     HwQH.EpChars 02002201
         DeviceAddress: 0x1
         IBit: 0x0
         EndpointNumber: 0x2
    ...
slot[0] dt usbehci!_ENDPOINT_SLOT ffffe000001ab798 - slot_NotBusy
```

```
----
     ffffd00021e65100
     dt usbehci!_HCD_TRANSFER_DESCRIPTOR ffffd00021e65100
    ....
```

В предыдущем выходных данных `ffffd00021e65100` используется адрес **структуры usbehci!_TRANSFER_CONTEXT** . Передайте этот адрес **в !_ehcitd**.

dbgcmd

```
0: kd> !_ehcitd ffffd00021e65100
*USBEHCI TD 21e65100
Sig 20td
     qTD
     Next_qTD: d83cc200
     AltNext_qTD: d83cc180
     Token: 0x00000c00
         PingState: 0x0
         SplitXstate: 0x0
         MissedMicroFrame: 0x0
         XactErr: 0x0
         BabbleDetected: 0x0
         DataBufferError: 0x0
         Halted: 0x0
         Active: 0x0
         Pid: 0x0 - HcTOK_Out
         ErrorCounter: 0x3
         C_Page: 0x0
         InterruptOnComplete: 0x0
         BytesToTransfer: 0x0
         DataToggle: 0x0
     BufferPage[0]: 0x 0bad0-000  0bad0000  BufferPage64[0]: 00000000
     BufferPage[1]: 0x 0bad0-000  0bad0000  BufferPage64[1]: 00000000
     BufferPage[2]: 0x 0bad0-000  0bad0000  BufferPage64[2]: 00000000
     BufferPage[3]: 0x 0bad0-000  0bad0000  BufferPage64[3]: 00000000
     BufferPage[4]: 0x 0bad0-000  0bad0000  BufferPage64[4]: 00000000
Packet:00 52 e6 21 00 d0 ff ff
PhysicalAddress: d83cc100
EndpointData: 001ab618
TransferLength : 0000001f
TransferContext: 00000000
Flags: 00000041
    TD_FLAG_BUSY
NextHcdTD: 21e65200
AltNextHcdTD: 21e65180
SlotNextHcdTD: 21e65200
```

# См. также

Расширения отладчика USB 2.0

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehcitfer

Статья • 05.04.2024

Команда **!usbkd._ehcitfer** отображает сведения из **структуры usbehci!_HCD_TRANSFER_DESCRIPTOR** .

dbgcmd

```
!usbkd._ehcitfer StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbehci!_HCD_TRANSFER_DESCRIPTOR.**

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd._ehciitd

Статья • 05.04.2024

Команда **!usbkd._ehciitd** отображает сведения из **структуры usbehci!_HCD_HSISO_TRANSFER_DESCRIPTOR** .

```
dbgcmd
```

```
!usbkd._ehciitd StructAddr
```

## Параметры

*StructAddr*

**Адрес структуры usbehci!_HCD_HSISO_TRANSFER_DESCRIPTOR.**

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.doesdumphaveusbdata

Статья • 05.04.2024

**Команда !usbkd.doesdumphaveusbdata** проверка, чтобы узнать, какие типы USB-данных находятся в файле аварийного дампа, созданном **в результате проверки ошибок 0xFE**.

dbgcmd

```
!usbkd.doesdumphaveusbdata
```

## DLL-библиотеки

Usbkd.dll

## Замечания

Используйте эту команду только при отладке файла аварийного дампа, созданного **в результате проверки ошибок 0xFE: BUGCODE_USB_DRIVER**.

## Примеры

Ниже приведен пример выходных **данных !doesdumphaveusbdata**

dbgcmd

```
1: kd> !analyze -v
*** ...
BUGCODE_USB_DRIVER (fe)
...
1: kd> !usbkd.doesdumphaveusbdata

Retrieving crashdump information Please Wait...

Checking for GuidUsbHubPortArrayData information...
There is no data for this GUID in the mini dump.
No data to print

Checking for GuidUsbHubExt information...
There is no data for this GUID in the mini dump.
No data to print

Checking for GuidUsbPortLog information...
GuidUsbPortLog Exists with PORT Log Size = 8000
```

```
Checking for GuidUsbPortContextData information...
GuidUsbPortContextData Exists with Data Length = 4c8

Checking for GuidUsbPortExt information...
GuidUsbPortExt Exists (DEVICE_EXTENSION + DeviceDataSize ) = 2250
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.isthisdumpasyncissue

Статья • 05.04.2024

**Команда !usbkd.isthisdumpasyncissue** проверка файла аварийного дампа, созданного **0xFE** проверки ошибок, чтобы узнать, была ли причина сбоя прерыванием в Async Advance, связанной с контроллером узла USB EHCI.

dbgcmd

```
!usbkd.isthisdumpasyncissue
```

## DLL-библиотеки

Usbkd.dll

## Замечания

Используйте эту команду только при отладке файла аварийного дампа, созданного **в результате проверки ошибок 0xFE: BUGCODE_USB_DRIVER**.

## Примеры

Ниже приведен пример выходных данных **!usbkd.isthisdumpasyncissue**.

dbgcmd

```
1: kd> !analyze -v
*** ...
BUGCODE_USB_DRIVER (fe)
...
1: kd> !usbkd.isthisdumpasyncissue
This is *NOT* Async on Advance Issue because the EndPointData is NULL
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.urbfunc

Статья • 05.04.2024

Команда **!usbkd.urbfunc** отображает имя кода функции URB.

```
dbgcmd

!usbkd.urbfunc FunctionCode
```

## Параметры

*FunctionCode*
Шестнадцатеричное значение кода функции URB. Эти коды определены в usb.h.

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен пример выходных **данных !urbfunc**.

```
dbgcmd

0: kd> !usbkd.urbfunc 0xA

URB_FUNCTION_ISOCH_TRANSFER (0xA)
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !usbkd.usb2

Статья • 05.04.2024

Команда **!usbkd.usb2** отображает список конечных точек USB с сведениями о планировании USB 2.0.

```
dbgcmd
```

```
!usbkd.usb2 DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства (FDO) контроллера USB-узла.

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен один из способов найти адрес расширения устройства для FDO контроллера USB-узла. Сначала введите **!usbkd.usb2tree**.

```
dbgcmd
```

```
0: kd> !usbkd.usb2tree

EHCI MINIPORT(s) dt usbport!_USBPORT_MINIPORT_DRIVER ffffe00001f48bd0
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !ehci_info ffffe00001ca11a0. Передайте адрес расширения устройства команде **!usb2** .

```
dbgcmd
```

```
0: kd> !usbkd.usb2 ffffe00001ca11a0

Sig: HFDO
Hcd FDO Extension:
----------
----------
dt usbport!_HCD_ENDPOINT ffffe0000212d970  !usbep ffffe0000212d970
    Tt 0000000000000000 Device Address: 0x00, ep 0x81 Interrupt In
    dt _USB2LIB_ENDPOINT_CONTEXT ffffe000023b60f0    dt _USB2_EP
ffffe000023b6100
    Period,offset,Ordinal(32,0,0)   smask,cmask(00,00  ........ , ........)
maxpkt 1
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usb2tree

Статья • 05.04.2024

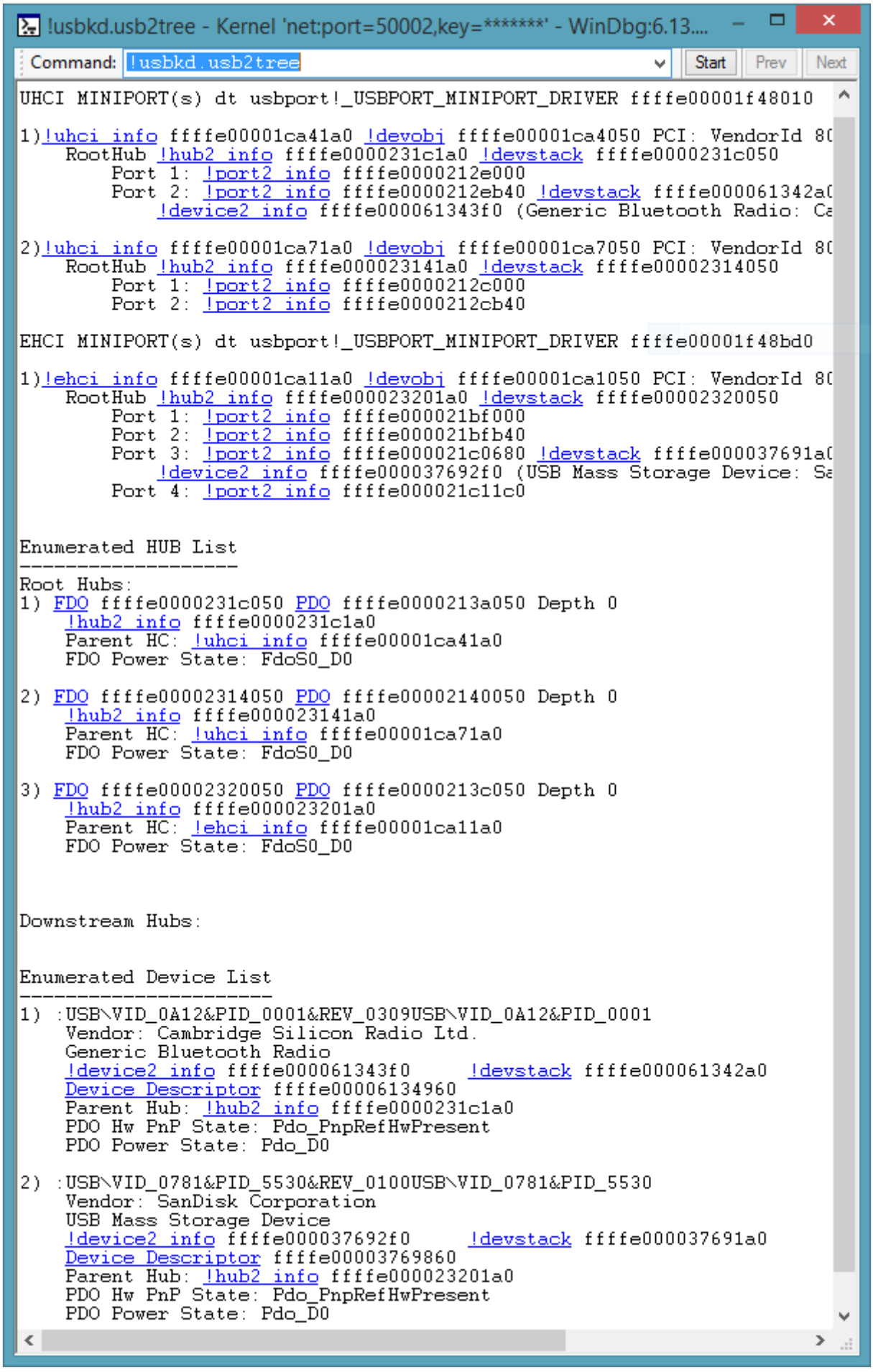
Команда **!usbkd.usb2tree** отображает дерево USB 2.0.

```
dbgcmd
```

```
!usbkd.usb2tree
```

## Примеры

На снимке экрана показан и пример выходных **данных команды !usb2tree** .




В выходных данных показана одна единица выполнения EHCI и две единицы выполнения UHCI. Единицы выполнения, показанные в этом примере, будут

находиться на одном устройстве контроллера USB-узла. Выходные данные также отображают корневые концентраторы и подключенные устройства.

Выходные данные используют язык разметки отладчика (DML) для предоставления ссылок. Ссылки выполняют команды, которые предоставляют подробные сведения, связанные с объектами в дереве. Например, можно щелкнуть одну из **ссылок !devobj** , чтобы получить сведения об объекте функционального устройства, связанном с единицей выполнения EHCI. В качестве альтернативы щелчку ссылки можно ввести команду вручную: **!devobj ffffe00001ca7050**

**Обратите внимание** , что функция DML доступна в WinDbg, но не в Visual Studio или KD.

# DLL-библиотеки

Usb3kd.dll

# Замечания

**Команда !usb2tree** — это родительская команда для многих команд отладчика USB 2.0. Сведения, отображаемые этими командами, основаны на структурах данных, поддерживаемых этими драйверами:

- usbehci.sys (мини-драйвер для контроллера узла USB 2)
- usbuhci.sys (мини-драйвер для контроллера узла USB 2)
- usbport.sys (драйвер порта для контроллера узла USB 2)
- usbhub.sys (драйвер концентратора USB 2)

Дополнительные сведения об этих драйверах см. в статье о драйверах на стороне USB-узла в Windows.

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

**Были ли сведения на этой странице полезными?**

Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbchain

Статья • 05.04.2024

Команда **!usbkd.usbchain** отображает цепочку USB-устройств, начиная с указанного PDO, и вернитесь в корневой концентратор.

```
dbgcmd
```

```
!usbkd.usbchain PDO
```

## Параметры

*PDO*
Адрес объекта физического устройства (PDO) устройства, подключенного к USB-концентратору.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов найти адрес PDO USB-устройства. Сначала введите **!usbkd.usb2tree**.

```
dbgcmd
```

```
 kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
        Port 1: !port2_info ffffe000021bf000
        Port 2: !port2_info ffffe000021bfb40
        Port 3: !port2_info ffffe000021c0680 !devstack ffffe00007c882a0
...
```

В предыдущих выходных данных адрес PDO является аргументом предлагаемой команды **!devstack ffffe00007c882a0**. Передайте адрес PDO в **!usbkd.usbchain**.

```
dbgcmd
```

```
0: kd> !usbkd.usbchain ffffe00007c882a0

usbchain
**************************************************************************
*
HUB PDO ffffe00007c882a0 on port 3 !usbhubext ffffe00007c883f0 ArmedForWake
= 0
VID Xxxx PID Xxxx REV 0100  Xxxx Corporation
    HUB #3 FDO ffffe00002320050 , !usbhubext ffffe000023201a0  HWC_ARM=0
    ROOT HUB PDO(ext) @ffffe0000213c1a0
        ROOT HUB FDO @ffffe00001ca1050, !usbhcdext ffffe00001ca11a0 PCI
Vendor:Device:...
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbdevobj

Статья • 05.04.2024

Команда **!usbkd.usbdevobj** отображает сведения из объекта USB-устройства.

```
dbgcmd
```

```
!usbkd.usbdevobj DeviceObject
```

## Параметры

*DeviceObject*
**Адрес структуры _DEVICE_OBJECT**.

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbdpc

Статья • 05.04.2024

Команда **!usbkd.usbdpc** отображает сведения, хранящиеся в **структуре _XDPC_CONTEXT** .

dbgcmd

```
!usbkd.usbdpc StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbport!_XDPC_CONTEXT**. Чтобы получить список XDPC для usb-контроллера узла, используйте **команду !usbkd.usbhcdext** .

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов **найти адрес usbport!_XDPC_CONTEXT** структуры. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...
UHCI MINIPORT(s) dt usbport!_USBPORT_MINIPORT_DRIVER ffffe00001e77010
...
4)!uhci_info ffffe00001c7d1a0 !devobj ffffe00001c7d050 PCI: VendorId...
...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !uhci_info ffffe00001c7d1a0.

Щелкните команду DML или передайте адрес расширения **устройства в !usbhcdext** , чтобы получить список XDPC.

dbgcmd

```
0: kd> !usbkd.usbhcdext ffffe00001c7d1a0
...
## XDPC List

01) dt USBPORT!_XDPC_CONTEXT ffffe00001c7df18
02) dt USBPORT!_XDPC_CONTEXT ffffe00001c7db88
03) dt USBPORT!_XDPC_CONTEXT ffffe00001c7dd50
04) dt USBPORT!_XDPC_CONTEXT ffffe00001c7e0e0
...
```

В предыдущих выходных **данных** `ffffe00001c7df18` **— это адрес структуры** **_XDPC_CONTEXT**. Передайте этот адрес **в !usbdpc**.

dbgcmd

```
0: kd> !usbkd.usbdpc ffffe00001c7df18

dt USBPORT!_XDPC_CONTEXT ffffe00001c7df18

## XDPC HISTORY (latest at boottom)

##      EVENT               STATE                   NEXT

[01] Ev_Xdpc_End          XDPC_Running            XDPC_Enabled
[02] Ev_Xdpc_Signal       XDPC_Enabled            XDPC_DpcQueued
[03] Ev_Xdpc_Signal       XDPC_DpcQueued          XDPC_DpcQueued
[04] Ev_Xdpc_Worker       XDPC_DpcQueued          XDPC_Running
[05] Ev_Xdpc_Signal       XDPC_Running            XDPC_Signaled
[06] Ev_Xdpc_End          XDPC_Signaled           XDPC_DpcQueued
[07] Ev_Xdpc_Worker       XDPC_DpcQueued          XDPC_Running
[08] Ev_Xdpc_End          XDPC_Running            XDPC_Enabled
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.ehci_info_from_fdo

Статья • 22.04.2025

Команда **!usbkd.ehci_info_from_fdo** отображает сведения о контроллере USB-узла.

```
dbgcmd

!usbkd.ehci_info_from_fdo fdo
```

## Параметры

*fdo*
Адрес функционального объекта устройства (FDO) контроллера узла UHCI или EHCI USB. Адрес FDO можно получить из выходных данных команды **!usb2tree** .

## DLL

Usbkd.dll

## Примеры

Сначала используйте команду **!usb2tree** , чтобы получить адрес FDO.

```
dbgcmd

0: kd> !usbkd.usb2tree

EHCI MINIPORT(s) dt usbport!_USBPORT_MINIPORT_DRIVER ffffe00001f48bd0

1)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086 DeviceId
293c RevisionId 0002
...
```

В предыдущих выходных данных можно увидеть, что адрес FDO контроллера `ffffe00001ca1050` USB-узла . Передайте адрес FDO в **!ehci_info_from_fdo**.

```
dbgcmd

0: kd> !usbkd.ehci_info_from_fdo ffffe00001ca1050

HC Flavor 1000  FDO ffffe00001ca1050
Root Hub: FDO ffffe00002320050 !hub2_info ffffe000023201a0
Operational Registers ffffd000228bf020
```

```
Device Data ffffe00001ca2da0
dt USBPORT!_FDO_EXTENSION ffffe00001ca15a0
DM Timer Flags ffffe00001ca16d4
FDO Flags ffffe00001ca16d0
HCD Log ffffe00001ca11a0

DeviceHandleList: !usblist ffffe00001ca23b8, DL
DeviceHandleDeletedList: !usblist ffffe00001ca23c8, DL [Empty]
GlobalEndpointList: !usblist ffffe00001ca2388, EP
EpNeoStateChangeList: !usblist ffffe00001ca2370, SC [Empty]
GlobalTtListHead: !usblist ffffe00001ca23a8, TT [Empty]
BusContextHead: !usblist ffffe00001ca16b0, BC

## Pending Requests

[001] dt USBPORT!_USB_IOREQUEST_CONTEXT ffffe00001ca1450 Tag: AddD Obj:
ffffe00001ca11a0
...

## XDPC List

01) dt USBPORT!_XDPC_CONTEXT ffffe00001ca1f18
...

## PnP FUNC HISTORY (latest at bottom)

[01] IRP_MN_QUERY_CAPABILITIES
...
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# !usbkd.usbdevh

Статья • 05.04.2024

Команда **!usbkd.usbdevh** отображает сведения об дескрипторе USB-устройства.

```
dbgcmd

!usbkd.usbdevh StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbport!_USBD_DEVICE_HANDLE**. Чтобы получить список дескрипторов устройства для контроллера USB-узла, используйте **команду !usbkd.usbhcdext** .

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен один из способов **найти адрес структуры usbport!_USBD_DEVICE_HANDLE** . Сначала введите **!usbkd.usb2tree**.

```
dbgcmd

0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
     ...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !ehci_info ffffe00001ca11a0.

Щелкните команду DML или передайте адрес расширения **устройства в !usbhcdext** , чтобы получить список дескрипторов устройства.

```
dbgcmd
```

```
0: kd> !usbkd.usbhcdext ffffe00001ca11a0

HC Flavor 1000  FDO ffffe00001ca1050
Root Hub: FDO ffffe00002320050 !hub2_info ffffe000023201a0
Operational Registers ffffd000228bf020
Device Data ffffe00001ca2da0
dt USBPORT!_FDO_EXTENSION ffffe00001ca15a0
DM Timer Flags ffffe00001ca16d4
FDO Flags ffffe00001ca16d0
HCD Log ffffe00001ca11a0

DeviceHandleList: !usblist ffffe00001ca23b8, DL
DeviceHandleDeletedList: !usblist ffffe00001ca23c8, DL [Empty]
...
```

Теперь используйте **команду !usbkd.usblist** , чтобы получить адреса **структур usbport!_USBD_DEVICE_HANDLE** .

dbgcmd

```
0: kd> !usblist ffffe00001ca23b8, DL
list: ffffe00001ca23b8 DL
----------
!usbdevh ffffe000020f9590
SSP [IdleReady] (0)
...
```

В предыдущем выходных данных `ffffe000020f9590` используется адрес **структуры _USBD_DEVICE_HANDLE** . Передайте этот адрес **в !usbdevh**.

dbgcmd

```
0: kd> !usbkd.usbdevh ffffe000020f9590

dt USBPORT!_USBD_DEVICE_HANDLE ffffe000020f9590
SSP [IdleReady] (0)
PCI\VEN_8086&DEV_293C  Intel Corporation
Root Hub
DriverName :

## DEVICE HANDLE HISTORY (latest at boottom)

##      EVENT                         STATE                          NEXT

[01] Ev_CreateRoothub_Success     Devh_Created                 Devh_Valid

## Referene List: Head(ffffe000020f9668)

[00] dt USBPORT!_DEVH_REF_OBJ ffffe000021944a0  Object: ffffe000020f9590
Tag: dvh+ PendingFlag(0)
```

```
[01] dt USBPORT!_DEVH_REF_OBJ ffffe000020bbcb0  Object: ffffe000020ba7e0
Tag: bsCT PendingFlag(0)
[02] dt USBPORT!_DEVH_REF_OBJ ffffe000032b91a0  Object: ffffe0000269e670
Tag: UrbT PendingFlag(1)

## TtList: Head(ffffe000020f9658)


## PipeHandleList: Head(ffffe000020f9640)

[00] dt USBPORT!_USBD_PIPE_HANDLE_I ffffe000020f95e0 !usbep ffffe000020f6970
        Device Address: 0x00, Endpoint Address 0x00 Endpoint Type: Control
[01] dt USBPORT!_USBD_PIPE_HANDLE_I ffffe000023bd278 !usbep ffffe0000212d970
        Device Address: 0x00, Endpoint Address 0x81 Endpoint Type: Interrupt
In

Config Information: dt USBPORT!_USBD_CONFIG_HANDLE ffffe000023cd0b0

## Interface List:

[00] dt USBPORT!_USBD_INTERFACE_HANDLE_I ffffe000023bd250
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbep

Статья • 05.04.2024

Команда **!usbkd.usbep** отображает сведения о конечной точке USB.

```
dbgcmd

!usbkd.usbep StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbport!_HCD_ENDPOINT**. Чтобы получить список конечных точек для контроллера USB-узла, используйте **команду !usbkd.usbhcdext** .

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен один из способов **найти адрес структуры usbport!_HCD_ENDPOINT** . Сначала введите **!usbkd.usb2tree**.

```
dbgcmd

0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
     ...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !ehci_info ffffe00001ca11a0.

Щелкните команду DML или передайте адрес расширения **устройства в !usbhcdext** , чтобы получить глобальный список конечных точек.

```
dbgcmd
```

```
0: kd> !usbkd.usbhcdext ffffe00001ca11a0
...
DeviceHandleList: !usblist ffffe00001ca23b8, DL
DeviceHandleDeletedList: !usblist ffffe00001ca23c8, DL [Empty]
GlobalEndpointList: !usblist ffffe00001ca2388, EP
...
```

Теперь используйте **команду !usbkd.usblist**, чтобы получить адреса _HCD_ENDPOINT структур.

dbgcmd

```
0: kd> !usblist ffffe00001ca2388, EP

list: ffffe00001ca2388 EP
----------
dt usbport!_HCD_ENDPOINT ffffe000020f6970  !usbep ffffe000020f6970
Device Address: 0x00, ep 0x00 Control  Flags: 00000002 dt
_USB_ENDPOINT_FLAGS ffffe000020f6990
dt usbport!_ENDPOINT_PARAMETERS ffffe000020f6b18    RootHub Endpoint
...
```

В предыдущих выходных **данных `ffffe000020f6970` — это адрес структуры _HCD_ENDPOINT**. Передайте этот адрес в **!usbkd.usbep**.

dbgcmd

```
0: kd> !usbep ffffe000020f6970
Device Address: 0x00, Endpoint Address 0x00 Endpoint Type: Control
dt USBPORT!_HCD_ENDPOINT ffffe000020f6970
dt USBPORT!_ENDPOINT_PARAMETERS ffffe000020f6b18
RootHub Endpoint

## Transfer(s) List: (HwPendingListHead)

    [EMPTY]

## Endpoint Reference List: (EpRefListHead)

[00] dt USBPORT!_USBOBJ_REF ffffe000021a64a0 Object ffffe000020f6970
Tag:EPop Endpoint:ffffe000020f6970
[01] dt USBPORT!_USBOBJ_REF ffffe000021264a0 Object ffffe000020f95e0
Tag:EPpi Endpoint:ffffe000020f6970

## GEP HISTORY (latest at bottom)

##      EVENT                   STATE                     NEXT
HwEpState

[01] Ev_gEp_Open             GEp_Init                  GEp_Paused
```

```
ENDPOINT_PAUSE
[02] Ev_gEp_ReqActive        GEp_Paused              GEp_Active
ENDPOINT_ACTIVE
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbfaildata

Статья • 05.04.2024

Команда **!usbkd.usbfaildata** отображает данные об ошибках (если таковые) хранятся для USB-устройства.

dbgcmd

```
!usbkd.usbfaildata PDO
```

## Параметры

*PDO*
Адрес объекта физического устройства (PDO) устройства, подключенного к USB-концентратору.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов найти адрес PDO USB-устройства. Введите **!usbkd.usb2tree**.

dbgcmd

```
 kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
        Port 1: !port2_info ffffe000021bf000
        Port 2: !port2_info ffffe000021bfb40
        Port 3: !port2_info ffffe000021c0680 !devstack ffffe00007c882a0
...
```

В предыдущих выходных данных адрес PDO отображается в качестве аргумента предлагаемой команды **!devstack ffffe00007c882a0**.

Теперь передайте адрес PDO в **!usbkd.usbfaildata**.

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhcdext

Статья • 22.04.2025

Команда **!usbkd.usbhcdext** отображает сведения из расширения устройства контроллера USB-узла или корневого концентратора USB.

```
dbgcmd
```

```
!usbkd.usbhcdext DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес одного из следующих элементов:

- Расширение устройства для функционального объекта устройства (FDO) контроллера USB-узла.
- Расширение устройства для объекта физического устройства (PDO) корневого концентратора USB.

## DLL

Usbkd.dll

## Примеры

Ниже приведен один из способов найти адрес расширения устройства для FDO узла EHCI. Сначала введите **!usbkd.usb2tree**.

```
dbgcmd
```

```
0: kd> !usbkd.usb2tree

EHCI MINIPORT(s) dt usbport!_USBPORT_MINIPORT_DRIVER ffffe00001f48bd0

1)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086 DeviceId
293c RevisionId 0002
 ...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML**!ehci_info ffffe00001ca11a0**.

Теперь передайте адрес расширения устройства команде **!usbhcdext** .

dbgcmd

```
0: kd> !usbkd.usbhcdext ffffe00001ca11a0

HC Flavor 1000  FDO ffffe00001ca1050
Root Hub: FDO ffffe00002320050 !hub2_info ffffe000023201a0
Operational Registers ffffd000228bf020
Device Data ffffe00001ca2da0
dt USBPORT!_FDO_EXTENSION ffffe00001ca15a0
DM Timer Flags ffffe00001ca16d4
FDO Flags ffffe00001ca16d0
HCD Log ffffe00001ca11a0

DeviceHandleList: !usblist ffffe00001ca23b8, DL
DeviceHandleDeletedList: !usblist ffffe00001ca23c8, DL [Empty]
GlobalEndpointList: !usblist ffffe00001ca2388, EP
EpNeoStateChangeList: !usblist ffffe00001ca2370, SC [Empty]
GlobalTtListHead: !usblist ffffe00001ca23a8, TT [Empty]
BusContextHead: !usblist ffffe00001ca16b0, BC

## Pending Requests

[001] dt USBPORT!_USB_IOREQUEST_CONTEXT ffffe00001ca1450 Tag: AddD Obj:
ffffe00001ca11a0
...

## XDPC List

01) dt USBPORT!_XDPC_CONTEXT ffffe00001ca1f18
...
```

Ниже приведен один из способов найти адрес расширения устройства для PDO корневого концентратора. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree

EHCI MINIPORT(s) dt usbport!_USBPORT_MINIPORT_DRIVER ffffe00001f48bd0

1)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086 DeviceId
293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
 ...
```

В предыдущих выходных данных можно увидеть адрес FDO корневого концентратора, отображаемый в качестве аргумента команды **!devstack ffffe00002320050**. Используйте команду **!devstack** , чтобы найти адрес PDO и расширения устройства PDO.

dbgcmd

```
0: kd> !kdexts.devstack ffffe00002320050
  !DevObj           !DrvObj            !DevExt           ObjectName
> ffffe00002320050  \Driver\usbhub     ffffe000023201a0  0000002d
  ffffe0000213c050  \Driver\usbehci    ffffe0000213c1a0  USBPDO-3
...
```

В предыдущих выходных данных можно увидеть, что адрес расширения устройства для PDO корневого концентратора `ffffe0000213c1a0`.

Теперь передайте адрес расширения устройства команде **!usbhcdext** .

dbgcmd

```
0: kd> !usbkd.usbhcdext ffffe0000213c1a0

Root Hub PDO Extension
Parent HC: FDO ffffe00001ca1050 !ehci_info ffffe00001ca11a0
HUB FDO ffffe00002320050 !hub2_info ffffe000023201a0
dt USBPORT!_PDO_EXTENSION ffffe0000213c5a0

## Pending Requests

[001] dt USBPORT!_USB_IOREQUEST_CONTEXT ffffe0000213c450 Tag: RHcr Obj:
ffffe0000213c1a0
[002] dt USBPORT!_USB_IOREQUEST_CONTEXT ffffe00003ce5800 Tag: iIRP Obj:
ffffe00002182210

## POWER FUNC HISTORY (latest at bottom)

[00] IRP_MN_WAIT_WAKE (PowerSystemHibernate)
...

## PnP STATE LOG (latest at bottom)

##      EVENT                          STATE               NEXT

[01] EvPDO_IRP_MN_START_DEVICE      PnpNotStarted       PnpStarted
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# !usbkd.usbdstatus

Статья • 05.04.2024

Команда **!usbkd.usbdstatus** отображает имя кода состояния USBD.

```
dbgcmd

!usbkd.usbdstatus StatusCode
```

## Параметры

*StatusCode*
Шестнадцатеричное значение кода состояния USBD. Эти коды определены в usb.h.

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен пример выходных **данных !usbdstatus**.

```
dbgcmd

1: kd> !usbkd.usbdstatus 0xC0000008

USBD_STATUS_DATA_OVERRUN (0xC0000008)
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !usbkd.usbhcdhccontext

Статья • 05.04.2024

Команда **!usbkd.usbhcdhccontext** отображает **USB2LIB_HC_CONTEXT** для контроллера USB-узла.

dbgcmd

```
!usbkd.usbhcdhccontext DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства (FDO) контроллера USB-узла.

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhcdlist

Статья • 22.04.2025

Команда **!usbkd.usbhcdlist** отображает сведения обо всех контроллерах USB-узла, представленных драйвером USB-порта (Usbport.sys). Сведения о драйвере USB-порта и связанных драйверах мини-порта см. в статье о драйверах usb-узла в Windows.

dbgcmd

```
!usbkd.usbhcdlist
```

## DLL

Usbkd.dll

## Примеры

Ниже приведен пример части выходных данных **!usbhcdlist**.

dbgcmd

```
0: kd> !usbkd.usbhcdlist
MINIPORT List @ fffff80001e5bbd0

## List of UHCI controllers

!drvobj ffffe00002000060 dt USBPORT!_USBPORT_MINIPORT_DRIVER ffffe00001f48010
Registration Packet ffffe00001f48048

01
...

## List of EHCI controllers

!drvobj ffffe00001fd33a0 dt USBPORT!_USBPORT_MINIPORT_DRIVER ffffe00001f48bd0
Registration Packet ffffe00001f48c08

01. Xxxxx Corporation PCI: VendorID Xxxx DeviceID Xxxx RevisionId 0002
    !devobj ffffe00001ca1050
    !ehci_info ffffe00001ca11a0
    Operational Registers ffffd000228bf020
    Device Data ffffe00001ca2da0
    !usbhcdlog ffffe00001ca11a0
    nt!_KINTERRUPT ffffe000020abe78
    Device Capabilities ffffe00001ca135c
    Pending IRP's: 0, Transfers: 0 (Periodic(0), Async(0))
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# !usbkd.usbhcdlistlogs

Статья • 16.07.2024

Команда **!usbkd.usbhcdlistlogs** отображает список всех функциональных объектов устройств (FDO), связанных с драйвером USB-порта (Usbport.sys). Команда также отображает полные журналы отладки для всех контроллеров узлов EHCI.

dbgcmd

```
!usbkd.usbhcdlistlogs
```

## DLL-библиотеки

Usbkd.dll

## Примеры

В этом примере показана часть выходных данных **команды !usbhcdlistlogs** .

dbgcmd

```
0: kd> !usbkd.usbhcdlistlogs

MINIPORT List @ fffff80001e5bbd0
*flink, blink ffffe00001f48018,ffffe00001f48bd8

[0] entry @: ffffe00001f48010 dt usbport!_USBPORT_MINIPORT_DRIVER
ffffe00001f48010
UHCI MINIPORT @ ffffe00001f48010 !drvobj ffffe00002000060 regpkt
ffffe00001f48048
02. !devobj ffffe00001ca4050 !usbhcdext ffffe00001ca41a0  HFDO 8086 2938
RHPDO ffffe0000213a050
03. !devobj ffffe00001ca7050 !usbhcdext ffffe00001ca71a0  HFDO 8086 2937
RHPDO ffffe00002140050

[1] entry @: ffffe00001f48bd0 dt usbport!_USBPORT_MINIPORT_DRIVER
ffffe00001f48bd0
EHCI MINIPORT @ ffffe00001f48bd0 !drvobj ffffe00001fd33a0 regpkt
ffffe00001f48c08
01. !devobj ffffe00001ca1050 !usbhcdext ffffe00001ca11a0  HFDO 8086 293c
RHPDO ffffe0000213c050

LOG@: ffffe00001ca11b8
>LOG mask = 3ff idx = fff69e8d (28d)
*LOG: ffffe000020191a0  LOGSTART: ffffe00002014000 *LOGEND: ffffe0000201bfe0
# -1
```

```
[ 000] ffffe000020191a0 Tmt0 0000000000000000 0000000000000000
0000000000000000
[ 001] ffffe000020191c0 _Pop 0000000000000003 0000000000003000
0000000000000000
...
[ 005] ffffe00002019240 chgZ 0000000000000000 0000000000b7228f
0000000000000000
[ 006] ffffe00002019260 nes- 0000000000000000 0000000000000000
0000000000000000
[ 007] ffffe00002019280 nes+ ffffe00001ca2370 ffffe00001ca2370
00000000000ced39
...
[1014] ffffe00002019060 tmo4 0000000000000000 0000000000000040
ffffe00001ca1c20
...
[1022] ffffe00002019160 xdw2 ffffe00001ca1b88 ffffe00001ca1050
0000000000000002
[1023] ffffe00002019180 xdB0 ffffe00001ca1b88 ffffe00001ca1050
0000000000000000
```

В выходных данных команды показаны два FDOs, представляющих контроллеры узлов UHCI и один FDO, представляющий контроллер узла EHCI. Затем выходные данные показывают журнал отладки для одного контроллера узла EHCI.

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhcdlog

Статья • 22.04.2025

Команда **!usbkd.usbhcdlog** отображает часть журнала отладки для контроллера USB-узла.

dbgcmd

```
!usbkd.usbhcdlog DeviceExtension[, NumberOfEntries]
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства (FDO) контроллера usb-узла UHCI или EHCI.

*NumberOfEntries*
Число отображаемых записей журнала. Чтобы отобразить весь журнал, задайте для этого параметра значение -1.

## DLL

Usbkd.dll

## Примеры

Ниже приведен один из способов найти адрес расширения устройства для FDO контроллера USB-узла. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0 kd> !usbkd.usb2tree

EHCI MINIPORT(s) dt usbport!_USBPORT_MINIPORT_DRIVER ffffe00001f48bd0
...

2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086 DeviceId
293c RevisionId 0002
...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML**!ehci_info ffffe00001ca11a0**.

Теперь передайте адрес расширения устройства в команду **!usbhcdlog** . В этом примере второй аргумент ограничивает отображение четырех записей журнала.

dbgcmd

```
0: kd> !usbkd.usbhcdlog ffffe00001ca11a0, 4

LOG@: ffffe00001ca11b8
>LOG mask = 3ff idx = fff68e95 (295)
*LOG: ffffe000020192a0  LOGSTART: ffffe00002014000 *LOGEND: ffffe0000201bfe0 # 4
[ 000] ffffe000020192a0 xSt0 ffffe00001ca1b88 0000000000000006 0000000000000001
[ 001] ffffe000020192c0 xnd8 ffffe00001ca1b88 ffffe00001ca1050 0000000000000000
[ 002] ffffe000020192e0 xnd0 ffffe00001ca1b88 ffffe00001ca1050 0000000000000000
[ 003] ffffe00002019300 gNX0 0000000000000000 0000000000000000 ffffe00001ca1b88
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# !usbkd.usbhcdlogex

Статья • 05.04.2024

Команда **!usbkd.usbhcdlogex** отображает аннотированный журнал отладки для контроллера USB-узла.

dbgcmd

```
!usbkd.usbhcdlogex DeviceExtension[, NumberOfEntries]
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства (FDO) контроллера usb-узла UHCI или EHCI.

*NumberOfEntries*
Число отображаемых записей журнала. Чтобы отобразить весь журнал, задайте для этого параметра значение -1.

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен один из способов найти адрес расширения устройства для FDO контроллера USB-узла. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0 kd> !usbkd.usb2tree

EHCI MINIPORT(s) dt usbport!_USBPORT_MINIPORT_DRIVER ffffe00001f48bd0
...

2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !ehci_info ffffe00001ca11a0.

Теперь передайте адрес расширения **устройства в команду !usbhcdlogex** . В этом примере второй аргумент ограничивает отображение 20 записей журнала.

dbgcmd

```
0: kd> !usbkd.usbhcdlogex ffffe00001ca11a0, 20
LOG@: ffffe00001ca11b8
>LOG mask = 3ff idx = fff68e95 (295)
*LOG: ffffe000020192a0  LOGSTART: ffffe00002014000 *LOGEND: ffffe0000201bfe0
# 20
[ 000] ffffe000020192a0 xSt0 ffffe00001ca1b88 0000000000000006
0000000000000001
[ 001] ffffe000020192c0 xnd8 ffffe00001ca1b88 ffffe00001ca1050
0000000000000000
[ 002] ffffe000020192e0 xnd0 ffffe00001ca1b88 ffffe00001ca1050
0000000000000000
//
// USBPORT_Xdpc_End()    - USBPORT_Core_UsbHcIntDpc_Worker() DPC/XDPC: 2 of
4

[ 003] ffffe00002019300 gNX0 0000000000000000 0000000000000000
ffffe00001ca1b88
[ 004] ffffe00002019320 xbg1 ffffe00001ca1b88 ffffe00001ca1050
0000000000000000
[ 005] ffffe00002019340 xbg0 ffffe00001ca1b88 ffffe00001ca1050
ffffe00001ca22e8
//
// USBPORT_Xdpc_iBegin() - USBPORT_Core_UsbHcIntDpc_Worker() DPC/XDPC: 2 of
4

[ 006] ffffe00002019360 tmo4 0000000000000000 0000000000000040
ffffe00001ca1c20
[ 007] ffffe00002019380 tmo3 0000000000000000 0000000000989680
ffffe00001ca1c20
[ 008] ffffe000020193a0 tmo2 0000000000000000 00000000000003e8
ffffe00001ca1c20
[ 009] ffffe000020193c0 tmo1 0000000000000000 000000000002625a
ffffe00001ca1c20
[ 010] ffffe000020193e0 tmo0 00000000000003e8 ffffe00001ca1b88
ffffe00001ca1c20
[ 011] ffffe00002019400 hci0 0000000000000000 0000000000000000
0000000000000000
[ 012] ffffe00002019420 xSt0 ffffe00001ca1b88 0000000000000008
0000000000000003
[ 013] ffffe00002019440 xdw4 ffffe00001ca1b88 0000000000000000
0000000000000000
[ 014] ffffe00002019460 xdw2 ffffe00001ca1b88 ffffe00001ca1050
0000000000000002
[ 015] ffffe00002019480 xdB0 ffffe00001ca1b88 ffffe00001ca1050
0000000000000000
```

```
//
// USBPORT_Xdpc_Worker_HcIntDpc()  DPC/XDPC: 2 of 4

[ 016] ffffe000020194a0 iDP- 0000000000000000 0000000000b73e26
0000000000000000
//
// USBPORT_IsrDpc() - Exit()

[ 017] ffffe000020194c0 xSt0 ffffe00001ca1b88 0000000000000007
0000000000000002
[ 018] ffffe000020194e0 Xsi1 ffffe00001ca1b88 0000000000000000
0000000000000000
//
// USBPORT_Xdpc_iSignal()- USBPORT_Core_UsbHcIntDpc_Worker() DPC/XDPC: 2 of
4

[ 019] ffffe00002019500 chgZ 0000000000000000 0000000000b73e26
0000000000000000
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhcdpnp

Статья • 05.04.2024

Команда **!usbkd.usbhcdpnp** отображает журнал состояний самонастраивающийся (PnP) для контроллера USB-узла или корневого концентратора.

```
dbgcmd
```

```
!usbkd.usbhcdpnp DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес одного из следующих элементов:

- Расширение устройства для функционального объекта устройства (FDO) контроллера USB-узла.
- Расширение устройства для объекта физического устройства (PDO) корневого концентратора USB.

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен один из способов найти адрес расширения устройства для FDO контроллера USB-узла. Сначала введите **!usbkd.usb2tree**.

```
dbgcmd
```

```
0: kd> !usbkd.usb2tree

UHCI MINIPORT(s) dt usbport!_USBPORT_MINIPORT_DRIVER ffffe0000090c3d0
...
4)!uhci_info ffffe00001c8f1a0 !devobj ffffe00001c8f050 PCI: VendorId 8086
DeviceId 2938 RevisionId 0002
...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !uhci_info ffffe00001c8f1a0.

Теперь передайте адрес расширения **устройства в команду !usbhcdpnp** .

dbgcmd

```
0: kd> !usbkd.usbhcdpnp ffffe00001c8f1a0

## PNP STATE LOG (latest at bottom)

##      EVENT                          STATE               NEXT

[01] EvFDO_IRP_MN_START_DEVICE      PnpNotStarted       PnpStarted
[02] EvFDO_IRP_MN_QBR_RH            PnpStarted          PnpStarted
```

Ниже приведен один из способов найти адрес расширения устройства для PDO корневого концентратора. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
4)!uhci_info ffffe00001c8f1a0 !devobj ffffe00001c8f050 PCI: VendorId 8086
DeviceId 2938 RevisionId 0002
    RootHub !hub2_info ffffe00000d941a0 !devstack ffffe00000d94050
```

В предыдущих выходных данных можно увидеть адрес FDO корневого концентратора, отображаемый в качестве аргумента команды **!devstack ffffe00000094050**. **Используйте команду !devstack**, чтобы найти адрес PDO и расширения устройства PDO.

dbgcmd

```
0: kd> !kdexts.devstack ffffe00000d94050
  !DevObj           !DrvObj            !DevExt           ObjectName
> ffffe00000d94050  \Driver\usbhub     ffffe00000d941a0  0000006b
  ffffe00000ed4050  \Driver\usbuhci    ffffe00000ed41a0  USBPDO-2
```

В предыдущих выходных данных можно увидеть, что адрес расширения устройства для PDO корневого концентратора `ffffe00000ed41a0`.

Теперь передайте адрес расширения **устройства в команду !usbhcdpnp** .

dbgcmd

```
0: kd> !usbkd.usbhcdpnp ffffe00000ed41a0

## PNP STATE LOG (latest at bottom)

##      EVENT                          STATE               NEXT
```

```
[01] EvPDO_IRP_MN_START_DEVICE      PnpNotStarted       PnpStarted
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhcdpow

Статья • 05.04.2024

Команда **!usbkd.usbhcdpow** отображает журнал состояний питания для контроллера USB-узла или корневого концентратора.

```
dbgcmd
```

```
!usbkd.usbhcdpow DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес одного из следующих элементов:

- Расширение устройства для функционального объекта устройства (FDO) контроллера USB-узла.
- Расширение устройства для объекта физического устройства (PDO) корневого концентратора USB.

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен один из способов найти адрес расширения устройства для FDO узла EHCI. Сначала введите **!usbkd.usb2tree**.

```
dbgcmd
```

```
0: kd> !usbkd.usb2tree
...

2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
     ...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !ehci_info ffffe00001ca11a0.

Теперь передайте адрес расширения устройства команде **!usbhcdpow** .

dbgcmd

```
0: kd> !usbkd.usbhcdpow ffffe00001ca11a0

dt USBPORT!_FDO_EXTENSION ffffe00001ca15a0

## State History (latest at bottom)

##      EVENT                              STATE
NEXT

[00] FdoPwrEv_D0_DoSetD0_2              FdoPwr_D0_WaitWorker2
FdoPwr_D0_WaitSyncUsb2                dt:0 ms
[01] FdoPwrEv_SyncUsb2_DoChirp          FdoPwr_D0_WaitSyncUsb2
FdoPwr_D0_WaitSyncUsb2                dt:0 ms
[02] FdoPwrEv_Rh_SetPowerSys            FdoPwr_D0_WaitSyncUsb2
FdoPwr_D0_WaitSyncUsb2                dt:0 ms
[03] FdoPwrEv_Rh_SetD0                  FdoPwr_D0_WaitSyncUsb2
FdoPwr_D0_WaitSyncUsb2                dt:0 ms
[04] FdoPwrEv_SyncUsb2_Complete         FdoPwr_D0_WaitSyncUsb2
FdoPwr_WaitSx                         dt:50 ms
[05] FdoPwrEv_Rh_Wake                   FdoPwr_WaitSx
FdoPwr_WaitSx                         dt:3412 ms
[06] FdoPwrEv_Rh_Wake                   FdoPwr_WaitSx
FdoPwr_WaitSx                         dt:283872 ms
[07] FdoPwrEv_Rh_Wake                   FdoPwr_WaitSx
FdoPwr_WaitSx                         dt:25481267 ms
```

Ниже приведен один из способов найти адрес расширения устройства для PDO корневого концентратора. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...

2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
        ...
```

В предыдущих выходных данных можно увидеть адрес FDO корневого концентратора, отображаемый в качестве аргумента команды **!devstack ffffe00002320050**. **Используйте команду !devstack**, чтобы найти адрес PDO и расширения устройства PDO.

dbgcmd

```
0: kd> !kdexts.devstack ffffe00002320050
  !DevObj           !DrvObj            !DevExt           ObjectName
> ffffe00002320050  \Driver\usbhub     ffffe000023201a0  0000002d
  ffffe0000213c050  \Driver\usbehci    ffffe0000213c1a0  USBPDO-3
...
```

В предыдущих выходных данных можно увидеть, что адрес расширения устройства для PDO корневого концентратора `ffffe0000213c1a0`.

Теперь передайте адрес расширения устройства команде **!usbhcdpow** .

dbgcmd

```
0: kd> !usbkd.usbhcdpow ffffe0000213c1a0

dt USBPORT!_FDO_EXTENSION ffffe0000213c5a0

## State History (latest at bottom)

##      EVENT                              STATE
NEXT

...
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.hub2_info_from_fdo

Статья • 05.04.2024

Команда **!usbkd.hub2_info_from_fdo** отображает сведения о USB-концентраторе.

dbgcmd

```
!usbkd.hub2_info_from_fdo FDO
```

## Параметры

*FDO*
Адрес объекта функционального устройства (FDO) для USB-концентратора.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов найти адрес FDO для USB-концентратора. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
```

В предыдущих выходных данных адрес FDO для концентратора отображается в качестве аргумента предлагаемой команды **!devstack ffffe00002320050**.

Теперь передайте адрес FDO **в команду !hub2_info_from_fdo** .

dbgcmd

```
0: kd> !usbkd.hub2_info_from_fdo ffffe00002320050
usbhubext
**************************************************************************
*
```

```
FDO ffffe00002320050 PDO ffffe0000213c050 HubNumber# 3
dt USBHUB!_DEVICE_EXTENSION_HUB ffffe000023201a0
!usbhublog ffffe000023201a0
RemoveLock ffffe00002320668
FdoFlags ffffe00002320ba0

CurrentPowerIrp: System (0000000000000000) Device (0000000000000000)

ObjReferenceList: !usblist ffffe00002320b70, RL
ExceptionList: !usblist ffffe00002321498, EL [Empty]
DmTimerListHead: !usblist ffffe00002321040, TL [Empty]
PdoRemovedListHead: !usblist ffffe00002321478, PL [Empty]
PdoPresentListHead: !usblist ffffe00002321468, PL
WorkItemListHead: !usblist ffffe00002320c80, WI [Empty]
SshBusyListHead: !usblist ffffe00002320dc0, BL


## PnP FUNC HISTORY (latest at bottom)

01. IRP_MN_QUERY_DEVICE_RELATIONS
...

## POWER FUNC HISTORY (latest at bottom)

01. IRP_MN_QUERY_POWER    - PowerSystemHibernate
...

## HARD RESET STATE HISTORY (latest at bottom)

##     EVENT                           STATE
NEXT

01. HRE_Pause                       HReset_WaitReady
HReset_Paused
...

## PNP STATE HISTORY (latest at bottom)

##     EVENT                           STATE
NEXT

01. Ev_SYSTEM_POWER                 FDO_WaitPnpStop
FDO_WaitPnpStop
...

## POWER STATE HISTORY (latest at bottom)

##     EVENT                           STATE
NEXT

01. Ev_SET_POWER_S0                 FdoSx_Dx
FdoWaitS0IoComplete_Dx
...
```

```
## BUS STATE HISTORY (latest at bottom)

##     EVENT                            STATE
NEXT

01. BE_BusSuspend                   BS_BusPause
BS_BusSuspend
...

SSH_EnabledStatus: [SSH_ENABLED_VIA_POWER_POLICY]

## SSH STATE HISTORY (latest at bottom)

##     EVENT                            STATE
NEXT

01. SSH_Event_ResumeHubComplete     SSH_State_HubPendingResume
SSH_State_HubActive
...

## PORT DATA

PortData 1: !port2_info ffffe000021bf000 Port State = PS_WAIT_CONNECT
PortChangeLock: 0, Pcq_State: Pcq_Run_Idle
     PDO 0000000000000000
...
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhuberr

Статья • 05.04.2024

Команда **!usbkd.usbhuberr** отображает запись об ошибке USB-концентратора.

```
dbgcmd

!usbkd.usbhuberr StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbhub!_HUB_EXCEPTION_RECORD**.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов найти адрес **usbhub!_HUB_EXCEPTION_RECORD**. Сначала введите **!usbkd.usb2tree**.

```
dbgcmd

0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
      ...
```

В предыдущих выходных данных можно увидеть предложенную команду **!devstack ffffe00002320050**. Введите эту команду.

```
dbgcmd

0: kd> !kdexts.devstack ffffe000011f7050

  !DevObj           !DrvObj            !DevExt           ObjectName
> ffffe000011f7050  \Driver\usbhub     ffffe000011f71a0  0000006f
```

```
  ffffe00000a21050  \Driver\usbehci    ffffe00000a211a0  USBPDO-8
...
```

В предыдущих выходных данных `ffffe000011f71a0` — это адрес расширения устройства для функционального объекта устройства (FDO) концентратора. Передайте адрес расширения устройства в !usbkd.usbhubext.

dbgcmd

```
0: kd> !usbkd.usbhubext ffffe000011f71a0

FDO ffffe000011f7050 PDO ffffe00000a21050 HubNumber# 7
dt USBHUB!_DEVICE_EXTENSION_HUB ffffe000011f71a0
!usbhublog ffffe000011f71a0
RemoveLock ffffe000011f7668
FdoFlags ffffe000011f7ba0

CurrentPowerIrp: System (0000000000000000) Device (0000000000000000)

ObjReferenceList: !usblist ffffe000011f7b70, RL
ExceptionList: !usblist ffffe000011f8498, EL [Empty]
...
```

В предыдущих выходных данных `ffffe000011f8498` — это адрес списка исключений. Если список исключений не пуст, он будет содержать адреса _HUB_EXCEPTION_RECORD структур.

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhubext

Статья • 05.04.2024

Команда **!usbkd.usbhubext** отображает сведения о USB-концентраторе.

```
dbgcmd

!usbkd.usbhubext DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес одного из следующих элементов:

- Расширение устройства для функционального объекта устройства (FDO) USB-концентратора.
- Расширение устройства для объекта физического устройства (PDO) устройства, подключенного к USB-концентратору.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов найти адрес расширения устройства для FDO USB-концентратора. Сначала введите **!usbkd.usb2tree**.

```
dbgcmd

0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
```

В предыдущих выходных данных можно увидеть предложенную команду **!devstack ffffe00002320050**. Введите эту команду.

```
dbgcmd
```

```
0: kd> !kdexts.devstack ffffe00002320050

  !DevObj           !DrvObj            !DevExt           ObjectName
> ffffe00002320050  \Driver\usbhub     ffffe000023201a0  0000002d
  ffffe0000213c050  \Driver\usbehci    ffffe0000213c1a0  USBPDO-3
...
```

В предыдущих выходных данных можно увидеть, что адрес расширения устройства для FDO концентратора `ffffe000023201a0`.

Теперь передайте адрес расширения устройства команде **!usbkd.usbhubext** .

dbgcmd

```
0: kd> !usbkd.usbhubext ffffe000023201a0

FDO ffffe00002320050 PDO ffffe0000213c050 HubNumber# 3
dt USBHUB!_DEVICE_EXTENSION_HUB ffffe000023201a0
!usbhublog ffffe000023201a0
RemoveLock ffffe00002320668
FdoFlags ffffe00002320ba0

CurrentPowerIrp: System (0000000000000000) Device (0000000000000000)

ObjReferenceList: !usblist ffffe00002320b70, RL
ExceptionList: !usblist ffffe00002321498, EL [Empty]
DmTimerListHead: !usblist ffffe00002321040, TL [Empty]
PdoRemovedListHead: !usblist ffffe00002321478, PL [Empty]
PdoPresentListHead: !usblist ffffe00002321468, PL
WorkItemListHead: !usblist ffffe00002320c80, WI [Empty]
SshBusyListHead: !usblist ffffe00002320dc0, BL


## PnP FUNC HISTORY (latest at bottom)

01. IRP_MN_QUERY_DEVICE_RELATIONS
...
## POWER FUNC HISTORY (latest at bottom)

01. IRP_MN_QUERY_POWER    - PowerSystemHibernate
...

## HARD RESET STATE HISTORY (latest at bottom)

##     EVENT                           STATE
NEXT

01. HRE_Pause                       HReset_WaitReady
HReset_Paused
...

## PNP STATE HISTORY (latest at bottom)
```

```
##     EVENT                           STATE
NEXT

01. Ev_SYSTEM_POWER                 FDO_WaitPnpStop
FDO_WaitPnpStop
...

## POWER STATE HISTORY (latest at bottom)

##     EVENT                           STATE
NEXT

01. Ev_SET_POWER_S0                 FdoSx_Dx
FdoWaitS0IoComplete_Dx
...

## BUS STATE HISTORY (latest at bottom)

##     EVENT                           STATE
NEXT

01. BE_BusSuspend                   BS_BusPause
BS_BusSuspend
...

SSH_EnabledStatus: [SSH_ENABLED_VIA_POWER_POLICY]

## SSH STATE HISTORY (latest at bottom)

##     EVENT                           STATE
NEXT

01. SSH_Event_ResumeHubComplete     SSH_State_HubPendingResume
SSH_State_HubActive
...

## PORT DATA

PortData 1: !port2_info ffffe000021bf000 Port State = PS_WAIT_CONNECT
PortChangeLock: 0, Pcq_State: Pcq_Run_Idle
     PDO 0000000000000000
...
```

Ниже приведен один из способов найти адрес расширения устройства для PDO устройства, подключенного к USB-концентратору. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
```

```
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
        Port 1: !port2_info ffffe000021bf000
        Port 2: !port2_info ffffe000021bfb40
        Port 3: !port2_info ffffe000021c0680 !devstack ffffe00007c882a0
```

В предыдущих выходных данных можно увидеть предложенную команду **!devstack ffffe00007c882a0**. Введите эту команду.

dbgcmd

```
0: kd> !kdexts.devstack ffffe00007c882a0

  !DevObj           !DrvObj            !DevExt           ObjectName
  ffffe00006ce2260  \Driver\USBSTOR    ffffe00006ce23b0  00000070
> ffffe00007c882a0  \Driver\usbhub     ffffe00007c883f0  USBPDO-4
...
```

В предыдущих выходных данных видно, что адрес расширения устройства для PDO устройства . `ffffe00007c883f0`

Теперь передайте адрес расширения устройства в команду !usbhcdpnp.

dbgcmd

```
0: kd> !usbkd.usbhubext ffffe00007c883f0

dt USBHUB!_DEVICE_EXTENSION_PDO ffffe00007c883f0
PARENT HUB: FDO ffffe00002320050 !hub2_info ffffe000023201a0
!usbhubinfo ffffe00002320050
PORT NUMBER : 3
IoList: !usblist ffffe00007c888b0, IO
LatchList: !usblist ffffe00007c888e0, LA

## PnP ID's

DeviceId:USB\VID_0781&PID_5530
HardwareId:USB\VID_0781&PID_5530&REV_0100USB\VID_0781&PID_5530
CompatibleId:USB\Class_08&SubClass_06&Prot_50USB\Class_08&SubClass_06USB\Cla
ss_08
SerialNumberId:20052444100A47F319CB
UniqueId:3
ProductId:Cruzer

## Pnp Func History (latest at bottom)

01. IRP_MN_QUERY_BUS_INFORMATION
...

## Power Func History (latest at bottom)
```

```
## PNP STATE HISTORY (latest at bottom)

##     EVENT                              STATE
NEXT

01. (6)                              (0)
PDO_WaitPnpStart
02. Ev_PDO_IRP_MN_START              PDO_WaitPnpStart
PDO_WaitPnpStop

## POWER STATE HISTORY (latest at bottom)

##     EVENT                              STATE
NEXT

    [EMPTY]

## HARDWARE STATE HISTORY (latest at bottom)

##     EVENT                              STATE
NEXT

01. PdoEv_CreatePdo                  (0)
Pdo_Created
02. PdoEv_RegisterPdo                Pdo_Created
Pdo_HwPresent
03. PdoEv_QBR                        Pdo_HwPresent
Pdo_PnpRefHwPresent

## IDLE STATE HISTORY (latest at bottom)

##     EVENT                              STATE
NEXT

    [EMPTY]
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhubinfo

Статья • 05.04.2024

Команда **!usbkd.hubinfo** отображает сведения о USB-концентраторе.

```
dbgcmd

!usbkd.hubinfo FDO
```

## Параметры

*FDO*
Адрес объекта функционального устройства (FDO) для USB-концентратора.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов найти адрес FDO для USB-концентратора. Сначала введите **!usbkd.usb2tree**.

```
dbgcmd

0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
```

В предыдущих выходных данных адрес FDO для концентратора отображается в качестве аргумента предлагаемой команды **!devstack ffffe00002320050**.

Теперь передайте адрес FDO в **команду !usbhubinfo** .

```
dbgcmd

0: kd> !usbkd.usbhubinfo ffffe00002320050

    !DevObj ffffe00002320050 !usbhubext ffffe000023201a0
On Host Controller (0x8086, 0x293c)
```

```
        Stat_AsyncResumeStartAt: 2437ee39d29bd528
        Stat_AsyncResumeCompleteAt: 24413c77d29bd528
        Stat_AsyncResume: 0x3c(60) ms
        Stat_SyncResumeStartAt: 2437ee39d29bd528
        Stat_SyncResumeCompleteAt: 2437ee39d29bd528
        Stat_SyncResume: 0x0(0) ms
Trap Regs: Event, Port, Event (ffffe000023204d0)
    Enable: 0 Port: 0 Event 00000000
Hub Number: # 3
Number Of Ports: 4
dt usbhub!_USBHUB_FDO_FLAGS ffffe00002320ba0
>Is Root
>Power Switching:
     No Power Switching
>Overcurrent:
     Global Overcurrent
>PortIndicators:
     No PortIndicators present
>AllowWakeOnConnect:
     DO NOT WakeOnConnect
>CURRENT Hub Wake on Connectstate:
     HWC_DISARM:- do not wake system on connect/disconnect event
>CURRENT Bus Wake state:
     BUS_DISARM:- bus not armed for wake by this hub
>CURRENT Wake Detect state (WW Irp):
     HUB_DISARM:- no ww irp pending (HUB_WAKESTATE_DISARMED)
Milliamps/Port : 500ma
Power caps (0 = not reported)
     PortPower_Registry : 0
     PortPower_DeviceStatus : 500
     PortPower_CfgDescriptor : 500
     PortPower_HubStatus : 500
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhublog

Статья • 05.04.2024

Команда **!usbkd.usbhublog** отображает журнал отладки для USB-концентратора.

dbgcmd

```
!usbkd.usbhublog DeviceExtension[, NumberOfEntries]
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства (FDO) USB-концентратора.

*NumberOfEntries*
Число отображаемых записей журнала. Чтобы отобразить весь журнал, задайте для этого параметра значение -1.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов найти адрес расширения устройства для FDO USB-концентратора. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
      ...
```

В предыдущих выходных данных можно увидеть предложенную команду **!devstack ffffe00002320050**. Введите эту команду.

dbgcmd

```
0: kd> !kdexts.devstack ffffe00002320050

  !DevObj           !DrvObj            !DevExt           ObjectName
> ffffe00002320050  \Driver\usbhub     ffffe000023201a0  0000002d
  ffffe0000213c050  \Driver\usbehci    ffffe0000213c1a0  USBPDO-3
...
```

В предыдущих выходных данных `ffffe000023201a0` — это адрес расширения устройства для FDO концентратора.

Теперь передайте адрес расширения **устройства в !usbhublog**. В этом примере второй аргумент ограничивает отображение 10 записей журнала.

dbgcmd

```
0: kd> !usbkd.usbhublog ffffe000023201a0, 10

LOG@: ffffe000023201a0 (usbhub!_DEVICE_EXTENSION_HUB)
>LOG mask = ff idx = ffffa333 (33)
*LOG: ffffe00002321ca0  LOGSTART: ffffe00002321640 *LOGEND: ffffe00002323620
# 20
[ 000] ffffe00002321ca0 HDec 0000000000000000 ffffe000002904d0
0000000000000001
[ 001] ffffe00002321cc0 HPCd 0000000000000000 0000000000000002
0000000000000004
[ 002] ffffe00002321ce0 qwk- 0000000000000000 ffffe000021c11c0
0000000000000000
[ 003] ffffe00002321d00 pq-- 0000000000000000 0000000000000002
0000000000000004
[ 004] ffffe00002321d20 _6p4 0000000000000000 0000000000000000
0000000000000004
[ 005] ffffe00002321d40 _6p1 0000000000000000 0000000000000003
0000000000000004
[ 006] ffffe00002321d60 pq++ 0000000000000000 0000000000000003
0000000000000004
[ 007] ffffe00002321d80 pq++ 0000000000000000 0000000000000006
0000000000000004
[ 008] ffffe00002321da0 _6p0 0000000000000000 ffffe000021c11c0
0000000000000004
[ 009] ffffe00002321dc0 pqDP 0000000000000000 ffffe000021c11d8
0000000000000006
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhubmddevext

Статья • 05.04.2024

Команда !usbkd.usbhubmdevext отображает структуру usbhub!_DEVICE_EXTENSION_HUB, **если она присутствует в аварийном дампе, созданном в результате проверки ошибок 0xFE.**

dbgcmd

```
!usbkd.usbhubmddevext
```

## DLL-библиотеки

Usbkd.dll

## Замечания

Используйте эту команду только при отладке файла аварийного дампа, созданного **в результате проверки ошибок 0xFE: BUGCODE_USB_DRIVER**.

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhubmdpdd

Статья • 05.04.2024

Команда !usbkd.usbhubmdpdd отображает структуру usbhub!_HUB_PORT_DATA, **если она присутствует в аварийном дампе, созданном в результате проверки ошибок 0xFE**.

```
dbgcmd
```

```
!usbkd.usbhubmdpd [PortNum]
```

## Параметры

*PortNum*
Номер USB-порта. Если указать номер порта, эта команда отображает структуру (если она присутствует), представляющую указанный порт. Если номер порта не указан, эта команда отображает структуру (если она присутствует), в которой **была инициирована 0xFE** проверка ошибок.

## DLL-библиотеки

Usbkd.dll

## Замечания

Используйте эту команду только при отладке файла аварийного дампа, созданного **в результате проверки ошибок 0xFE: BUGCODE_USB_DRIVER**.

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?**

Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhubpd

Статья • 05.04.2024

Команда **!usbkd.usbhubpd** отображает сведения о USB-порту.

dbgcmd

```
!usbkd.usbhubpd StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbhub!_HUB_PORT_DATA**. Чтобы получить адреса этих структур, используйте **!usbhubext**.

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов **найти адрес usbhub!_HUB_PORT_DATA**. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
    RootHub !hub2_info ffffe000023201a0 !devstack ffffe00002320050
        ...
```

В предыдущих выходных данных можно увидеть предложенную команду **!devstack ffffe00002320050**. Введите эту команду.

dbgcmd

```
0: kd> !kdexts.devstack ffffe00002320050

  !DevObj           !DrvObj            !DevExt           ObjectName
> ffffe00002320050  \Driver\usbhub     ffffe000023201a0  0000002d
```

```
  ffffe0000213c050  \Driver\usbehci    ffffe0000213c1a0  USBPDO-3
...
```

В предыдущих выходных данных можно увидеть, что адрес расширения устройства для FDO концентратора `ffffe000023201a0`.

Передайте адрес расширения устройства команде **!usbhubext** .

dbgcmd

```
0: kd> !usbkd.usbhubext ffffe000023201a0

FDO ffffe00002320050 PDO ffffe0000213c050 HubNumber# 3
dt USBHUB!_DEVICE_EXTENSION_HUB ffffe000023201a0
!usbhublog ffffe000023201a0
RemoveLock ffffe00002320668
FdoFlags ffffe00002320ba0

CurrentPowerIrp: System (0000000000000000) Device (0000000000000000)
...
## PORT DATA

PortData 1: !port2_info ffffe000021bf000 Port State = PS_WAIT_CONNECT
PortChangeLock: 0, Pcq_State: Pcq_Run_Idle
     PDO 0000000000000000
....
```

В предыдущих выходных **данных** `ffffe000021bf000` **используется адрес структуры _HUB_PORT_DATA**. Передайте этот адрес **в !usbhubpd**.

dbgcmd

```
0: kd> !usbkd.usbhubpd ffffe000021bf000
PortNumber: 1
Parent Hub FDO ffffe00002320050
Device PDO <NULL>
dt USBHUB!_HUB_PORT_DATA ffffe000021bf000
dt USBHUB!_PORTDATA_FLAGS ffffe000021bf968

PortChangelist: !usblist ffffe000021bf1c8, CL [Empty]

## Port Indicators Log (latest at bottom)

##     Event           State                Next

    [EMPTY]

## Port Change Queue History (latest at bottom)

##     Event                State                    Next
PcqEv_Suspend PcqEv_Resume  PcqEv_ChDone  Tag
```

```
01. PCE_Resume           Pcq_Stop                Pcq_Pause
PcqEv_Reset   PcqEv_Reset   REQUEST_RESUME
...          Pcq_Run_wBusy            Pcq_Run_Idle

## Port Status History (latest at bottom)

##     Current State          Change Eve nt        PDO              CEOSP
H/W Port REG Frame Inserted

01. PS_WAIT_CONNECT        REQUEST_PAUSE       0000000000000000 00000 100
Age:000 512498
...
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbhubs

Статья • 05.04.2024

Команда **!usbkd.usbhubs** отображает сведения о USB-концентраторах.

dbgcmd

```
!usbkd.usbhubs a[v]
!usbkd.usbhubs x[v]
!usbkd.usbhubs r[v]
```

## Параметры

**a**
Отображение всех центров.

**r**
Отображение корневых концентраторов.

**x**
Отображение внешних концентраторов.

**v**
Выходные данные подробные. Например, **!usbhubs rv** отображает подробные выходные данные обо всех корневых концентраторах.

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен пример подробных выходных данных из **команды !usbhubs** .

dbgcmd

```
0: kd> !usbkd.usbhubs rv

args 'rv'
level: r v
ROOT HUBS
*LIST -- Root Hub List @ fffff8000217f440
```

```
*flink, blink ffffe000023215c0,ffffe000011f85c0
----------

[0] entry @: ffffe000023201a0
    !DevObj ffffe00002320050 !usbhubext ffffe000023201a0
On Host Controller (0x8086, 0x293c)
        Stat_AsyncResumeStartAt: 2437ee39d29bd498
        Stat_AsyncResumeCompleteAt: 24413c77d29bd498
        Stat_AsyncResume: 0x3c(60) ms
        Stat_SyncResumeStartAt: 2437ee39d29bd498
        Stat_SyncResumeCompleteAt: 2437ee39d29bd498
        Stat_SyncResume: 0x0(0) ms
Trap Regs: Event, Port, Event (ffffe000023204d0)
    Enable: 0 Port: 0 Event 00000000
Hub Number: # 3
Number Of Ports: 4
dt usbhub!_USBHUB_FDO_FLAGS ffffe00002320ba0
>Is Root
>Power Switching:
     No Power Switching
>Overcurrent:
     Global Overcurrent
>PortIndicators:
     No PortIndicators present
>AllowWakeOnConnect:
     DO NOT WakeOnConnect
>CURRENT Hub Wake on Connectstate:
     HWC_DISARM:- do not wake system on connect/disconnect event
>CURRENT Bus Wake state:
     BUS_DISARM:- bus not armed for wake by this hub
>CURRENT Wake Detect state (WW Irp):
     HUB_DISARM:- no ww irp pending (HUB_WAKESTATE_DISARMED)
Milliamps/Port : 500ma
Power caps (0 = not reported)
     PortPower_Registry : 0
     PortPower_DeviceStatus : 500
     PortPower_CfgDescriptor : 500
##      PortPower_HubStatus : 500

----------

[1] entry @: ffffe000008b91a0
    !DevObj ffffe000008b9050 !usbhubext ffffe000008b91a0
On Host Controller (0x8086, 0x2937)
...
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usblist

Статья • 05.04.2024

Команда **!usbkd.usblist** отображает связанный список структур указанного типа.

```
dbgcmd
```

```
!usbkd.usblist ListAddr, ListType
```

## Параметры

*ListAddr*
Адрес связанного списка структур. Чтобы найти адреса связанных списков, поддерживаемых драйвером USB-порта, используйте **!usbhcdext**. Чтобы найти адреса связанного списка, поддерживаемого драйвером USB-концентратора, используйте **!usbhubext**.

*ListType*
Один из следующих типов списка.

Развернуть таблицу

| Тип списка | Структура |
|---|---|
| BC | usbport!_BUS_CONTEXT |
| EP | usbport!_HCD_ENDPOINT |
| TT | usbport!_TRANSACTION_TRANSLATOR |
| DL | usbport!_USBD_DEVICE_HANDLE |
| PL | usbhub!_DEVICE_EXTENSION_PDO |
| ЭЛЬ | usbhub!_HUB_EXCEPTION_RECORD |
| RL | usbhub!_HUB_REFERENCE_LIST_ENTRY |
| TL | usbhub!_HUB_TIMER_OBJECT |
| WI | usbhub!_HUB_WORKITEM |
| IO | usbhub!_IO_LIST_ENTRY |
| LA | usbhub!_LATCH_LIST_ENTRY |

| Тип списка | Структура |
| --- | --- |
| CL | usbhub!_PORT_CHANGE_CONTEXT |
| BL | usbhub!_SSP_BUSY_HANDLE |

# DLL-библиотеки

Usbkd.dll

# Примеры

Вот один из способов найти адрес связанного списка. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 ...
   ...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !ehci_info ffffe00001ca11a0.

Щелкните команду DML или передайте адрес расширения устройства в !usbhcdext.

dbgcmd

```
0: kd> !usbkd.usbhcdext ffffe00001ca11a0

HC Flavor 1000  FDO ffffe00001ca1050
Root Hub: FDO ffffe00002320050 !hub2_info ffffe000023201a0
...
DeviceHandleList: !usblist ffffe00001ca23b8, DL
...
```

В предыдущем выходных данных ffffe00001ca23b8 — это адрес связанного **списка структур usbport!_USBD_DEVICE_HANDLE** .

Теперь передайте адрес связанного списка **в !usblist**.

dbgcmd

```
0: kd> !usblist ffffe00001ca23b8, DL
list: ffffe00001ca23b8 DL
----------
!usbdevh ffffe000020f9590
SSP [IdleReady] (0)
PCI\VEN_Xxxx  Xxxx Corporation
Root Hub
DriverName :
----------
!usbdevh ffffe00001bce250
SSP [IdleReady] (0)
USB\Xxxx  Xxxx Corporation
Speed: HIGH, Address:  1, PortPathDepth: 1, PortPath: [3 0 0 0 0 0]
DriverName :\Driver\USBSTOR      !devstack ffffe000053ef2a0
----------
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbpo

Статья • 05.04.2024

Команда **!usbkd.usbpo** отображает внутренний список невыполненных запросов на питание USB.

dbgcmd

```
!usbkd.usbpo
```

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbpdos

Статья • 05.04.2024

Команда **!usbkd.usbpdos** отображает сведения обо всех физических объектах устройств (PDOS), созданных драйвером USB-концентратора.

dbgcmd

```
!usbkd.usbpdos
```

## DLL-библиотеки

Usbkd.dll

## Примеры

Ниже приведен пример выходных **данных команды !usbpdos** .

dbgcmd

```
0: kd> !usbkd.usbpdos

ext ffffe00006c513f0
VID 0a12 PID 0001 REV 0309
 dt USBHUB!_USB_DEVICE_DESCRIPTOR ffffe00006c51960
Vendor: Xxxx
 Xxxx Corporation
 dd Class: (224)(e0)
 (224)(e0)
HubFdo: ffffe00000d94050, port: 2
<null>
SystemWake 5 SystemHibernate(S4)
wake_irp_list_head = ffffe00006c51cb0
wake_irp = ffffe00006c51cb0
PDO Times:
    Stat_PdoCreatedAt: 0d29bbd58
    Stat_PdoEnumeratedAt: b65ace75d29bbd58
    Stat: Enumeration Time: 0xa7d3842a(-1479310294) ms
    Stat_Pdo_SetD0_StartAt: 0d29bbd58
    Stat_Pdo_SetD0_CompleteAt: 0d29bbd58
    Stat: PDO Set_D0 time: 0x0(0) ms

##

ext ffffe00007c883f0
VID 0781 PID 5530 REV 0100
```

```
 dt USBHUB!_USB_DEVICE_DESCRIPTOR ffffe00007c88960
Vendor: Xxxx
 Xxxx Corporation
 dd Class: (0)(0)
 (8)Class_UsbStorage
HubFdo: ffffe00002320050, port: 3
:Xxxx
SystemWake 5 SystemHibernate(S4)
wake_irp_list_head = ffffe00007c88cb0
wake_irp = ffffe00007c88cb0
PDO Times:
    Stat_PdoCreatedAt: 0d29bbd58
    Stat_PdoEnumeratedAt: 2380af52d29bbd58
    Stat: Enumeration Time: 0x9a24226c(-1708907924) ms
    Stat_Pdo_SetD0_StartAt: 0d29bbd58
    Stat_Pdo_SetD0_CompleteAt: 0d29bbd58
##     Stat: PDO Set_D0 time: 0x0(0) ms
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbpdoxls

Статья • 05.04.2024

Команда **!usbkd.usbpdoxls** отображает сведения обо всех физических объектах устройств (PDOS), созданных драйвером USB-концентратора. Для каждого PDO эта команда отображает строку разделенных запятыми значений.

dbgcmd

```
!usbkd.usbpdoxls
```

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbpnp

Статья • 05.04.2024

Команда **!usbkd.usbpnp** отображает сведения о контексте состояния о USB-концентраторе.

```
dbgcmd
```

```
!usbkd.usbpnp DeviceExtension
```

## Параметры

*DeviceExtension*
Адрес расширения устройства для функционального объекта устройства (FDO) USB-концентратора.

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbportisasyncadv

Статья • 05.04.2024

**Команда !usbkd.usbportisasyncadv** проверка всех драйверов минипорта EHCI для прерывания EHCI в проблеме Async Advance.

```
dbgcmd
```

```
!usbkd.usbportisasyncadv
```

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbportmdportlog

Статья • 05.04.2024

Команда **!usbkd.usbportmdportlog** отображает журнал отладки USBPORT, если он присутствует в аварийном дампе, созданном в результате **проверки ошибок 0xFE**.

dbgcmd

```
!usbkd.usbportmdportlog
```

## DLL-библиотеки

Usbkd.dll

## Замечания

Используйте эту команду только при отладке файла аварийного дампа, созданного **в результате проверки ошибок 0xFE: BUGCODE_USB_DRIVER**.

## Примеры

Ниже приведен пример части выходных данных **!usbportmdportlog**.

dbgcmd

```
1: kd> !analyze -v
*** ...
BUGCODE_USB_DRIVER (fe)
...
1: kd> !usbkd.usbportmdportlog
Minidump USBPORT DEBUG_LOG buffer size 32768, entries 1024, index 400
*LOG: 0000000113be9600  LOGSTART: 0000000113be6400 *LOGEND: 0000000113bee3e0
# 1024
[ 000] 0000000113be9600 Bfe2 ffffe0001416802c ffffe000020a44f0
ffffe0001416801c
[ 001] 0000000113be9620 Bfe0 0000000000000000 ffffe000039f4720
ffffe00000b76cb0
[ 002] 0000000113be9640 epr+ ffffe000043ee010 ffffe000008f5b80
ffffe00002820a0c
[ 003] 0000000113be9660 alTR ffffe00002820a0c ffffe000039f4720
ffffe00000b76cb0
//
// USBPORT_Core_AllocTransferEx()
// transfer:                ffffe00002820a0c
```

```
// Urb:                     ffffe000039f4720
// Irp:                     ffffe00000b76cb0

[ 004] 0000000113be9680 TRcs 0000000000000060 0000000000000468
0000000000000001
[ 005] 0000000113be96a0 urbi ffffe0001422c4cc 0000000000000012
000000000000000b
[ 006] 0000000113be96c0 dURB ffffe000039f4720 ffffe00000b76cb0
000000000000000b
//
// USBPORT_ProcessURB() - Device Handle valid and has been referenced
// Urb:                     ffffe000039f4720
// Irp:                     ffffe00000b76cb0
// function:                URB_FUNCTION_GET_DESCRIPTOR_FROM_DEVICE

[ 007] 0000000113be96e0 vld> ffffe0001421ddd0 0000000000000004
ffffe000039f4720
//
// USBPORT_NeoValidDeviceHandle()
// DeviceHandle:            ffffe0001421ddd0
// ReferenceObj:            ffffe000039f4720

[ 008] 0000000113be9700 devH ffffe0001421ddd0 ffffe000039f4720
0000000000000000
[ 009] 0000000113be9720 pURB ffffe000039f4720 ffffe00000b76cb0
000000000000000b
//
// USBPORT_ProcessURB()
// Urb:                     ffffe000039f4720
// Irp:                     ffffe00000b76cb0
// function:                URB_FUNCTION_GET_DESCRIPTOR_FROM_DEVICE

[ 010] 0000000113be9740 PURB ffffe000039f4720 ffffe000020a44f0
000000000000000b
//
// USBPORT_ProcessURB() - Exit STATUS_PENDING
// Urb:                     ffffe000039f4720
// Irp:                     ffffe000020a44f0
// function:                URB_FUNCTION_GET_DESCRIPTOR_FROM_DEVICE

[ 011] 0000000113be9760 Urb< 0000000000000103 000000000000000b
0000000000000000
[ 012] 0000000113be9780 xSt0 ffffe000012bbf18 0000000000000006
0000000000000001
[ 013] 0000000113be97a0 xnd8 ffffe000012bbf18 ffffe000012bb050
0000000000000000
[ 014] 0000000113be97c0 xnd0 ffffe000012bbf18 ffffe000012bb050
0000000000000000
//
// USBPORT_Xdpc_End()

[ 015] 0000000113be97e0 mapF 0000000000000000 0000000000000000
0000000000000000
[ 016] 0000000113be9800 map5 0000000000000000 0000000000000000
0000000000000000
```

```
[ 017] 0000000113be9820 DMAx 0000000000000000 0000000000000000
0000000000000000
[ 018] 0000000113be9840 subx 0000000000000000 0000000000000000
ffffe0001416801c
[ 019] 0000000113be9860 tmoZ 0000000000000000 ffffe0001416801c
ffffe000141680a4
...
...
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbportmddcontext

Статья • 05.04.2024

Команда **!usbkd.usbportmddcontext** отображает данные контекста USBPORT, если он присутствует в аварийном дампе, созданном в результате **проверки ошибок 0xFE**.

dbgcmd

```
!usbkd.usbportmddcontext
```

## DLL-библиотеки

Usbkd.dll

## Замечания

Используйте эту команду только при отладке файла аварийного дампа, созданного **в результате проверки ошибок 0xFE: BUGCODE_USB_DRIVER**.

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbportmddevext

Статья • 05.04.2024

Команда !usbkd.usbportmddevext отображает структуру usbport!_DEVICE_EXTENSION, **если она присутствует в аварийном дампе, созданном в результате проверки ошибок 0xFE**.

dbgcmd

```
!usbkd.usbportmddevext
```

## DLL-библиотеки

Usbkd.dll

## Замечания

Используйте эту команду только при отладке файла аварийного дампа, созданного **в результате проверки ошибок 0xFE: BUGCODE_USB_DRIVER**.

## Примеры

Ниже приведен пример выходных **данных !usbportmddevext**.

dbgcmd

```
1: kd> !analyze -v
*** ...
BUGCODE_USB_DRIVER (fe)
...
1: kd> !usbkd.usbportmddevext
USBPORT.SYS DEVICE_EXTENSION DATA:
Hcd FDO Extension:
Sig:4f444648 HFDO
CurrentPnpFunc: 0x00000008
PnpFuncHistoryIdx: 0x0000000d
CurrentPowerFunc: 0x00000000
PowerFuncHistoryIdx: 0x00000000
PnpLogIdx: 0x00000002
IoRequestCount: 0x00000007
IoRequestAsyncCallbackCount: 0xffffffff
IoRequestAllow: 0x00000000
Pnp Func History (idx 13)
...
```

```
[02] pnp 13 (0d) IRP_MN_FILTER_RESOURCE_REQUIREMENTS
[...
Power Func History (idx 0)
[01] pnp 255 (ff) ??? (x0) PowerDeviceUnspecified
...
    **Power and Wake ----------------------------------------------
    selective suspend:on (1)
    PowerFlags (00000080):
*---FDO---*
PMDebug: 0x00000000
MinAllocedBw: 0x00000000
MaxAllocedBw: 0x00000000
## ...

## XDPC HISTORY_UsbHcIntDpc

State History (idx 2)
EVENT, STATE, NEXT
Log[3] @ 000000d9e7c615cc
Ev_Xdpc_Worker       XDPC_DpcQueued          XDPC_Running
## ...

## XDPC HISTORY_UsbDoneDpc

State History (idx 0)
EVENT, STATE, NEXT
Log[1] @ 000000d9e7c61774
Ev_Xdpc_Worker       XDPC_DpcQueued          XDPC_Running
## ...

## XDPC HISTORY_UsbMapDpc

State History (idx 3)
EVENT, STATE, NEXT
Log[4] @ 000000d9e7c6196c
## ...

## XDPC HISTORY_UsbIocDpc

State History (idx 0)
EVENT, STATE, NEXT
Log[1] @ 000000d9e7c61b04
Ev_Xdpc_Worker       XDPC_DpcQueued          XDPC_Running
...
```

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbtriage

Статья • 05.04.2024

Команда **!usbkd.usbtriage** отображает список usb-драйверов и объектов устройства.

dbgcmd

```
!usbkd.usbtriage
```

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbtt

Статья • 05.04.2024

Команда **!usbkd.usbtt** отображает сведения из **структуры USBPORT!_TRANSACTION_TRANSLATOR** .

dbgcmd

```
!usbkd.usbtt StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbport!_TRANSACTION_TRANSLATOR**. Чтобы получить список переводчика транзакций для контроллера USB-узла, используйте **команду !usbkd.usbhcdext** .

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов **найти адрес структуры usbport!_TRANSACTION_TRANSLATOR** . Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...
2)!ehci_info ffffe00001ca11a0 !devobj ffffe00001ca1050 PCI: VendorId 8086
DeviceId 293c RevisionId 0002
...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !ehci_info ffffe00001ca11a0.

Щелкните команду DML или передайте адрес расширения устройства в !usbhcdext, **чтобы получить адрес `GlobalTtListHead`**. Передайте этот адрес в **!usbkd.usblist**, который будет отображать адреса **_TRANSACTION_TRANSLATOR** структур.

# См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbtx

Статья • 05.04.2024

Команда **!usbkd.usbtx** отображает сведения из **структуры usbport!_HCD_TRANSFER_CONTEXT** .

dbgcmd

```
!usbkd.usbtx StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbport!_HCD_TRANSFER_CONTEXT**. Чтобы получить список передачи для контроллера USB-узла, используйте **команду !usbkd.usbhcdext** .

## DLL-библиотеки

Usbkd.dll

## Примеры

Вот один из способов **найти адрес usbport!_HCD_TRANSFER_CONTEXT** структуры. Сначала введите **!usbkd.usb2tree**.

dbgcmd

```
0: kd> !usbkd.usb2tree
...
4)!uhci_info ffffe00001c8f1a0 !devobj ffffe00001c8f050 PCI: VendorId 8086
DeviceId 2938 RevisionId 0002
...
```

В предыдущих выходных данных адрес расширения FDO устройства отображается в качестве аргумента команды DML !uhci_info ffffe00001c8f1a0.

Щелкните команду DML или передайте адрес расширения **устройства в !usbhcdext** , чтобы получить список передачи.

dbgcmd

```
0: kd> !usbkd.usbhcdext ffffe00001c8f1a0
...
## I/O TRANSFER LIST(s)

1.) Transfer Request Priority List: (TxQueued) Type: 0-NotSplit, 1-Parent,
2-Child
    ------------------------------------------------------------------------
--------
    [000]!usbtx ffffe0000653401c !usbep ffffe00004730c60 !irp
ffffe00004221220 State: (7)TX_Mapped_inMp
        Priority: 0, Type: 0, Flags= 0000000a, SequenceNum: 10, SplitIdx: 0
        InLen: 4096, OutLen: 0 Status: USBD_STATUS_PENDING (0x40000000)
    ...
```

В предыдущих выходных данных `ffffe0000653401c` используется адрес **структуры _HCD_TRANSFER_CONTEXT**. Передайте этот адрес **в !usbtx**.

dbgcmd

```
0: kd> !usbkd.usbtx ffffe0000653401c

dt usbport!_HCD_TRANSFER_CONTEXT ffffe0000653401c
dt usbport!_TRANSFER_PARAMETERS ffffe0000653417c

## TX HISTORY

## EVENT, STATE, NEXT (latest at bottom)

[01]    (23)Ev_TX_Icsq, (0)TX_Undefined, (1)TX_InQueue
[02]    (5)Ev_TX_MapTransfer, (1)TX_InQueue, (2)TX_MapPending
[03]    (7)Ev_TX_MpSubmitSuccess, (2)TX_MapPending, (7)TX_Mapped_inMp

**DMA**
dt usbport!_TRANSFER_SG_LIST ffffe0000653439c
SgCount:  1  MdlVirtualAddress: ffffe00000437000  MdlSystemAddress:
ffffe00000437000
    [0] dt usbport!_TRANSFER_SG_ENTRY ffffe000065343bc
    : sysaddr: 0000000000000000 len 0x00001000(4096) offset 0x00000000(0)
phys 00000000'ded90000
---
dt usbport!_SCATTER_GATHER_ENTRY ffffe000065343ec
dt _SCATTER_GATHER_LIST ffffe00001bc231c
NumberOfElements = 1
    [0] dt _SCATTER_GATHER_ELEMENT ffffe00001bc232c
     :phys 00000000'ded90000 len 0x00001000(4096)
```

# См. также

Расширения отладчика USB 2.0

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbusb2ep

Статья • 05.04.2024

Команда **!usbkd.usbusb2ep** отображает сведения из **структуры usbport!_USB2_EP** .

dbgcmd

```
!usbkd.usbusb2ep StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbport!_USB2_EP**. Чтобы получить адрес **структуры usbport!_USB2_EP** , используйте **!usbkd.usb2**.

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbusb2tt

Статья • 05.04.2024

Команда **!usbkd.usbusb2tt** отображает сведения из **структуры usbport!_TT** .

```
dbgcmd
```

```
!usbkd.usbusb2tt StructAddr
```

## Параметры

*StructAddr*
**Адрес структуры usbport!_TT**.

## DLL-библиотеки

Usbkd.dll

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !usbkd.usbver

Статья • 05.04.2024

Команда **!usbkd.usbver** отображает версию интерфейса USBD в стеке USB-драйверов.

```
dbgcmd

!usbkd.usbver
```

## DLL-библиотеки

Usbkd.dll

## Замечания

Значение версии интерфейса USBD хранится в переменной `usbport!usbd_version`.

## Примеры

Ниже приведен пример выходных **данных !usbkd.usbver**.

```
dbgcmd

1: kd> !usbkd.usbver

USBD VER 600 USB stack is VISTA
```

## См. также

Расширения отладчика USB 2.0

Драйверы универсальной последовательной шины (USB)

**USBD_IsInterfaceVersionSupported**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# RCDRKD Extensions

18.07.2025

В этом разделе описываются команды расширения отладчика RCDRKD. Эти команды отображают сообщения трассировки WPP, созданные драйверами. Начиная с Windows 8, для анализа сообщений WPP больше не требуется отдельный формат сообщения трассировки (TMF). Сведения TMF хранятся в обычном файле символов (PDB-файл).

Начиная с Windows 10 драйверы режима ядра и пользовательского режима могут использовать средство записи трассировки inflight (IFR) для отслеживания журналов. Драйвер режима ядра может использовать команды RCDRKD для чтения сообщений из циклических буферов, форматирования сообщений и отображения сообщений в отладчике.

**Note** You cannot use the RCDRKD commands to view UMDF driver logs, UMDF framework logs, and KMDF framework logs. Чтобы просмотреть эти журналы, используйте команды расширений Windows Driver Framework (Wdfkd.dll).

Команды расширения отладчика RCDRKD реализуются в Rcdrkd.dll. To load the RCDRKD commands, enter **.load rcdrkd.dll** in the debugger.

Следующие две команды являются основными командами для отображения сообщений трассировки.

- **!rcdrkd.rcdrlogdump**
- **!rcdrkd.rcdrcrashdump**

Следующие вспомогательные команды предоставляют службы, связанные с отображением и сохранением сообщений трассировки.

- **!rcdrkd.rcdrloglist**
- **!rcdrkd.rcdrlogsave**
- **!rcdrkd.rcdrsearchpath**
- **!rcdrkd.rcdrsettraceprefix**
- **!rcdrkd.rcdrtmffile**
- **!rcdrkd.rcdrtraceprtdebug**

The **!rcdrkd.rcdrhelp** displays help for the RCDRKD commands in the debugger.

## См. также

Трассировка программного обеспечения WPP

Использование средства ведения журнала событий фреймворка

Расширения USB 3.0

# !rcdrkd.rcdrhelp

Статья • 05.04.2024

Команда **!rcdrkd.rcdrhelp** отображает справку по командам расширения отладчика RCDRKD.

## DLL-библиотеки

Rcdrkd.dll

## См. также

Расширения RCDRKD

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rcdrkd.rcdrcrashdump

Статья • 05.04.2024

**Расширение !rcdrkd.rcdrcrashdump** используется с мини-файлом для отображения журнала записи (если журнал присутствует в мини-dump).

dbgcmd

```
!rcdrkd.rcdrcrashdump TraceProviderGuid
!rcdrkd.rcdrcrashdump DriverName
```

## Параметры

*TraceProviderGuid*
GUID поставщика трассировки. Этот параметр должен включать фигурные скобки: {*guid*}

*DriverName*
Имя драйвера. Имя драйвера можно использовать вместо GUID поставщика трассировки для драйверов, использующих средство записи трассировки inflight (IFR).

## DLL-библиотеки

Rcdrkd.dll

## См. также

Расширения RCDRKD

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rcdrkd.rcdrlogdump

Статья • 05.04.2024

Расширение **!rcdrkd.rcdrlogdump** отображает сообщения трассировки из всех буферов записей драйвера или набора драйверов.

dbgcmd

```
!rcdrkd.rcdrlogdump DriverName [DriverName ...]
!rcdrkd.rcdrlogdump DriverName -a Address
```

## Параметры

*DriverName*
Имя драйвера, не включая расширение .sys.

*Адрес*
Если указан адрес, эта команда отображает сообщения трассировки из буфера журнала по указанному адресу.

## DLL-библиотеки

Rcdrkd.dll

## Примеры

В следующем примере показана часть выходных **данных команды !rcdrlogdump**.

dbgcmd

```
3: kd> !rcdrlogdump usbxhci.sys
Trace searchpath is:

Trace format prefix is: %7!u!: %!FUNC! -
Log dump command                          Log ID                  Size
================                          ======                  ====
!rcdrlogdump  usbxhci -a fffffa8005ff2b60  03 SLT02 DCI04          1024
!rcdrlogdump  usbxhci -a fffffa8005ff2010  03 SLT02 DCI03          1024
!rcdrlogdump  usbxhci -a fffffa8005b36010  03 SLT01 DCI03          1024
!rcdrlogdump  usbxhci -a fffffa8005b379e0  03 SLT01 DCI04          1024
!rcdrlogdump  usbxhci -a fffffa8005b33350  03 SLT02 DCI01          1024
!rcdrlogdump  usbxhci -a fffffa8005b2bb60  03 SLT01 DCI01          1024
!rcdrlogdump  usbxhci -a fffffa8005a2bb60  03 CMD                  1024
```

```
!rcdrlogdump  usbxhci -a fffffa8005a1ab60  03 INT00                  1024
!rcdrlogdump  usbxhci -a fffffa8005085330  03 RUNDOWN               512
!rcdrlogdump  usbxhci -a fffffa8005311780  03 1033 0194             1024
Trying to extract TMF information from -
C:\ProgramData\dbg\sym\usbxhci.pdb\D4C85D5D3E2843879EDE226A334D69552\usbxhci
.pdb
--- start of log ---
03 RUNDOWN     6: Controller_RetrievePciData - PCI Bus.Device.Function:
48.0.0
03 RUNDOWN     7: Controller_RetrievePciData - PCI: VendorId 0x1033 DeviceId
0x0194 RevisionId 0x03
03 RUNDOWN     8: Controller_QueryDeviceFlagsFromKse - Found DeviceFlags
0x101800 for USBXHCI:PCI\VEN_1033&DEV_0194
03 RUNDOWN     9: Controller_RetrieveDeviceFlags - DeviceFlags 0x101804
...
03 SLT01 DCI03 89911: TransferRing_DispatchEventsAndReleaseLock - 1.3.0:
Mapping Complete : Path 1 TransferRingState_Idle Events 0x00000000
03 SLT01 DCI04 89912: TransferRing_DispatchEventsAndReleaseLock - 1.4.0:
Mapping Begin : Path 3 TransferRingState_Mapping Events 0x00000000
03 SLT01 DCI04 89913: TransferRing_DispatchEventsAndReleaseLock - 1.4.0:
Mapping Complete : Path 3 TransferRingState_Idle Events 0x00000000
---- end of log ----
```

Предыдущие выходные данные содержат сообщения из нескольких буферов журнала. Чтобы просмотреть сообщения из одного буфера журнала, используйте **параметр -a** и укажите адрес буфера журнала. В следующем примере показано, как отображать сообщения из буфера журнала по адресу fffffa8005ff2b60.

dbgcmd

```
3: kd> !rcdrlogdump  usbxhci -a fffffa8005ff2b60
Trace searchpath is:

Trace format prefix is: %7!u!: %!FUNC! -
Trying to extract TMF information from -
C:\ProgramData\dbg\sym\usbxhci.pdb\D4C85D5D3E2843879EDE226A334D69552\usbxhci
.pdb
--- start of log ---
70914: TransferRing_DispatchEventsAndReleaseLock - 2.4.0: Mapping Complete :
Path 1 TransferRingState_Idle Events 0x00000000
70916: TransferRing_TransferEventHandler - 2.4.0: TransferEventTrb
0xFFFFFA8005A3FBF0 CC_SUCCESS Length 31 EventData 1 Pointer
0xfffffa8005b96210
70917: TransferRing_TransferEventHandler - 2.4.0: WdfRequest
0x0000057FFA469FD8 transferData 0xFFFFFA8005B961B0
70918: TransferRing_TransferComplete - 2.4.0: WdfRequest 0x0000057FFA469FD8
completed with STATUS_SUCCESS BytesProcessed 31
70922: TransferRing_DispatchEventsAndReleaseLock - 2.4.0: Mapping Begin :
Path 3 TransferRingState_Mapping Events 0x00000000
70923: TransferRing_DispatchEventsAndReleaseLock - 2.4.0: Mapping Complete :
Path 3 TransferRingState_Idle Events 0x00000000
81064: TransferRing_DispatchEventsAndReleaseLock - 2.4.0: Mapping Begin :
```

```
Path 1 TransferRingState_Mapping Events 0x00000000
81065: TransferRing_StageRetrieveFromRequest - 2.4.0: WdfRequest
0x0000057FFA469FD8 TransferData 0xFFFFFA8005B961B0 Function 0x9 Length 31
TransferSize 31 BytesProcessed 0
81066: TransferRing_DispatchEventsAndReleaseLock - 2.4.0: Mapping Complete :
Path 1 TransferRingState_Idle Events 0x00000000
81068: TransferRing_TransferEventHandler - 2.4.0: TransferEventTrb
0xFFFFFA8005A40270 CC_SUCCESS Length 31 EventData 1 Pointer
0xfffffa8005b96210
81069: TransferRing_TransferEventHandler - 2.4.0: WdfRequest
0x0000057FFA469FD8 transferData 0xFFFFFA8005B961B0
81070: TransferRing_TransferComplete - 2.4.0: WdfRequest 0x0000057FFA469FD8
completed with STATUS_SUCCESS BytesProcessed 31
81074: TransferRing_DispatchEventsAndReleaseLock - 2.4.0: Mapping Begin :
Path 3 TransferRingState_Mapping Events 0x00000000
81075: TransferRing_DispatchEventsAndReleaseLock - 2.4.0: Mapping Complete :
Path 3 TransferRingState_Idle Events 0x00000000
---- end of log ----
```

## См. также

Расширения RCDRKD

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rcdrkd.rcdrloglist

Статья • 05.04.2024

Расширение **!rcdrkd.rcdrloglist** отображает список журналов записей, принадлежащих драйверу или набору драйверов.

dbgcmd

```
!rcdrkd.rcdrloglist DriverName [DriverName ...]
```

## Параметры

*DriverName*
Имя драйвера, не включая расширение .sys.

## DLL-библиотеки

Rcdrkd.dll

## Замечания

Эта команда относится только к драйверам, которые регистрируют сообщения в разные журналы с помощью API WppRecorder.

## Примеры

В следующем примере отображается список всех журналов записей, принадлежащих драйверу контроллера узла USB 3.0 (usbxhci.sys).

dbgcmd

```
3: kd> !rcdrloglist usbxhci
Log dump command                          Log ID                  Size
================                          ======                  ====
!rcdrlogdump  usbxhci -a fffffa8005ff2b60  03 SLT02 DCI04          1024
!rcdrlogdump  usbxhci -a fffffa8005ff2010  03 SLT02 DCI03          1024
!rcdrlogdump  usbxhci -a fffffa8005b36010  03 SLT01 DCI03          1024
!rcdrlogdump  usbxhci -a fffffa8005b379e0  03 SLT01 DCI04          1024
!rcdrlogdump  usbxhci -a fffffa8005b33350  03 SLT02 DCI01          1024
!rcdrlogdump  usbxhci -a fffffa8005b2bb60  03 SLT01 DCI01          1024
!rcdrlogdump  usbxhci -a fffffa8005a2bb60  03 CMD                  1024
```

```
!rcdrlogdump  usbxhci -a fffffa8005a1ab60  03 INT00                  1024
!rcdrlogdump  usbxhci -a fffffa8005085330  03 RUNDOWN               512
!rcdrlogdump  usbxhci -a fffffa8005311780  03 1033 0194            1024
```

## См. также

Расширения RCDRKD

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rcdrkd.rcdrlogsave

Статья • 05.04.2024

Расширение **!rcdrkd.rcdrlogsave** сохраняет буферы записей драйвера.

dbgcmd

```
!rcdrkd.rcdrlogsave DriverName [CaptureFilename ]
```

## Параметры

*DriverName*
Имя драйвера, не включая расширение .sys.

*CaptureFileName*
Имя файла (не включая расширение ETL), в котором сохраняются буферы записей. Если *CaptureFileName* не задано, буферы записей сохраняются в *DriverName.etl*.

## DLL-библиотеки

Rcdrkd.dll

## См. также

Расширения RCDRKD

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rcdrkd.rcdrsearchpath

Статья • 05.04.2024

Расширение **!rcdrkd.rcdrsearchpath** задает путь поиска для файлов формата сообщения трассировки (TMF) и управления сообщениями трассировки (TMC).

dbgcmd

```
!rcdrkd.rcdrsearchpath FilePath
```

## Параметры

*FilePath*
Путь к файлам форматирования.

## DLL-библиотеки

Rcdrkd.dll

## Замечания

Путь поиска, заданный этой командой, имеет приоритет над путем поиска, указанным в переменной среды TRACE_FORMAT_SEARCH_PATH.

## См. также

Расширения RCDRKD

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rcdrkd.rcdrsettraceprefix

Статья • 05.04.2024

Расширение **!rcdrkd.rcdrsettraceprefix** задает префикс сообщения трассировки.

```
dbgcmd

!rcdrkd.rcdrsettraceprefix TracePrefixString
```

## Параметры

*TracePrefixString*
Строка префикса сообщения трассировки.

## DLL-библиотеки

Rcdrkd.dll

## Замечания

Каждое сообщение в журнале записи имеет префикс, который можно контролировать, указав строку префикса сообщения трассировки. Дополнительные сведения см. в разделе Префикс сообщения трассировки.

## Примеры

В следующем примере префикс сообщения трассировки изначально **равен %7!u!: %! FUNC! -** . Параметр **%7!u!** указывает, что префикс содержит номер последовательности сообщений. Параметр **%! FUNC!** указывает, что префикс содержит имя функции, создающей сообщение. В примере вызывается !rcdrsettraceprefix**, чтобы изменить строку префикса на %7!u!.** После этого отображение журнала содержит номера последовательности сообщений, но не включает имена функций.

```
dbgcmd

0: kd> !rcdrlogdump USBXHCI -a 0xfffffa8010737b60
Trace searchpath is:

Trace format prefix is: %7!u!: %!FUNC! -
```

```
Trying to extract TMF information from -
C:\ProgramData\dbg\sym\usbxhci.pdb\D4C85D5D3E2843879EDE226A334D69552\usbxhci
.pdb
--- start of log ---
405: TransferRing_DispatchEventsAndReleaseLock - 1.8.1: Mapping Begin : Path
1 TransferRingState_Mapping Events 0x00000000
406: TransferRing_StageRetrieveFromRequest - 1.8.1: WdfRequest
0x0000057FEE117A88 TransferData 0xFFFFFA8011EE8700 Function 0x9 Length 500
TransferSize 500 BytesProcessed 0
...
---- end of log ----

0: kd> !rcdrsettraceprefix %7!u!:
SetTracePrefix: "%7!u!: "
0: kd> !rcdrlogdump USBXHCI -a 0xfffffa8010737b60
Trace searchpath is:

Trace format prefix is: %7!u!:
Trying to extract TMF information from -
C:\ProgramData\dbg\sym\usbxhci.pdb\D4C85D5D3E2843879EDE226A334D69552\usbxhci
.pdb
--- start of log ---
405: 1.8.1: Mapping Begin : Path 1 TransferRingState_Mapping Events
0x00000000
406: 1.8.1: WdfRequest 0x0000057FEE117A88 TransferData 0xFFFFFA8011EE8700
Function 0x9 Length 500 TransferSize 500 BytesProcessed 0
...
---- end of log ----
```

## См. также

Расширения RCDRKD

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rcdrkd.rcdrtmffile

Статья • 05.04.2024

Расширение **!rcdrkd.rcdrtmffile** задает или очищает имя файла формата сообщения трассировки (TMF).

dbgcmd

```
!rcdrkd.rcdrtmffile [Filename]
```

## Параметры

*Имя файла*
Имя файла TMF. Если этот параметр не указан, имя файла очищается.

## DLL-библиотеки

Rcdrkd.dll

## См. также

Расширения RCDRKD

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rcdrkd.rcdrtraceprtdebug

Статья • 08.07.2024

Расширение **!rcdrkd.rcdrtraceprtdebug** включает или отключает режим диагностики TracePrt. Это расширение должно использоваться в направлении поддержки.

dbgcmd

```
!rcdrkd.rcdrtraceprtdebug {on|off}
```

## Параметры

**on**

Включает режим диагностики TracePrt.

**от**

Включает режим диагностики TracePrt.

## DLL-библиотеки

Rcdrkd.dll

## См. также

Расширения RCDRKD

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# HID Extensions

18.07.2025

В этом разделе описываются команды расширения отладчика "Устройство с человеческим интерфейсом" (HID).

Команды расширения отладчика HID реализуются в Hidkd.dll. To load the HID commands, enter **.load hidkd.dll** in the debugger.

## Начало работы с расширениями HID

To start debugging a HID issue, enter the **!hidtree** command. The **!hidtree** command displays a list of commands and addresses that you can use to investigate device objects, preparsed HID data, and HID report descriptors.

## В этом разделе

Развернуть таблицу

| Topic | Description |
| --- | --- |
| !hidkd.help | The !hidkd.help command displays help for the HID debugger extension commands. |
| !hidkd.hidfdo | The !hidkd.hidfdo command displays HID information associated with a functional device object (FDO). |
| !hidkd.hidpdo | The !hidkd.hidpdo command displays HID information associated with a physical device object (PDO). |
| !hidkd.hidtree | The !hidkd.hidtree command displays a list of all device nodes that have a HID function driver along with their child nodes. Дочерние узлы имеют объект физического устройства (PDO), созданный драйвером функции HID родительского узла. |
| !hidkd.hidppd | The !hidkd.hidppd command displays HID preparsed data. |
| !hidkd.hidrd | The !hidkd.hidrd command displays a HID report descriptor in both raw and parsed format. |

# См. также

RCDRKD Extensions

Специализированные команды расширения

# !hidkd.help

Статья • 05.04.2024

Расширение **!hidkd.help** отображает справку по командам расширения отладчика HID.

## DLL-библиотеки

Hidkd.dll

## См. также

Расширения HID

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !hidkd.hidfdo

Статья • 05.04.2024

Команда **расширения !hidkd.hidfdo** отображает сведения HID, связанные с функциональным объектом устройства (FDO).

dbgcmd

```
!hidkd.hidfdo fdo
```

## Параметры

*Fdo*
Адрес FDO. Чтобы получить адреса FDOs, связанные с драйверами HID, используйте **команду !usbhid.hidtree** .

## DLL-библиотеки

Hidkd.dll

## Примеры

Ниже приведен пример выходных **данных команды !hidfdo** . В примере сначала вызывается **!hidtree** , чтобы получить адрес FDO.

dbgcmd

```
0: kd> !hidkd.hidtree
HID Device Tree
...
FDO  VendorID:0x045E(Microsoft Corporation) ProductID:0x0745 Version:0x0634
!hidfdo 0xffffe00004f466e0
...
0: kd> !hidfdo 0xffffe00004f466e0
# FDO 0xffffe00004f466e0  (!devobj/!devstack)

  Name              : \Device\_HID00000002
  Vendor ID         : 0x045E(Microsoft Corporation)
  Product ID        : 0x0745
  Version Number    : 0x0634
  Is Present?       : Y
  Report Descriptor : !hidrd 0xffffe00004281a80 0x127
  Per-FDO IFR Log   : !rcdrlogdump HIDCLASS -a 0xFFFFE0000594D000
```

```
Position in HID tree

dt FDO_EXTENSION 0xffffe00004f46850
```

## См. также

Расширения HID

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !hidkd.hidpdo

Статья • 05.04.2024

Команда **расширения !hidkd.hidpdo** отображает сведения HID, связанные с объектом физического устройства (PDO).

dbgcmd

```
!hidkd.hidpdo pdo
```

## Параметры

*Pdo*
Адрес PDO. Чтобы получить адреса PDOS, связанные с драйверами HID, используйте **команду !usbhid.hidtree** .

## DLL-библиотеки

Hidkd.dll

## Примеры

Ниже приведен пример выходных **данных команды !hidpdo** . В примере сначала вызывается **!hidtree** , чтобы получить адрес PDO.

dbgcmd

```
0: kd> !hidkd.hidtree
HID Device Tree
...
FDO  VendorID:0x045E(Microsoft Corporation) ProductID:0x0745 Version:0x0634
...
    PDO  Generic Desktop Controls (0x01) | Mouse (0x02)
    !hidpdo 0xffffe000056281e0
    ...

0: kd> !hidpdo 0xffffe000056281e0
# PDO 0xffffe000056281e0  (!devobj/!devstack)

  Collection Num  : 1
  Name            : \Device\_HID00000002#COLLECTION00000001
  FDO             : !hidfdo 0xffffe00004f466e0
  Per-FDO IFR Log : !rcdrlogdump HIDCLASS -a 0xFFFFE0000594D000
```

```
  Usage Page      : Generic Desktop Controls (0x01)
  Usage           : Mouse (0x02)
  Report Length   : 0xa(Input) 0x0(Output) 0x2(Feature)
  Pre-parsed Data : 0xffffe00003742840
  Position in HID tree

  dt PDO_EXTENSION 0xffffe00005628350

## Power States

  Power States  : S0/D0
  Wait Wake IRP : !irp 0xffffe00004fc57d0 (pending on \Driver\HidUsb)
```

# См. также

Расширения HID

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !hidkd.hidtree

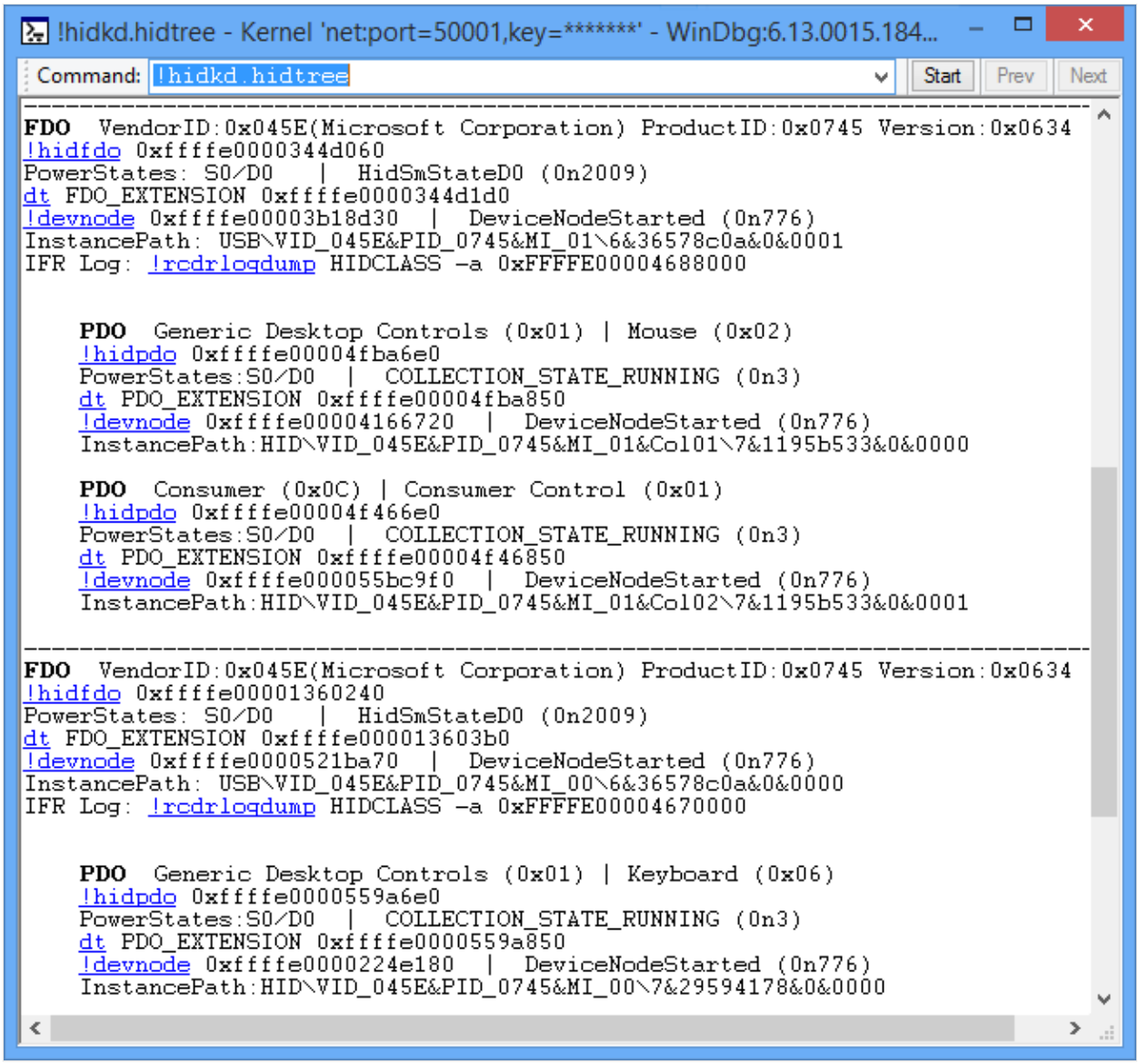
Статья • 05.04.2024

Команда **расширения !hidkd.hidtree** отображает список всех узлов устройств с драйвером функции HID вместе со своими дочерними узлами. Дочерние узлы имеют объект физического устройства (PDO), созданный драйвером функции HID родительского узла.

dbgcmd

```
!hidkd.hidtree
```

На снимке экрана показан пример выходных **данных команды !hidtree** .




В этом примере есть два узла устройства с драйвером функции HID. Функциональный объект устройства (FDO) представляет драйвер HID в этих двух узлах. Первый узел FDO имеет два дочерних узла, а второй узел FDO имеет один

дочерний узел. В выходных данных отладчика дочерние узлы имеют заголовок PDO.

**Обратите внимание,** что этот набор узлов устройств не формирует дерево с одним корневым узлом. Узлы устройства с драйверами функций HID можно изолировать друг от друга.

При отладке проблемы **HID функция !hidtree** является хорошим местом для запуска, так как команда отображает несколько адресов, которые можно передать другим командам отладчика HID. Выходные данные используют язык разметки отладчика (DML) для предоставления ссылок. Ссылки выполняют команды, которые предоставляют подробные сведения, связанные с отдельным узлом устройства. Например, можно получить сведения о FDO, щелкнув одну из ссылок !hidfdo**.** В качестве альтернативы щелчку ссылки можно ввести команду. Например, чтобы просмотреть подробные сведения о первом узле в предыдущих выходных данных, можно ввести команду **!devnode 0xffffe00003b18d30**.

**Обратите внимание** , что функция DML доступна в WinDbg, но не в Visual Studio или KD.

## DLL-библиотеки

Hidkd.dll

## См. также

Расширения HID

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !hidkd.hidppd.hidppd

Статья • 05.04.2024

Команда **расширения !hidkd.hidppd отображает предварительно подготовленные данные HID** .

dbgcmd

```
!hidkd.hidppd ppd
```

## Параметры

*Ppd*
**Адрес структуры HIDP_PREPARSED_DATA**. Чтобы получить адрес структуры HIDP_PREPARSED_DATA, используйте **!hidfdo или !hidpdo**.

## DLL-библиотеки

Hidkd.dll

## Примеры

В этом примере показано, как использовать !hidpdo**, за которым следует !hidppd.** В выходных **данных !hidpdo** отображается адрес **структуры HIDP_PREPARSED_DATA** .

dbgcmd

```
0: kd> !hidpdo 0xffffe000029f6060
## PDO 0xffffe000029f6060  (!devobj/!devstack)

  Collection Num  : 1
  Name            : \Device\_HID00000000#COLLECTION00000001
  ...
  Pre-parsed Data : 0xffffe000029d1010
  ...

0: kd> !hidkd.hidppd 0xffffe000029d1010
Reading preparsed data...
Preparsed Data at 0xffffe000029d1010

## Summary
```

```
  UsagePage             : Vendor-defined (0xFFA0)
  Usage                 : 0x01
  Report Length         : 0x2(Input) 0x2(Output) 0x0(Feature)
  Link Collection Nodes : 2
  Button Caps           : 0(Input) 0(Output) 0(Feature)
  Value Caps            : 1(Input) 1(Output) 0(Feature)
  Data Indices          : 1(Input) 1(Output) 0(Feature)

## Input Value Capability #0

  Report ID         : 0x0
  Usage Page        : Vendor-defined (0xFFA1)
  Bit Size          : 0x8
  Report Count      : 0x1
  ...
```

# См. также

Расширения HID

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !hidkd.hidrd

Статья • 05.04.2024

Команда **расширения !hidkd.hidrd** отображает дескриптор отчета HID в формате необработанного и синтаксического анализа.

dbgcmd

```
!hidkd.hidrd rd Length
```

## Параметры

*rd*
Адрес необработанных данных дескриптора отчета. Чтобы получить адрес данных дескриптора, используйте **команду !hidfdo** .

*Длина*
Длина в байтах необработанных данных дескриптора отчета. Чтобы получить длину **, используйте команду !hidfdo** .

## DLL-библиотеки

Hidkd.dll

## Примеры

В этом примере показано, как использовать команду !hidfdo**, за которой следует команда !hidrd.** Выходные данные **!hidfdo** показывают адрес и длину необработанных данных дескриптора отчета.

dbgcmd

```
0: kd> !hidfdo 0xffffe00004f466e0
# FDO 0xffffe00004f466e0  (!devobj/!devstack)

  Name              : \Device\_HID00000002
  ...
  Report Descriptor : !hidrd 0xffffe00004281a80 0x127
  ...

0: kd> !hidrd 0xffffe00004281a80 0x127
Report Descriptor at 0xffffe00004281a80
```

```
## Raw Data

0x0000: 05 01 09 02 A1 01 05 01-09 02 A1 02 85 1A 09 01
0x0010: A1 00 05 09 19 01 29 05-95 05 75 01 15 00 25 01
0x0020: 81 02 75 03 95 01 81 01-05 01 09 30 09 31 95 02
...

## Parsed

Usage Page (Generic Desktop Controls).....................0x0000: 05 01
Usage (Mouse).............................................0x0002: 09 02
Collection (Application)..................................0x0004: A1 01
..Usage Page (Generic Desktop Controls)...................0x0006: 05 01
..Usage (Mouse)...........................................0x0008: 09 02
..Collection (Logical)....................................0x000A: A1 02
....Report ID (26)........................................0x000C: 85 1A
...
End Collection ()........................................0x0126: C0
```

## См. также

Расширения HID

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения средства ведения журнала (Logexts.dll)

Статья • 26.10.2023

Команды расширения, связанные с средствами Logger и LogViewer, можно найти в Logexts.dll.

Эта библиотека DLL отображается в каталоге winext. Его можно использовать со всеми операционными системами Windows.

# !logexts.help

Статья • 05.04.2024

Расширение **!logexts.help** отображает текст справки в окне командной строки со всеми командами расширения Logexts.dll.

dbgcmd

```
!logexts.help
```

## DLL-библиотеки

Logexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе Loger и LogViewer.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !logexts.logb

Статья • 05.04.2024

Расширение **!logexts.logb** отображает или очищает выходной буфер.

```
dbgcmd
```

```
!logexts.logb p
!logexts.logb f
```

## Параметры

**p**
Приводит к отображению содержимого выходного буфера в отладчике.

**f**
Сбрасывает выходной буфер на диск.

## DLL-библиотеки

Logexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе Loger и LogViewer.

## Замечания

В качестве рекомендации по производительности выходные данные журнала сбрасываются на диск только при заполнении выходного буфера. По умолчанию буфер составляет 2144 байта.

Расширение **P !logexts.logb** отображает содержимое буфера в отладчике.

Расширение **!logexts.logb f** сбрасывает буфер в файлы журнала. Так как буферная память управляется целевым приложением, автоматическое запись буфера на диск не произойдет, если в целевом приложении возникает нарушение доступа или другая неустранимая ошибка. В таких случаях следует использовать эту команду, чтобы вручную очистить буфер на диск. В противном случае в файлах журнала могут не отображаться последние api-интерфейсы.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !logexts.logc

Статья • 05.04.2024

Расширение **!logexts.logc** отображает все категории API, отображает все API в определенной категории или включает и отключает ведение журнала API в одной или нескольких категориях.

dbgcmd

```
!logexts.logc e Categories
!logexts.logc d Categories
!logexts.logc p Category
!logexts.logc
```

## Параметры

**e**
Включает ведение журнала указанных категорий.

**дн.**
Отключает ведение журнала указанных категорий.

*Категории*
Указывает категории, которые необходимо включить или отключить. Если перечислены несколько категорий, разделите их пробелами. Звездочка (*) может использоваться для указания всех категорий.

**p**
Отображает все API, принадлежащие указанной категории.

*Категория*
Указывает категорию, api-интерфейсы которой будут отображаться. С параметром **p** можно указать только одну категорию.

## DLL-библиотеки

Logexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе Loger и LogViewer.

# Замечания

Без каких-либо параметров **!logexts.logc** отобразит текущий список доступных категорий и будет указывать, какие из них включены и отключены.

Если категория отключена, перехватчики для всех API в этой категории будут удалены, чтобы больше не было затрат на производительность. Перехватчики COM не удаляются, так как их невозможно повторно включить.

Включение только определенных категорий может быть полезно, если вы заинтересованы только в определенном типе взаимодействия, которое программа имеет с Windows (например, операции с файлами). Это уменьшает размер файла журнала, а также снижает влияние средства ведения журнала на скорость выполнения процесса.

Следующая команда включает ведение журнала всех категорий:

dbgcmd

```
0:000> !logexts.logc e *
```

Следующая команда отключит ведение журнала категории 7:

dbgcmd

```
0:000> !logexts.logc d 7
```

Следующая команда включает ведение журнала категорий 13 и 15:

dbgcmd

```
0:000> !logexts.logc e 13 15
```

Следующая команда отображает все API, принадлежащие категории 3:

dbgcmd

```
0:000> !logexts.logc p 3
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !logexts.logd

Статья • 05.04.2024

Расширение **!logexts.logd** отключает ведение журнала.

dbgcmd

```
!logexts.logd
```

## DLL-библиотеки

Logexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе Loger и LogViewer.

## Замечания

Это приведет к удалению всех перехватчиков API в попытке разрешить программе работать свободно. Перехватчики COM не удаляются, так как их невозможно повторно включить.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !logexts.loge

Статья • 05.04.2024

Расширение **!logexts.loge** включает ведение журнала. Если ведение журнала не было инициализировано, оно будет инициализировано и включено.

```
dbgcmd

    !logexts.loge [OutputDirectory]
```

## Параметры

*OutputDirectory*
Указывает каталог, используемый для вывода. Если *указан выходной каталог* , он должен существовать - отладчик не создаст его. Если указан относительный путь, он будет относительно каталога, из которого был запущен отладчик. Если *outputDirectory* опущен, будет использоваться путь к рабочему столу. Отладчик создаст подкаталог LogExts outputDirectory*, и все выходные данные средства ведения журнала будут помещены в этот подкаталог*.

## DLL-библиотеки

Logexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе Loger и LogViewer.

## Замечания

Если средство ведения журнала еще не было внедрено в целевое приложение **расширением !logexts.logi** , **расширение !logexts.loge** введет средство ведения журнала в целевой объект, а затем включите ведение журнала.

Если **!logexts.logi** уже запущен, нельзя включить *OutputDirectory*, так как выходной каталог уже установлен.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !logexts.logi

Статья • 05.04.2024

Расширение **!logexts.logi** инициализирует ведение журнала путем внедрения средства ведения журнала в целевое приложение.

dbgcmd

```
    !logexts.logi [OutputDirectory]
```

## Параметры

*OutputDirectory*
Указывает каталог, используемый для вывода. Если *указан выходной каталог* , он должен существовать - отладчик не создаст его. Если указан относительный путь, он будет относительно каталога, из которого был запущен отладчик. Если *outputDirectory* опущен, будет использоваться путь к рабочему столу. Отладчик создаст подкаталог LogExts outputDirectory*, и все выходные данные средства ведения журнала будут помещены в этот подкаталог*.

## DLL-библиотеки

Logexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе Loger и LogViewer.

## Замечания

Эта команда инициализирует ведение журнала, но на самом деле не включает его. Ведение журнала можно включить с помощью команды !logexts.loge.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !logexts.logm

Статья • 05.04.2024

Расширение **!logexts.logm** создает или отображает список включения модуля или список исключений модуля.

dbgcmd

```
!logexts.logm i Modules
!logexts.logm x Modules
!logexts.logm
```

## Параметры

**i**
Заставляет средство ведения журнала использовать список включения модуля. Он будет состоять из указанных модулей.

**x**
Вызывает использование списка исключений модуля в средстве ведения журнала. Он будет состоять из Logexts.dll, kernel32.dll и указанных модулей.

*Модули*
Указывает модули, которые должны быть включены или исключены. Этот список не является накопительным; каждое использование этой команды создает совершенно новый список. Если перечислены несколько модулей, разделите их пробелами. Звездочка (*) может использоваться для указания всех модулей.

## DLL-библиотеки

Logexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе Loger и LogViewer.

## Замечания

Без параметров **расширение !logexts.logm** отображает текущий список включения или список исключений.

Расширения **!logexts.logm x *** и **!logexts.logm i** эквивалентны: они приводят к совершенно пустому списку включения.

Расширения **!logexts.logm i *** и **!logexts.logm x** эквивалентны: они приводят к списку исключений, который содержит только Logexts.dll и kernel32.dll. Эти два модуля всегда исключены, так как средство ведения журнала не разрешено регистрировать себя.

Далее приводятся некоторые примеры.

dbgcmd

```
0:001> !logm
Excluded modules:
  LOGEXTS.DLL      [mandatory]
  KERNEL32.DLL     [mandatory]
  USER32.DLL
  GDI32.DLL
  ADVAPI32.DLL

0:001> !logm x winmine.exe
Excluded modules:
  Logexts.dll      [mandatory]
  kernel32.dll     [mandatory]
  winmine.exe

0:001> !logm x user32.dll gdi32.dll
Excluded modules:
  Logexts.dll      [mandatory]
  kernel32.dll     [mandatory]
  user32.dll
  gdi32.dll

0:001> !logm i winmine.exe mymodule2.dll
Included modules:
  winmine.exe
  mymodule2.dll
```


## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !logexts.logo

Статья • 05.04.2024

Расширение **!logexts.logo** задает или отображает параметры вывода средства ведения журнала.

dbgcmd

```
!logexts.logo {e|d} {d|t|v}
!logexts.logo
```

## Параметры

**e|d**
Указывает, следует ли включить (e) или отключить (d) указанный тип вывода.

**d|t|v**
Указывает тип выходных данных. Возможны три типа выходных данных средства ведения журнала: сообщения, отправленные непосредственно отладчику (d), текстовый файл (t) или подробный LGV-файл (v).

## DLL-библиотеки

Logexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе Loger и LogViewer.

## Замечания

Если **!logexts.logo** используется без каких-либо параметров, отображается текущее состояние ведения журнала, выходной каталог и текущие параметры отладчика, текстовый файл и подробный журнал:

dbgcmd

```
0:000> !logo
Logging currently enabled.
```

```
Output directory: MyLogs\LogExts\

Output settings:
  Debugger            Disabled
  Text file           Enabled
  Verbose log         Enabled
```

В предыдущем примере выходной каталог является относительным путем, поэтому он расположен относительно каталога, в котором были запущены отладчики.

Чтобы отключить подробное ведение журнала, используйте следующую команду:

dbgcmd

```
0:000> !logo d v
  Debugger            Disabled
  Text file           Enabled
  Verbose log         Disabled
```

Текстовый файл и LGV-файлы будут помещены в текущий выходной каталог. Чтобы прочитать LGV-файл, используйте LogViewer.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения NDIS (Ndiskd.dll)

16.08.2025

В этом разделе описываются команды, доступные в !ndiskd, расширение отладчика, полезное для отладки драйверов NDIS (спецификация интерфейса сетевого устройства). Эти команды позволяют разработчикам сетевых драйверов видеть более широкий вид сетевого стека Windows и взаимодействие их драйверов с ним. С помощью !ndiskd можно просмотреть состояние всех сетевых адаптеров (**!ndiskd.netadapter**), визуальную схему сетевого стека компьютера (**!ndiskd.netreport**), журнал трафика на сетевых адаптерах (**!ndiskd.nbllog**) или список всех ожидающих запросов OID (**!ndiskd.oid**).

Команды можно найти в Ndiskd.dll. Чтобы загрузить символы, введите **reload /f ndis.sys** в командном окне отладчика. Чтобы убедиться, что символы загружены успешно, используйте расширение **!lmi ndis** и найдите фразу "Символы загружены успешно" в нижней части. Ваш результат должен выглядеть подобно следующему примеру:

dbgcmd

```
0: kd> !lmi ndis
Loaded Module Info: [ndis]
         Module: ndis
   Base Address: fffff80174570000
     Image Name: ndis.sys
   Machine Type: 34404 (X64)
     Time Stamp: 938f9f4e (This is a reproducible build file hash, not a true
timestamp)
           Size: 16f000
       CheckSum: 167a05
Characteristics: 22
Debug Data Dirs: Type  Size     VA  Pointer
             CODEVIEW    21, d4060,   d2c60 RSDS - GUID: {9CC82DBE-96A0-773D-29E0-
62B698C4C3A8}
               Age: 1, Pdb: ndis.pdb
                 POGO   988, d4084,   d2c84 [Data not mapped]
                REPRO    24, d4a0c,   d360c Reproducible build[Data not mapped]
     Image Type: MEMORY   - Image read successfully from loaded memory.
    Symbol Type: PDB      - Symbols loaded successfully from symbol server.

C:\ProgramData\Dbg\sym\ndis.pdb\9CC82DBE96A0773D29E062B698C4C3A81\ndis.pdb
    Load Report: public symbols , not source indexed

C:\ProgramData\Dbg\sym\ndis.pdb\9CC82DBE96A0773D29E062B698C4C3A81\ndis.pdb
```

# Гиперссылки !ndiskd

Многие команды расширения в !ndiskd представляют гиперссылки в результатах, которые они отображаются в окне отладчика. Текст этих гиперссылок был оставлен в примерах, предоставленных для иллюстрации точного формата того, что вы увидите при выполнении команды на компьютере отладочного пользователя. Некоторые из примеров также ссылаются на явное нажатие этих ссылок, чтобы понять типичные потоки использования, хотя примеры также предоставляют альтернативные формы командной строки каждой команды.

# Общие параметры

Все команды !ndiskd поддерживают следующие универсальные параметры.

*-многословный*
Отображает дополнительные сведения.

*-сжатый*
Подавляет некоторые стандартные выходные данные.

*-статический*
Подавляет некоторые интерактивные выходные данные.

*-dml 0|1*
Определяет, включены ли выходные данные DML (язык разметки отладчика).

*-юникод 0|1*
Определяет, разрешен ли вывод символов Юникода.

*-indent N*
Использует *N* пробелов на уровне отступа.

*-сила*
Переопределяет некоторые проверки безопасности на удаленных данных.

*-tracedata*
Отображает подробные сообщения трассировки для отладки !ndiskd.

# Net Adapter, NDIS Driver и General Commands

В следующих командах отображаются сведения о сетевых адаптерах компьютера, сетевых драйверах и общих командах, связанных с сетевым стеком (например, rcvqueues, открытие, фильтры, OID и блокировки RW).

- **!ndiskd.netadapter**

- !ndiskd.minidriver
- !ndiskd.rcvqueue
- !ndiskd.protocol
- !ndiskd.mopen
- !ndiskd.filter
- !ndiskd.filterdriver
- !ndiskd.oid
- !ndiskd.ndisrwlock
- !ndiskd.netreport

## команды NET_BUFFER_LIST и NET_BUFFER

В следующих командах отображаются сведения о NET_BUFFER_LIST и структурах NET_BUFFER .

- !ndiskd.nbl
- !ndiskd.nb
- !ndiskd.nblpool
- !ndiskd.nbpool
- !ndiskd.pendingnbls
- !ndiskd.nbllog

## Команды NetAdapterCx

В следующих командах отображаются сведения, связанные с расширением класса WDF WDF NetAdapterCx и связанными структурами, NET_RING_BUFFER и NET_PACKET.

- !ndiskd.cxadapter
- !ndiskd.netqueue
- !ndiskd.netrb
- !ndiskd.netpacket
- !ndiskd.netfragment
- !ndiskd.nrc
- !ndiskd.netring

## Команды сетевого интерфейса

В следующих командах отображаются сведения, относящиеся к сетевым интерфейсам.

- !ndiskd.interfaces
- !ndiskd.ifprovider

# Команды NDIS_PACKET

В следующих командах отображаются сведения о структурах NDIS_PACKET . Эти расширения предназначены для устаревших драйверов NDIS 5.x. Структура NDIS_PACKET и связанная с ней архитектура устарели.

- **!ndiskd.pkt**
- **!ndiskd.pktpools**
- **!ndiskd.findpacket**

# Команды CoNDIS

В следующих командах отображаются сведения о подключенияхConnection-Oriented NDIS .

- **!ndiskd.vc**
- **!ndiskd.af**

# Команды отладки NDIS

В следующих командах отображаются сведения, касающиеся ссылок NDIS, журналов событий, трассировок стека и отладки трассировок.

- **!ndiskd.ndisref**
- **!ndiskd.ndisevent**
- **!ndiskd.ndisstack**
- **!ndiskd.dbglevel**
- **!ndiskd.dbgsystems**

# Команды WDI

В следующих командах отображаются сведения о драйверах минипорта WDI.

- **!ndiskd.wdiadapter**
- **!ndiskd.wdiminidriver**
- **!ndiskd.nwadapter**

# Команды NDIS и !ndiskd Information Commands

Следующие команды отображают сведения о NDIS.sys и ndiskd.dll.

- **!ndiskd.ndis**
- **!ndiskd.ndiskdversion**

## Другие команды

- **!ndiskd.ifstacktable**
- **!ndiskd.compartments**
- **!ndiskd.ndisslot**

## Связанные разделы

Дополнительные сведения о разработке драйверов NDIS для Windows Vista и более поздних версий см. в руководстве по проектированию сетевых драйверов.

Дополнительные сведения о справочнике по драйверам NDIS для Windows Vista и более поздних версий см. в справочнике по сети Windows Vista и более поздних версий.

Демонстрация использования команд отладчика !ndiskd для отладки сетевого стека см. в видео отладке канала Network Stack 9.

# !ndiskd.help

16.08.2025

Команда **!ndiskd.help** отображает список доступных команд !ndiskd с кратким описанием каждого из них.

Консоль

```
!ndiskd.help
```

# Библиотека dll

Ndiskd.dll

# Примеры

В следующем примере показан список команд справки с помощью **!ndiskd.help**.

Консоль

```
3: kd> !ndiskd.help

NDIS KD EXTENSIONS

    help               This help and lots more
    netadapter         Show network adapters  (this is a good starting place)
    protocol           Show protocol drivers
    mopen              Show open bindings between netadapter and protocol
    filter             Show light-weight filters
    nbl                Show information about an NET_BUFFER_LIST
    oid                Show pending OID Requests
    ndis               Show NDIS.sys build info
    netreport          Draw a box diagram of your network stack

    Show more extensions
    View examples & tutorials online
```

Используя **!ndiskd.help -all**, вы получите более подробный список, как показано в следующем примере.

**Заметка**
В нижней части этого примера перечислены некоторые альтернативные команды. Эти команды доступны разработчикам драйверов NDIS, которые использовали их раньше, но мы рекомендуем использовать основные команды.

Консоль

```
3: kd> !ndiskd.help -all

NDIS KD EXTENSIONS

    Primary commands:
    help               This help and lots more
    netadapter         Show network adapters  (this is a good starting place)
        minidriver     Show network adapter drivers
        rcvqueue       Show a receive queue (c.f. VMQ)
    protocol           Show protocol drivers
    mopen              Show open bindings between netadapter and protocol
    filter             Show light-weight filters
        filterdriver   Show filter drivers (not to be confused with filters)
    nbl                Show information about an NET_BUFFER_LIST
    nb                 Show information about an NET_BUFFER
    nblpool            Show information about an NET_BUFFER_LIST pool
    nbpool             Show information about an NET_BUFFER pool
    pendingnbls        Show all NET_BUFFER_LISTs that are in transit
    nbllog             Show a log of all NET_BUFFER_LIST activity
    oid                Show pending OID Requests
    interfaces         Show interfaces (à la NDISIF)
        ifprovider     Show registered NDIS interface providers
        ifstacktable   Show the ifStackTable
        networks       Show networks (probably not what you think)
        compartments   Show compartments
    pkt                Show a NDIS_PACKET structure
        pktpools       Show all allocated packet pools
        findpacket     Find a packet in memory
    vc                 Show a CoNDIS virtual connection
    af                 Show a CoNDIS address family
    ndisref            Show a debug log of refcounts
    ndisevent          Show a debug log of events
    ndisstack          Show a debug stack trace
    wdiadapter         Shows one or more WDIWiFi!CAdapter structures
    wdiminidriver      Shows one or more CMiniportDriver structures
    nwadapter          Shows one or more nwifi!ADAPT structures
    ndisrwlock         Show an NDIS_RW_LOCK_EX lock
        dbglevel       Change the debugging level [checked NDIS.sys only]
        dbgsystems     Toggle subsystems being debugged [checked NDIS.sys only]
        ndiskdversion  Show info about NDISKD itself
    netreport          Draw a box diagram of your network stack
    cxadapter          Show information about an NETADAPTER
    netqueue           Show information about a NetAdapterCx datapath queue
    nrc                Show information about an NET_RING_COLLECTION
    netring            Show information about an NET_RING
    netpacket          Show information about an NET_PACKET
    netfragment        Show information about an NET_FRAGMENT

    Alternate commands:
    miniport           Same as !ndiskd.netadapter
    gminiports         Same as !ndiskd.netadapter
    miniports          "Classic" version of !ndiskd.netadapter
```

```
    filterdb            Same as !ndiskd.netadapter  -filterdb
    offload             Same as !ndiskd.netadapter  -offloads
    ports               Same as !ndiskd.netadapter  -ports
    rcvqueues           Same as !ndiskd.netadapter  -rcvqueues
    filters             Same as !ndiskd.filter
    opens               Same as !ndiskd.mopen
    protocols           Same as !ndiskd.protocol
    nblpools            Same as !ndiskd.nblpool
    nbpools             Same as !ndiskd.nbpool
```

## См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

# !ndiskd.netadapter

Статья • 05.04.2024

Расширение **!ndiskd.netadapter** отображает сведения о минипортах NDIS или сетевых адаптерах, которые активны в системе. Если выполнить эту команду без параметров, !ndiskd отобразит список всех сетевых адаптеров.

Консоль

```
!ndiskd.netadapter [-handle <x>] [-basic] [-diag] [-state] [-bindings]
   [-ports] [-offloads] [-filterdb] [-timers] [-rst]
   [-pm] [-ss] [-aoac] [-wol] [-protocoloffloads]
   [-rss] [-hw] [-device] [-wmi] [-customwmi]
   [-ndiswmi] [-ref] [-log] [-grovel] [-findname <any>]
   [-rcvfilter] [-nicswitch] [-rcvqueues] [-nicswitches] [-iov]
   [-vfs] [-vports] [-iftrace] [-ip]
```

# Параметры

*-Обрабатывать*
Дескриптор мини-порта NDIS.

*-Основные*
Отображает сводные сведения о минипорте.

*-Рис*
Отображает оповещения автоматической диагностики (если таковые есть).

*-Государства*
Отображает текущее состояние минипорта.

*-Привязки*
Отображает минипортные привязки.

*-Порты*
Отображает список портов NDIS.

*-offloads*
Отображает состояние и возможности разгрузки задачи.

*-filterdb*
Отображает текущий фильтр пакетов.

*-Таймеры*
Отображает объекты таймера, выделенные минипортом.

*-Rst*
Отображает состояние регулирования на стороне получения.

*-pm*
Показывает состояние и возможности управления питанием.

*-Ss*
Показывает состояние выборочной приостановки.

*-aoac*
Отображает состояние AOAC (Подключение ожидания).

*-Воль*
Отображает конфигурацию пробуждения по локальной сети (WoL).

*-protocoloffloads*
Отображает разгрузки протокола активного управления питанием.

*-Rss*
Отображает параметры масштабирования на стороне получения.

*-Hw*
Отображает аппаратные ресурсы.

*-Устройства*
Отображает сведения о базовом объекте устройства NT.

*-Wmi*
Отображает идентификаторы GUID WMI, зарегистрированные в адаптере.

*-customwmi*
Отображает пользовательские идентификаторы GUID WMI, зарегистрированные мини-портом.

*-ndiswmi*
Отображает идентификаторы GUID WMI, предоставляемые NDIS.

*-Ref*
Отображает разбивку ссылок на минипорт.

*-Журнала*
Отображает журнал событий PnP и Power.

*-Унижаться*
Принудительно выполняет поиск блоков минипорта в памяти.

*-Findname*
Фильтрует мини-порты по префиксу имени.

*-rcvfilter*
Отображает возможности фильтрации.

*-nicswitch*
Показывает возможности переключения сетевых адаптеров.

*-rcvqueues*
Отображает очереди получения.

*-nicswitches*
Отображает коммутаторы сетевого адаптера.

*-Iov*
Отображает возможности SR-IOV (виртуализация единого корневого ввода-вывода).

*-Vfs*
Отображает VFS SR-IOV (виртуальные фильтры).

*-Вэкран*
Отображает виртуальные порты (виртуальные порты).

*-ifrtrace*
Отображает трассировку записывающего средства в полете.

*-ip*
Отображает IP-адреса в интерфейсе сети.

## DLL-библиотеки

Ndiskd.dll

## Примеры

Запустив **!ndiskd.netadapter** без параметров, вы можете получить список всех сетевых адаптеров в системе вместе со связанными драйверами минипорта. В этом примере выходных данных найдите сетевой адаптер отладки ядра Майкрософт, дескриптор которого — ffffdf80140c71a0. Дополнительные сведения о том, что

такое сетевой адаптер отладки ядра, см. в разделе "Отладка ядра" по сети в блоге NDIS.

Консоль

```
3: kd> !ndiskd.netadapter
    Driver             NetAdapter          Name
    ffffdf8015a98380   ffffdf8015aa11a0    Microsoft ISATAP Adapter #2
    ffffdf801418d650   ffffdf80140c71a0    Microsoft Kernel Debug Network
Adapter
```

Щелкнув дескриптор для мини-порта или введя **!ndiskd.netadapter -handle**, теперь можно увидеть все состояние NDIS на этом устройстве. Это может быть полезно очень в качестве отправного места для устранения неполадок сетевого драйвера или для выяснения того, где проблема находится в сетевом стеке. Например, можно увидеть состояние Datapath для драйвера и узнать, подключено ли оно или нет.

В нижней части отчета для этого сетевого адаптера есть много других ссылок, которые можно щелкнуть, чтобы изучить дополнительные сведения, такие как все ожидающие OID и состояние разгрузки задач. Эти ссылки соответствуют многим параметрам для **!ndiskd.netadapter**.

Консоль

```
3: kd> !ndiskd.netadapter ffffdf80140c71a0


MINIPORT

    Microsoft Kernel Debug Network Adapter

    Ndis handle        ffffdf80140c71a0
    Ndis API version   v6.20
    Adapter context    ffffdf80147d7230
    Driver             ffffdf801418d650 - kdnic  v4.2
    Network interface  ffffdf80139b3a20

    Media type         802.3
    Physical medium    NdisPhysicalMediumOther
    Device instance    ROOT\KDNIC\0000
    Device object      ffffdf80140c7050    More information
    MAC address        18-03-73-c1-e8-72


STATE

    Miniport           Running
    Device PnP         Started             Show state history
```

```
    Datapath           Normal
    Interface          Up
    Media              Connected
    Power              D0
    References         0n10                Show detail
    Total resets       0
    Pending OID        None
    Flags              NOT_BUS_MASTER, ALLOW_BUGCHECK_CALLBACK,
                       BUGCHECK_CALLBACK_REGISTERED, DEFAULT_PORT_ACTIVATED,
                       SUPPORTS_MEDIA_SENSE, DOES_NOT_DO_LOOPBACK,
                       MEDIA_CONNECTED
    PnP flags          VIRTUAL_DEVICE, HIDDEN, NO_HALT_ON_SUSPEND,
                       RECEIVED_START


BINDINGS

    Protocol list      Driver              Open               Context
    MSLLDP             ffffdf80120a5c10    ffffdf8015a749c0
ffffdf8015d325e0
    TCPIP              ffffdf80131cc010    ffffdf801494a650
ffffdf801494aa50
    NDISUIO            ffffdf8015a58140    ffffdf8015a78c10
ffffdf8015a77e00
    TCPIP6             ffffdf80131c9c10    ffffdf80147875a0
ffffdf801494f010
    (RASPPPOE)         Not running
    RSPNDR             ffffdf80120a0c10    ffffdf8015a79c10
ffffdf8015a79010
    LLTDIO             ffffdf8015a5f9b0    ffffdf801406f010
ffffdf8015a786c0
    (RDMANDK)          ffffdf801406d8f0    Declined with
NDIS_STATUS_NOT_RECOGNIZED

    Filter list        Driver              Module             Context
    WFP 802.3 MAC Layer LightWeight Filter-0000
                       ffffdf80139a5a70    ffffdf801494c670
ffffdf801494a010
    QoS Packet Scheduler-0000
                       ffffdf8014039d90    ffffdf801494dc70
ffffdf80147dc2b0
    WFP Native MAC Layer LightWeight Filter-0000
                       ffffdf80139fcd70    ffffdf8014950c70
ffffdf8014950880



MORE INFORMATION

    Driver handlers                        Task offloads
    Power management                       PM protocol offloads
    Pending OIDs                           Timers
    Pending NBLs                           Receive side throttling
    Wake-on-LAN (WoL)                      Packet filter
    Receive queues                         Receive filtering
```

```
    RSS                                     NIC switch
    Hardware resources                      Selective suspend
    NDIS ports                              WMI guids
    Diagnostic log
```

Как пример использования **!ndiskd.netadapter** в качестве начального места для дальнейшей отладки, щелкните ссылку "Обработчики драйверов" в нижней части отчета, чтобы просмотреть список всех зарегистрированных обработчиков обратного вызова драйвера для мини-порта этого мини-драйвера адаптера. В следующем примере нажатие ссылки приводит к тому, что !ndiskd запускает **расширение !ndiskd.minidriver** с дескриптором мини-порта этого мини-адаптера. Минипорт драйвер является kdnic 4.2, и его дескриптор ffffdf801418d650.

Консоль

```
3: kd> !ndiskd.minidriver ffffdf801418d650 -handlers


HANDLERS

    NDIS Handler                        Function pointer   Symbol (if
available)
    InitializeHandlerEx                 fffff80f1fd78230  bp
    SetOptionsHandler                   fffff80f1fd72800  bp
    HaltHandlerEx                       fffff80f1fd78040  bp
    ShutdownHandlerEx                   fffff80f1fd722c0  bp

    CheckForHangHandlerEx               fffff80f1fd72810  bp
    ResetHandlerEx                      fffff80f1fd72f70  bp

    PauseHandler                        fffff80f1fd78000  bp
    RestartHandler                      fffff80f1fd78940  bp

    OidRequestHandler                   fffff80f1fd71c90  bp
    CancelOidRequestHandler             fffff80f1fd722c0  bp
    DirectOidRequestHandler             [None]
    CancelDirectOidRequestHandler       [None]
    DevicePnPEventNotifyHandler         fffff80f1fd789a0  bp

    SendNetBufferListsHandler           fffff80f1fd71870  bp
    ReturnNetBufferListsHandler         fffff80f1fd71b50  bp
    CancelSendHandler                   fffff80f1fd722c0  bp
```

Теперь можно щелкнуть ссылку "bp" справа от каждого обработчика, чтобы задать точку останова для этого обработчика для отладки конкретной проблемы. Например, если в пути данных есть зависание, можно исследовать драйвер SendNetBufferListsHandler или ReturnNetBufferListsHandler.

## См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Отладка ядра по сети

**!ndiskd.minidriver**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.minidriver

16.08.2025

Команда **!ndiskd.minidriver** отображает сведения о драйвере минипорта NDIS. Если вы запускаете это расширение без параметров, !ndiskd отобразит список драйверов минипорта NDIS, которые активны в системе.

Консоль

```
!ndiskd.minidriver [-handle <x>] [-basic] [-miniports] [-devices] [-handlers]
```

## Параметры

*-ручка*
Необязательный дескриптор минипорта NDIS.

*-основной*
Отображает основные сведения о драйвере минипорта.

*-miniports*
Отображает минипорты, связанные с этим драйвером минипорта.

*-приборы*
Отображает устройства, связанные с этим мини-драйвером.

*-Обработчики*
Отображает мини-обработчики этого драйвера.

## Библиотека dll

Ndiskd.dll

## Примеры

Введите команду **!ndiskd.minidriver** без параметров, чтобы получить список всех драйверов мини-порта NDIS, активных в системе. В следующем примере найдите дескриптор адаптера kdnic, ffffd20d12dec020

Консоль

```
1: kd> !ndiskd.minidriver -basic
    ffffd20d173deae0 - tunnel
    ffffd20d12dec020 - kdnic
```

Используя дескриптор адаптера kdnic, теперь можно щелкнуть дескриптор или ввести команду **!ndiskd.minidriver -handle** , чтобы просмотреть подробные сведения о драйвере минипорта туннеля, а также список минипортов, связанных с ним.

Консоль

```
1: kd> !ndiskd.minidriver ffffd20d12dec020


MINIPORT DRIVER

    kdnic

    Ndis handle        ffffd20d12dec020    [type it]
    Driver context     fffff80d2fa15100
    DRIVER_OBJECT      ffffd20d12dee540
    Driver image       kdnic.sys
    Registry path      \REGISTRY\MACHINE\SYSTEM\ControlSet001\Services\kdnic
    Reference Count    2
    Flags              [No flags set]


MINIPORTS

    Miniport
    ffffd20d12dd71a0 - Microsoft Kernel Debug Network Adapter

    Handlers
    Device objects
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

# !ndiskd.rcvqueue

16.08.2025

Команда **!ndiskd.rcvqueue** отображает сведения о очереди получения.

Консоль

```
!ndiskd.rcvqueue -handle <x> [-filters] [-mem] [-verbose] [-rcvqueueverbosity <x>]
```

## Параметры

*-ручка*
Обязательное. Дескриптор очереди получения.

*-Фильтры*
Отображает фильтры в очереди.

*-тем*
Показывает выделение общей памяти.

*-многословный*
Отображает дополнительные сведения.

*-rcvqueueverbosity*
Уровень детализации для отображения.

## Библиотека dll

Ndiskd.dll

## Примеры

Чтобы получить дескриптор очереди получения, сначала введите команду **!ndiskd.netadapter** без параметров, чтобы просмотреть список сетевых адаптеров, их драйверов и их дескрипторов. В следующем примере найдите дескриптор NetAdapter адаптера Microsoft ISATAP #2, ffff8083e02ce1a0.

Консоль

```
3: kd> !ndiskd.netadapter
    Driver             NetAdapter          Name
```

```
    ffff8083e2668970   ffff8083e02ce1a0    Microsoft ISATAP Adapter #2
    ffff8083e210fae0   ffff8083e0f501a0    Microsoft Kernel Debug Network Adapter
```

Затем с дескриптором сетевого адаптера используйте команду **!ndiskd.netadapter -handle -rcvqueues** , чтобы получить список очередей получения для этого сетевого адаптера вместе с их дескрипторами. В этом примере существует только одна очередь получения (по умолчанию) с дескриптором ffff8083e3a3a0.

Консоль

```
3: kd> !ndiskd.netadapter ffff8083e02ce1a0 -rcvqueues


RECEIVE QUEUES

    QueueId            Queue Handle        Processor Affinity
    0 [Default]        ffff8083e3a3d3a0    0:0000000000000000 (group:mask)
                       Queue Name:         [Zero-length string]
                       VM Name:            [Zero-length string]
```

Теперь можно использовать дескриптор очереди для проверки сведений очереди получения с помощью команды **!ndiskd.rcvqueue** .

Консоль

```
3: kd> !ndiskd.rcvqueue ffff8083e3a3d3a0


RECEIVE QUEUE

    [Zero-length string]

    VM name            [Zero-length string]
    QueueId            0
    Ndis handle        ffff8083e3a3d3a0

    Miniport           ffff8083e02ce1a0 - Microsoft ISATAP Adapter #2
    Open               [No associated Open]

    Type               Unspecified
    Flags              [No flags set]
    Allocated          Yes
    References         1

    Num filters        0
    Num buffers hint   0
    MSI-X entry        0
    Lookahead size     0
    Processor affinity 0:0000000000000000 (group:mask)
```

```
    Receive filter list
    Shared memory allocations
```

## См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**!ndiskd.netadapter**

# !ndiskd.protocol

16.08.2025

Команда **!ndiskd.protocol** отображает сведения о драйвере протокола NDIS. Если вы запускаете это расширение без параметров, !ndiskd отобразит список драйверов протоколов NDIS, активных в системе.

Консоль

```
!ndiskd.protocol [-handle <x>] [-findname <any>]
```

## Параметры

*-ручка*
Необязательный дескриптор протокола NDIS.

*-findname*
Фильтрует протоколы по префиксу имени.

## Библиотека dll

Ndiskd.dll

### Примеры

Введите команду **!ndiskd.protocol** , чтобы просмотреть список всех протоколов NDIS, их дескрипторов и открыть привязки к мини-портам (при наличии). В следующем примере найдите дескриптор протокола TCPIP6TUNNEL ffff8083e1a95c00.

Консоль

```
3: kd> !ndiskd.protocol
ffff8083e0114730 - NDISUIO
  ffff8083e55f3010 - Microsoft Kernel Debug Network Adapter

ffff8083e3e90c10 - MSLLDP
  ffff8083e3926010 - Microsoft Kernel Debug Network Adapter

ffff8083e3e98c10 - WANARPV6

ffff8083e3e97010 - WANARP

ffff8083e3e8f6b0 - RSPNDR
```

```
  ffff8083e11902c0 - Microsoft Kernel Debug Network Adapter

ffff8083e3e90800 - LLTDIO
  ffff8083e15537d0 - Microsoft Kernel Debug Network Adapter

ffff8083e1a9ac10 - RDMANDK

ffff8083e1a95c00 - TCPIP6TUNNEL
  ffff8083e56b8110 - Microsoft ISATAP Adapter #2

ffff8083e19bec10 - TCPIPTUNNEL

ffff8083e19bfc10 - TCPIP6
  ffff8083e504c770 - Microsoft Kernel Debug Network Adapter

ffff8083e11cec10 - TCPIP
  ffff8083e0c565a0 - Microsoft Kernel Debug Network Adapter
```

С помощью дескриптора протокола теперь можно ввести дескриптор или ввести команду **!ndiskd.protocol -handle** , чтобы просмотреть сведения для этого протокола, например дескриптор для мини-портов, привязанных к нему.

Консоль

```
3: kd> !ndiskd.protocol ffff8083e1a95c00


PROTOCOL

    TCPIP6TUNNEL

    Ndis handle        ffff8083e1a95c00
    Ndis API version   v6.40
    Driver context     fffff80e2e4f9de0
    Driver version     v0.0
    Reference count    2
    Flags              [No flags set]
    Driver image       [Not available]     Why not?


BINDINGS

    Open               Miniport            Miniport Name
    ffff8083e56b8110   ffff8083e02ce1a0    Microsoft ISATAP Adapter #2


HANDLERS

    Protocol handler                       Function pointer   Symbol (if
available)
    BindAdapterHandlerEx                   fffff80e2e3baab0  bp
    UnbindAdapterHandlerEx                 fffff80e2e3c1c80  bp
    OpenAdapterCompleteHandlerEx           fffff80e2e4bc940  bp
```

```
    CloseAdapterCompleteHandlerEx          fffff80e2e3d19b0  bp
    NetPnPEventHandler                     fffff80e2e3bb140  bp
    UninstallHandler                       [None]
    SendNetBufferListsCompleteHandler      fffff80e2e3919a0  bp
    ReceiveNetBufferListsHandler           fffff80e2e3918a0  bp
    StatusHandlerEx                        fffff80e2e3a9550  bp
    OidRequestCompleteHandler              fffff80e2e398120  bp
    DirectOidRequestCompleteHandler        fffff80e2e398120  bp
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

# !ndiskd.mopen

Статья • 05.04.2024

Расширение **!ndiskd.mopen** отображает сведения о привязках между минипортами и протоколами. Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех открытых привязок между драйверами мини-порта NDIS и драйверами протокола.

Консоль

```
!ndiskd.mopen [-handle <x>] [-ref]
```

# Параметры

*-Обрабатывать*
Необязательный дескриптор открытой привязки NDIS.

*-Ref*
Отображает ссылки на открытые привязки.

## DLL-библиотеки

Ndiskd.dll

## Примеры

Введите команду !ndiskd.mopen, чтобы получить список всех открытых привязок. В этом примере найдите привязку между минипортом Microsoft ISATAP Adapter #2 и протоколом TCPIP6TUNNEL. Его дескриптор — ffff8083e56b8110.

Консоль

```
3: kd> !ndiskd.mopen
Open ffff8083e56b8110
  Miniport: ffff8083e02ce1a0 - Microsoft ISATAP Adapter #2
  Protocol: ffff8083e1a95c00 - TCPIP6TUNNEL

Open ffff8083e11902c0
  Miniport: ffff8083e0f501a0 - Microsoft Kernel Debug Network Adapter
  Protocol: ffff8083e3e8f6b0 - RSPNDR

Open ffff8083e55f3010
  Miniport: ffff8083e0f501a0 - Microsoft Kernel Debug Network Adapter
```

```
  Protocol: ffff8083e0114730 - NDISUIO

Open ffff8083e15537d0
  Miniport: ffff8083e0f501a0 - Microsoft Kernel Debug Network Adapter
  Protocol: ffff8083e3e90800 - LLTDIO

Open ffff8083e3926010
  Miniport: ffff8083e0f501a0 - Microsoft Kernel Debug Network Adapter
  Protocol: ffff8083e3e90c10 - MSLLDP

Open ffff8083e0c565a0
  Miniport: ffff8083e0f501a0 - Microsoft Kernel Debug Network Adapter
  Protocol: ffff8083e11cec10 - TCPIP

Open ffff8083e504c770
  Miniport: ffff8083e0f501a0 - Microsoft Kernel Debug Network Adapter
  Protocol: ffff8083e19bfc10 - TCPIP6
```

Теперь можно щелкнуть дескриптор или использовать дескриптор, чтобы ввести **команду !ndiskd.mopen -handle** , которая позволяет просмотреть дополнительные сведения об этой открытой привязке, такой как состояние Datapath и сведения о пути получения.

Консоль

```
3: kd> !ndiskd.mopen ffff8083e56b8110


OPEN

    Ndis handle        ffff8083e56b8110
    Flags              [No flags set]
    References         1                   Show detail
    Source             1
    Datapath state     Running

    Protocol           ffff8083e1a95c00 - TCPIP6TUNNEL
    Protocol context   ffff8083e15b62e0

    Miniport           ffff8083e02ce1a0 - Microsoft ISATAP Adapter #2
    Miniport context   ffff8083e0772010


RECEIVE PATH

    Packet filter      [No flags set]
    Frame Type(s)      0x86dd
```

# См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.filter

Статья • 08.07.2024

Расширение **!ndiskd.filter** отображает сведения о фильтре NDIS с легким весом (LWF). Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех LWFs.

Консоль

```
!ndiskd.filter [-handle <x>] [-findname <any>] [-handlers]
```

## Параметры

*-ручка*
Необязательный дескриптор фильтра легкого веса NDIS.

*-findname*
Фильтрует LWFs по префиксу имени.

*-Обработчики*
Отображает обработчики фильтров LWF.

## DLL-библиотеки

Ndiskd.dll

## Примеры

**Введите команду !ndiskd.filter** без параметров, чтобы получить список всех фильтров. В этом примере найдите дескриптор ffff8083e14e8460. Обратите внимание, что этот дескриптор предназначен для самого фильтра и вложен в связанный драйвер *фильтра* планировщик пакетов QoS.

Консоль

```
3: kd> !ndiskd.filter
ffff8083e1a7fd90 - QoS Packet Scheduler
  Filter ffff8083e14e8460, Miniport ffff8083e0f501a0 - Microsoft Kernel
Debug Network Adapter
ffff8083e1a96b80 - Virtual WiFi Filter Driver
ffff8083e19c4b70 - WFP vSwitch Layers LightWeight Filter
ffff8083e19a6ad0 - WFP Native MAC Layer LightWeight Filter
```

```
  Filter ffff8083e43df8f0, Miniport ffff8083e0f501a0 - Microsoft Kernel
Debug Network Adapter
ffff8083e19a6d70 - WFP 802.3 MAC Layer LightWeight Filter
  Filter ffff8083e0d89c70, Miniport ffff8083e0f501a0 - Microsoft Kernel
Debug Network Adapter
```

С помощью этого дескриптора фильтра теперь можно просмотреть более подробные сведения об этом, например состояние, дескриптор более высокого уровня фильтра и дескриптор нижнего фильтра.

Консоль

```
3: kd> !ndiskd.filter ffff8083e14e8460


FILTER

    Microsoft Kernel Debug Network Adapter-QoS Packet Scheduler-0000

    Ndis handle         ffff8083e14e8460
    Filter driver       ffff8083e1a7fd90 - QoS Packet Scheduler
    Module context      ffff8083e26953e0
    Miniport            ffff8083e0f501a0 - Microsoft Kernel Debug Network
Adapter
    Network interface   ffff8083e200f010

    State               Running
    Datapath            Send only
    References          1
    Flags               RUNNING
    More flags          OID_TOP

    Higher filter       ffff8083e0d89c70 - Microsoft Kernel Debug Network
Adapter-WFP 802.3 MAC Layer LightWeight Filter-0000
    Lower filter        ffff8083e43df8f0 - Microsoft Kernel Debug Network
Adapter-WFP Native MAC Layer LightWeight Filter-0000

    Driver handlers
```

# См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.filterdriver

16.08.2025

Расширение **!ndiskd.filterdriver** отображает сведения о драйвере фильтра NDIS. Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех драйверов фильтров.

Консоль

```
!ndiskd.filterdriver -handle <x> [-filters] [-handlers]
```

# Параметры

*-ручка*
Необязательный дескриптор драйвера фильтра NDIS.

*-Фильтры*
Отображает экземпляры фильтров этого драйвера.

*-Обработчики*
Отображает обработчики фильтров этого драйвера.

## Библиотека dll

Ndiskd.dll

## Примеры

Запустите **!ndiskd.filterdriver** без параметров, чтобы просмотреть список всех драйверов фильтров в системе. В следующем примере найдите драйвер фильтра Виртуальной WiFi, дескриптор которого — ffffbc064cc83be0.

Консоль

```
0: kd> !ndiskd.filterdriver
    ffffbc064ccd4900 - QoS Packet Scheduler
    ffffbc064cc83be0 - Virtual WiFi Filter Driver
    ffffbc064cb91a10 - WFP vSwitch Layers LightWeight Filter
    ffffbc064cb8fd70 - WFP Native MAC Layer LightWeight Filter
    ffffbc064cb59b00 - WFP 802.3 MAC Layer LightWeight Filter
```

Щелкнув дескриптор драйвера фильтра из предыдущего примера или используя его, чтобы ввести команду **!ndiskd.filterdriver -handle** в окне команд, вы можете получить более подробные сведения об этом драйвере фильтра. Например, в этом случае для этого драйвера фильтра нет модулей фильтров.

Консоль

```
0: kd> !ndiskd.filterdriver ffffbc064cc83be0


FILTER DRIVER

    Virtual WiFi Filter Driver

    Ndis handle        ffffbc064cc83be0
    Driver context     ffffbc064cc8e9d0
    Ndis API version   v6.50
    Driver version     v1.0
    Driver object      ffffbc064cc8e9d0
    Driver image       vwififlt.sys

    Bind flags         Optional, Modifying
    Class              [Zero-length string]
    References         1


FILTER MODULES

    Filter module
    [No filter modules were found]


HANDLERS

    Filter handler                          Function pointer   Symbol (if
available)
    SetOptionsHandler                       [None]
    SetFilterModuleOptionsHandler           [None]
    AttachHandler                           fffff80787d83b60  bp
    DetachHandler                           fffff80787d84800  bp
    RestartHandler                          fffff80787d86e20  bp
    PauseHandler                            fffff80787d863e0  bp
    SendNetBufferListsHandler               fffff80787d814d0  bp
    SendNetBufferListsCompleteHandler       fffff80787d81940  bp
    CancelSendNetBufferListsHandler         fffff80787d842f0  bp
    ReceiveNetBufferListsHandler            fffff80787d817e0  bp
    ReturnNetBufferListsHandler             fffff80787d81a80  bp
    OidRequestHandler                       fffff80787d85ae0  bp
    OidRequestCompleteHandler               fffff80787d85fd0  bp
    DirectOidRequestHandler                 fffff80787d84af0  bp
    DirectOidRequestCompleteHandler         fffff80787d84e80  bp
    CancelDirectOidRequestHandler           fffff80787d841d0  bp
    DevicePnPEventNotifyHandler             fffff80787d849e0  bp
```

```
    NetPnPEventHandler                      fffff80787d85a40  bp
    StatusHandler                           fffff80787d877c0  bp
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

# !ndiskd.nbl

16.08.2025

Расширение **!ndiskd.nbl** отображает сведения о структуре **NET_BUFFER_LIST** (NBL).

Консоль

```
!ndiskd.nbl [-handle <x>] [-basic] [-chain] [-info] [-data]
[-netmon] [-capfile <str>] [-launch] [-overwrite] [-log]
[-stacks] [-NblCurrentOwner]
```

## Параметры

*-ручка*
Обязательное. Адрес структуры **NET_BUFFER_LIST** .

*-основной*
Отображает основные сведения о NBL.

*-цепь*
Отображает все NBLs и **NET_BUFFER**в цепочке NBL.

*-информация*
Отображает все внеполосные сведения, связанные с NBL.

*-данные*
Отображает фактические полезные данные NBL.

*-netmon*
Просматривает цепочку NBL в Microsoft Network Monitor.

*-capfile*
Указывает путь, к которому сохраняется запись netmon.

*-баркас*
Автоматически запускается netmon.exe после сохранения файла записи.

*-перезаписать*
Разрешает перезапись файла записи, если он уже существует.

*-журнал*
Отображает журнал NBL, если включено ведение журнала NBL.

*-Стеки*
Включает вызовы с журналом NBL (используется с -log).

*-NblCurrentOwner*
Показывает текущего владельца NBL.

# Библиотека dll

Ndiskd.dll

# Примеры

В следующем примере отслеживание NBL было включено для извлечения дескриптора NBL из журнала NBL. Дополнительные сведения об отслеживании NBL и журнале NBL см. в разделе **!ndiskd.nbllog**.

Во время сбора журналов NBL в этом примере возвращается протоколом TCPIP6 в фильтре МПП native Mac Layer LightWeight.

Консоль

```
2: kd> !ndiskd.nbl ffffdf80149524a0
    NBL                ffffdf80149524a0    Next NBL           NULL
    First NB           ffffdf8014952610    Source             ffffdf80140c71a0 -
Microsoft Kernel Debug Network Adapter
    Flags              INDICATED, RETURNED, NBL_ALLOCATED, PROTOCOL_020_0,
                       PROTOCOL_200_0

    Walk the NBL chain                     Dump data payload
    Show out-of-band information
    Review NBL history
```

Щелкнув ссылку "Полезные данные дампа" из предыдущего примера или введя команду **!ndiskd.nbl -handle -data** , можно просмотреть полезные данные этого NBL. В следующем примере NBL содержит только одну **NET_BUFFER** структуру. Для дальнейшего изучения содержимого этой структуры **NET_BUFFER** выполните команду **!ndiskd.nb -handle с его дескриптором** .

Консоль

```
2: kd> !ndiskd.nbl ffffdf80149524a0 -data
NET_BUFFER ffffdf8014952610
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**NET_BUFFER_LIST**

**NET_BUFFER**

**!ndiskd.nbllog**

**!ndiskd.nb**

# !ndiskd.nb

16.08.2025

Расширение **!ndiskd.nb** отображает сведения о структуре **NET_BUFFER** (NB).

Консоль

```
!ndiskd.nb [-handle <x>] [-verbosity <x>] [-basic] [-chain] [-data]
```

## Параметры

*-ручка*
Обязательное. Адрес структуры **NET_BUFFER** .

*-многословие*
Уровень детализации для отображения.

*-основной*
Отображает основные сведения о NB.

*-цепь*
Отображает все многомерные выражения, связанные с NB.

*-данные*
Дамп фактических полезных данных NB.

## Библиотека dll

Ndiskd.dll

## Примеры

**NET_BUFFER** в следующих примерах был получен из **NET_BUFFER_LIST** в разделе "Примеры" раздела **!ndiskd.nbl**. Дескриптор NB — ffffdf8014952610.

Консоль

```
2: kd> !ndiskd.nbl ffffdf80149524a0 -data
NET_BUFFER ffffdf8014952610
```

Вы можете щелкнуть дескриптор **NET_BUFFER** или запустить команду **!ndiskd.nb -handle** , чтобы просмотреть ее сведения.

Консоль

```
2: kd> !ndiskd.nb ffffdf8014952610
    NB                ffffdf8014952610    Next NB           0
    Length            0                   Source pool       ffffdf80147e4a40
    First MDL         ffffdf8014a37930    DataOffset        0
    Current MDL       [First MDL]         Current MDL offset 0

    View associated NBL
```

Используйте команду **!ndiskd.nb -chain** , чтобы увидеть эту цепочку MDL **NET_BUFFER** в дополнение к ее основным сведениям. В следующем примере существует только один MDL. Его дескриптор — ffffdf8014a37930.

Консоль

```
2: kd> !ndiskd.nb ffffdf8014952610 -chain
    NB                ffffdf8014952610    Next NB           0
    Length            0                   Source pool       ffffdf80147e4a40
    First MDL         ffffdf8014a37930    DataOffset        0
    Current MDL       [First MDL]         Current MDL offset 0
        MDL [current]     ffffdf8014a37930    MDL Flags            c
        MappedSystemVa    ffffdf8014bf0024    ByteCount          0n1514
        Process           [System process]    ByteOffset         0n36
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**NET_BUFFER**

**NET_BUFFER_LIST**

**!ndiskd.nbl**

# !ndiskd.nblpool

16.08.2025

Расширение **!ndiskd.nblpool** отображает сведения о пуле **NET_BUFFER_LIST** (NBL). Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех выделенных пулов NBL в системе.

Консоль

```
!ndiskd.nblpool [-handle <x>] [-basic] [-allocations] [-find <str>] [-findnb
<str>]
    [-findctx <str>] [-findctxtype <str>] [-findva <x>] [-findpa <x>]
```

## Параметры

*-ручка*
Дескриптор пула NBL.

*-основной*
Отображает основные сведения о пуле NBL.

*-Распределения*
Отображает все выделенные NBL.

*-находить*
Отфильтруйте список выделенных NBL с помощью выражения отладчика.

*-findnb*
Отфильтруйте список выделенных NBL по связанным **NET_BUFFER**(NOB).

*-findctx*
Отфильтруйте список выделенных NBL по области контекста.

*-findctxtype*
Переопределите тип данных области контекста.

*-findva*
Найдите списки NB, содержащие NB, которые привязаны к заданному виртуальному адресу.

*-findpa*
Найдите списки NB, содержащие NB, которые привязаны к заданному физическому адресу.

# Библиотека dll

Ndiskd.dll

# Примеры

Введите команду **!ndiskd.nblpool** без параметров, чтобы просмотреть список всех выделенных пулов NBL. В этом примере найдите пул NBL, выделенный картой сетевого интерфейса отладчика ядра (kdnic) с тегом KDNr. Его дескриптор — ffffdf80147e4a40.

Консоль

```
2: kd> !ndiskd.nblpool
    NBL Pool           Tag                 Allocated by
    ffffdf80179b6a40   NiBP                WdNisDrv!CWFPLayer::Initialize+c6
    ffffdf8015ac6a40   EUNP                tunnel!TunnelEtherUdpGlobalInit+81
    ffffdf8015a78040   Nuio                ndisuio!ndisuioCreateBinding+15f
    ffffdf8015a77800   Nuio                ndisuio!ndisuioCreateBinding+13c
    ffffdf8015a63040   BaNB                rspndr!TopStartNetBufferModule+6d
    ffffdf8015a68a40   LLnb                mslldp!lldpProtSetOptions+49
    ffffdf8014654040   BaNB                lltdio!TopStartNetBufferModule+6d
    ffffdf801494ca40   Pcsb                pacer!PcFilterAttach+142
    ffffdf80147e4a40   KDNr                kdnic!NICAllocAdapter+178
    ffffdf80131ce040   bnvW                wfplwfs!DriverEntry+7a0
    ffffdf80139ffa40   Wfdp                wfplwfs!WfpRioInitialize+a4
    ffffdf8012061200   UNbl
NETIO!NetioAllocateNetBufferListNetBufferMdlAndDataPool+49
    ffffdf8013968a40   TcDN
NETIO!NetioAllocateNetBufferListNetBufferMdlAndDataPool+49
    ffffdf8013969a40   TNbl
NETIO!NetioAllocateNetBufferListNetBufferMdlAndDataPool+49
    ffffdf801397c040   StBn                NETIO!StreamPoolsInit+c1
    ffffdf8013088040   Wfra                NETIO!WfpNblInfoLibraryInit+b8
    ffffdf8012067440   Nnnn
NETIO!NetioInitializeNetBufferListLibrary+13e
    ffffdf8012067a40   Nnbl
NETIO!NetioInitializeNetBufferListLibrary+112
    ffffdf80131caa40   NDrt                ndis!ndisInitializePeriodicReceives+22f
    ffffdf80131d5a40   NDnd                ndis!DriverEntry+5e9
```

Щелкните дескриптор пула NBL или введите команду **!ndiskd.nblpool -handle** , чтобы проверить ее сведения.

Консоль

```
2: kd> !ndiskd.nblpool ffffdf80147e4a40


NBL POOL
```

```
    Ndis handle        ffffdf80147e4a40
    Allocation tag     KDNr
    Owner
    Allocated by       kdnic!NICAllocAdapter+178

    Flags              CONTAINS_NET_BUFFER
    Structure size     0n544
    Context size       0
    Data size          0

    All allocated NBLs
```

Чтобы изучить NBL, содержащиеся в этом пуле NBL, щелкните ссылку "Все выделенные NBLs" в нижней части. Кроме того, можно ввести команду **!ndiskd.nblpool -handle -allocations** . Как показано в следующем примере, этот пул NBL содержит более 1024 NBLs, чтобы !ndiskd выйти рано. Вы можете использовать параметр -force для обхода этого ограничения и просмотра всех NBL в этом пуле NBL.

Консоль

```
2: kd> !ndiskd.nblpool ffffdf80147e4a40 -allocations


ALL ALLOCATED NBLs

    NBL                Active?
    ffffdf8014951940   Allocated
    ffffdf8014951b90   Allocated
    ffffdf8014951de0   Allocated
    ffffdf8014951030   Allocated
    ffffdf80149524a0   Allocated
    ffffdf80149526f0   Allocated
    ffffdf8014952940   Allocated
    ffffdf8014952b90   Allocated
    ffffdf8014952de0   Allocated
    ffffdf8014952030   Allocated
    ffffdf80149534a0   Allocated
    ffffdf80149536f0   Allocated
    ffffdf8014953940   Allocated
    ffffdf8014953b90   Allocated
    ffffdf8014953de0   Allocated
    ffffdf8014953030   Allocated
    ffffdf80149544a0   Allocated
    ffffdf80149546f0   Allocated
    ffffdf8014954940   Allocated

...

    ffffdf80148b0b90   Allocated
    ffffdf80148b0de0   Allocated
    ffffdf80148b0030   Allocated
```

```
    ffffdf80148b14a0   Allocated
    ffffdf80148b16f0   Allocated
    ffffdf80148b1940   Allocated
    ffffdf80148b1b90   Allocated
    ffffdf80148b1de0   Allocated
    ffffdf80148b1030   Allocated
    [Maximum of 1024 items read; quitting early. Rerun with the '-force' option
    to bypass this limit.]
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**NET_BUFFER_LIST**

**NET_BUFFER**

# !ndiskd.nbpool

16.08.2025

Расширение **!ndiskd.nbpool** отображает сведения о пуле **NET_BUFFER** (NB). Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех выделенных пулов NB в системе.

Консоль

```
!ndiskd.nbpool [-handle <x>] [-allocations] [-find <str>] [-findva <x>] [-findpa
<x>]
```

## Параметры

*-ручка*
Дескриптор пула NB.

*-Распределения*
Отображает все выделенные NB.

*-находить*
Отфильтруйте список выделенных NOB с помощью выражения отладчика.

*-findva*
Найдите NBS, которые перестраив заданный виртуальный адрес.

*-findpa*
Найдите NBS, которые перестраив заданный физический адрес.

## Библиотека dll

Ndiskd.dll

## Примеры

Введите команду **!ndiskd.nbpool** без параметров, чтобы просмотреть список всех выделенных пулов NB. В этом примере найдите пул NB, выделенный службой Netio с тегом Nnbf. Его дескриптор — ffffdf801308ca40.

Консоль

```
2: kd> !ndiskd.nbpool
    NB Pool            Tag                 Allocated by
    ffffdf8013963a40   UDNb
NETIO!NetioAllocateNetBufferMdlAndDataPool+3c
    ffffdf801396aa40   TSNb
NETIO!NetioAllocateNetBufferMdlAndDataPool+3c
    ffffdf801397d4c0   StBn                NETIO!StreamPoolsInit+90
    ffffdf801308ca40   Nnbf
NETIO!NetioInitializeNetBufferListLibrary+dd
    ffffdf80131cba40   NDnd                ndis!DriverEntry+615
```

Щелкните дескриптор пула NB или введите команду **!ndiskd.nbpool -handle** , чтобы проверить ее сведения.

Консоль

```
2: kd> !ndiskd.nbpool ffffdf801308ca40


NB POOL

    Ndis handle        ffffdf801308ca40
    Allocation tag     Nnbf
    Owner
    Allocated by       NETIO!NetioInitializeNetBufferListLibrary+dd

    Flags              [No flags set]
    Structure size     0n176
    Data size          0

    All allocated NBs
```

Чтобы изучить NBS, содержащиеся в этом пуле NB, щелкните ссылку "Все выделенные NBS" в нижней части экрана. Кроме того, можно ввести команду **!ndiskd.nbpool -handle -allocations** . Как показано в следующем примере, этот пул NB содержит более 1024 NBS, чтобы !ndiskd выйти рано. Вы можете использовать параметр -force для обхода этого ограничения и просмотра всех NBS в этом пуле NB.

Консоль

```
2: kd> !ndiskd.nbpool ffffdf801308ca40 -allocations


ALL ALLOCATED NBs

    NB                 Active?
    ffffdf8016ea4360   Allocated
    ffffdf801744df50   Allocated
    ffffdf8016932860   Allocated
```

```
    ffffdf8016e31500   Allocated
    ffffdf80174eade0   Allocated
    ffffdf8017daa900   Allocated
    ffffdf8017c8c680   Allocated
    ffffdf80166b23b0   Allocated
    ffffdf80164fea70   Allocated
    ffffdf8012845990   Allocated
    ffffdf8017d692d0   Allocated
    ffffdf8017cdc090   Allocated
    ffffdf8012771780   Allocated
    ffffdf80158a3550   Allocated
    ffffdf8012eef5c0   Allocated
    ffffdf80127719d0   Allocated
    ffffdf8015119570   Allocated
    ffffdf8012e18d40   Allocated
    ffffdf8017929b10   Allocated
    ffffdf8016d4e430   Allocated

...

    ffffdf8015ffbbd0   Allocated
    ffffdf8015ec1b10   Freed
    ffffdf80158e56d0   Allocated
    ffffdf8016272110   Freed
    ffffdf8015d8e030   Freed
    ffffdf8015d8e770   Freed
    ffffdf80158ddc30   Freed
    ffffdf801584acc0   Freed
    ffffdf8015846b40   Freed
    ffffdf8015a06c50   Freed
    ffffdf801480c300   Freed
    ffffdf8015e48f50   Freed
    ffffdf8015de64e0   Freed
    ffffdf8015ddff50   Freed
    [Maximum of 1024 items read; quitting early. Rerun with the '-force' option
    to bypass this limit.]
```

## См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**NET_BUFFER**

# !ndiskd.pendingnbls

16.08.2025

Расширение **!ndiskd.pendingnbls** отображает ожидающие NBLs (**NET_BUFFER_LISTs**), передаваемые.

Консоль

```
!ndiskd.pendingnbls [-handle <x>] [-fullstack] [-verbosity <x>]
```

# Параметры

*-ручка*
Дескриптор минипорта, фильтра или открытия NDIS.

*-fullstack*
Отображает ожидающие NBL из всего стека, связанного с дескриптором.

*-многословие*
Уровень детализации для отображения.

## Библиотека dll

Ndiskd.dll

## Примеры

**!ndiskd.pendingnbls** можно передать дескриптор минипорта NDIS, фильтра или открытия. В следующем ряде примеров используется минипорт-дескриптор. Чтобы просмотреть список всех мини-портов и связанных с ними мини-накопителей, запустите расширение **!ndiskd.netadapter** без параметров. В следующем примере выходных данных найдите сетевой адаптер отладки ядра Майкрософт, дескриптор которого ffffe00bc3f701a0. Его дескриптор мини-driver является ffffe00bc51b9ae0.

Консоль

```
0: kd> !ndiskd.netadapter
    Driver             NetAdapter         Name
    ffffe00bc6e12ae0   ffffe00bc6e4e1a0   Microsoft ISATAP Adapter #2
    ffffe00bc51b9ae0   ffffe00bc3f701a0   Microsoft Kernel Debug Network Adapter
```

Чтобы просмотреть ожидающие NBLs для мини-порта, установите точку останова в его minidriver SendNetBufferListsHandler. Используйте дескриптор **minidriver для запуска команды !ndiskd.minidriver -handle -handlers** , чтобы просмотреть список его обработчиков, а затем щелкните ссылку "bp" справа от SendNetBufferListsHandler. Можно также ввести команду **bp-handle** в командной строке.

Консоль

```
0: kd> !ndiskd.minidriver ffffe00bc51b9ae0 -handlers


HANDLERS

    NDIS Handler                            Function pointer   Symbol (if
available)
    InitializeHandlerEx                     fffff80ae9618230  bp
    SetOptionsHandler                       fffff80ae9612800  bp
    HaltHandlerEx                           fffff80ae9618040  bp
    ShutdownHandlerEx                       fffff80ae96122c0  bp

    CheckForHangHandlerEx                   fffff80ae9612810  bp
    ResetHandlerEx                          fffff80ae9612f70  bp

    PauseHandler                            fffff80ae9618000  bp
    RestartHandler                          fffff80ae9618940  bp

    OidRequestHandler                       fffff80ae9611c90  bp
    CancelOidRequestHandler                 fffff80ae96122c0  bp
    DirectOidRequestHandler                 [None]
    CancelDirectOidRequestHandler           [None]
    DevicePnPEventNotifyHandler             fffff80ae96189a0  bp

    SendNetBufferListsHandler               fffff80ae9611870  bp
    ReturnNetBufferListsHandler             fffff80ae9611b50  bp
    CancelSendHandler                       fffff80ae96122c0  bp
```

Установив точку останова в SendNetBufferListsHandler, введите команду **g** , чтобы позволить целевому компьютеру отладки запускаться и нажать точку останова.

Консоль

```
0: kd> bp fffff80ae9611870
0: kd> g
Breakpoint 0 hit
fffff80a`e9611870 4053            push    rbx
```

Теперь, после нажатия точки останова minidriver SendNetBufferListsHandler, вы можете увидеть любые ожидающие NBLs для мини-порта, введя команду **!ndiskd.pendingnbls -handle** с дескриптором минипорта.

**Заметка** Целевой компьютер отладки в этом примере загружал веб-страницу при попадании в точку останова, поэтому трафик проходит через путь к данным мини-порта. Таким образом, он имел ожидающий NBL для отправки. Даже после установки точки останова на одном или нескольких обработчиках NBL для мини-driver вы можете не видеть ожидающие NBL, если в пути данных не было действий.

Консоль

```
0: kd> !ndiskd.pendingnbls ffffe00bc3f701a0

PHASE 1/3: Found 20 NBL pool(s).
PHASE 2/3: Found 342 freed NBL(s).

    Pending Nbl        Currently held by
    ffffe00bc5545c60   ffffe00bc3f701a0 - Microsoft Kernel Debug Network Adapter
[NetAdapter]


PHASE 3/3: Found 1 pending NBL(s) of 4817 total NBL(s).
Search complete.
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**NET_BUFFER_LIST**

**!ndiskd.netadapter**

**!ndiskd.minidriver**

**bp, bu, bm (Установка точки останова)**

# !ndiskd.nbllog

Статья • 05.04.2024

Расширение **!ndiskd.nbllog** отображает журнал всех действий NBL (NET_BUFFER_LIST) в системе.

Консоль

```
!ndiskd.nbllog [-stacks]
```

## Параметры

*-Стеки*
Включите вызовы.

## DLL-библиотеки

Ndiskd.dll

## Замечания

**Важно!ndiskd.nbllog** требует включения отслеживания NBL на целевом компьютере отладочного компьютера. Отслеживание NBL по умолчанию не включено во всех конфигурациях Windows. Если отслеживание NBL не включено, !ndiskd предоставит инструкции по его включению, как показано в следующем фрагменте кода.

Консоль

```
0: kd> !ndiskd.nbllog
    This command requires NBL tracking to be enabled on the debugee target
    machine.  (By default, client operating systems have level 1, and
servers
    have level 0).  To enable, set this REG_DWORD value to a nonzero value
on
    the target machine and reboot the target machine:

    HKLM\SYSTEM\CurrentControlSet\Services\NDIS\Parameters ! TrackNblOwner
    Possible Values (features are cumulative)
    * 0:  Disable all tracking.
    * 1:  Track the most recent owner of each NBL (enables
!ndiskd.pendingnbls)
    * 2:  Scan for leaks at runtime (use with StuckNblReaction)
```

```
    * 3:  Keep a full history of all activity (enables !ndiskd.nbl -log)
    * 4:  Take stack capture snapshots (enables !ndiskd.nbl -log -stacks)
    This command requires level 3 or higher.
```

В журнале NBL отображается сетевой трафик в системе. **!ndiskd.netreport** анализирует журнал отслеживания NBL для визуального отображения этого сетевого трафика. Таким образом, если отслеживание NBL не включено, **!ndiskd.netreport** не сможет отобразить эти сведения.

# Примеры

После включения отслеживания NBL на целевом компьютере отладочного компьютера введите **команду !ndiskd.nbllog** , чтобы просмотреть журнал всего трафика NBL в системе. Как показано в приведенном ниже примере, запуск **!ndiskd.nbllog** без параметров не ограничивает выходные данные до 200 событий, которые можно обойти, повторно выполнив команду с параметром *-force* . Середина выходных данных в этом примере была обрезана для краткости.

Консоль

```
0: kd> !ndiskd.nbllog
    NBLs               Processor           Event              Detail

    ffffe00bc71453f0   CPU  0              Freed
    ffffe00bc7163b40   CPU  2              Allocated
    ffffe00bc7163b40   CPU  2              ProtocolSent
ffffe00bc5ac4880 - QoS Packet Scheduler-0000
    ffffe00bc7163b40   CPU  2              FilterSent
ffffe00bc5ac5c70 - WFP Native MAC Layer LightWeight Filter-0000
    ffffe00bc7163b40   CPU  2, IRQL=DPC    FilterSent
ffffe00bc3f701a0 - Microsoft Kernel Debug Network Adapter
    ffffe00bc7163b40   CPU  2, IRQL=DPC    SentToMiniport
ffffe00bc3f701a0 - Microsoft Kernel Debug Network Adapter
    ffffe00bc7163b40   CPU  0, IRQL=DPC    MiniportSendCompleted
ffffe00bc5ac5c70 - WFP Native MAC Layer LightWeight Filter-0000
    ffffe00bc7163b40   CPU  0, IRQL=DPC    FilterSendCompleted
ffffe00bc5ac4880 - QoS Packet Scheduler-0000
    ffffe00bc7163b40   CPU  0, IRQL=DPC    FilterSendCompleted send complete
in NDIS, sorting to Opens
    ffffe00bc7163b40   CPU  0, IRQL=DPC    SendCompleted
ffffe00bc5ab7c10 - TCPIP6

...

    ffffe00bc6b469b0   CPU  2              Allocated
    ffffe00bc6b469b0   CPU  2              Freed
    ffffe00bc64a3690   CPU  2              Allocated
    ffffe00bc64a3690   CPU  2              ProtocolSent
ffffe00bc5ac4880 - QoS Packet Scheduler-0000
```

```
    ffffe00bc64a3690   CPU  2              FilterSent
ffffe00bc5ac5c70 - WFP Native MAC Layer LightWeight Filter-0000
    ffffe00bc64a3690   CPU  2, IRQL=DPC    FilterSent
ffffe00bc3f701a0 - Microsoft Kernel Debug Network Adapter
    ffffe00bc64a3690   CPU  2, IRQL=DPC    SentToMiniport
ffffe00bc3f701a0 - Microsoft Kernel Debug Network Adapter
    ffffe00bc3cf2d10   CPU  1              Allocated
    ffffe00bc7bc6030   CPU  1              Allocated
    ffffe00bc3cf2d10   CPU  1              ProtocolSent
ffffe00bc5ac4880 - QoS Packet Scheduler-0000

    Maximum of 200 events printed; quitting early.
    Rerun with the '-force' option to bypass this limit.
```

Более подробное описание интерпретации результатов **!ndiskd.nbllog** см . в блоге !ndiskd.nbl -log в блоге NDIS.

## См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**NET_BUFFER_LIST**

!ndiskd.nbl -log

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.oid

Статья • 05.04.2024

Расширение **!ndiskd.oid** отображает сведения о запросе NDIS OID. Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех ожидающих запросов OID на всех минипортах и фильтрах. Каждый минипорт или фильтр имеет по крайней мере один ожидающий запрос OID и любое количество очередных запросов OID.

Обратите внимание, что фильтры обычно клонировать запросы OID и передают клонирование вниз. Это означает, что даже если протокол выдает один запрос OID, может быть несколько экземпляров клонированных запросов: по одному в каждом фильтре и другом в мини-порте. **!ndiskd.oid** будет отображать каждый клон отдельно, поэтому вы можете увидеть больше ожидающих OID, чем протокол фактически выдан.

Консоль

```
!ndiskd.oid [-handle <x>] [-legacyoid] [-nolimit>] [-miniport <x>]
```

# Параметры

*-Обрабатывать*
Дескриптор NDIS_OID_REQUEST

*-legacyoid*
Рассматривается как устаревшая NDIS_REQUEST вместо NDIS_OID_REQUEST.

*-nolimit*
Не ограничивает количество отображаемых OID.

*-Минипорта*
Находит ожидающие запросы OID в стеке мини-порта.

## DLL-библиотеки

Ndiskd.dll

## Замечания

**!ndiskd.oid** показывает список всех ожидающих OID в системе за раз, поэтому это может быть полезно в отладке системы зависания или 0x9F ошибок проверка ситуациях (DRIVER_POWER_STATE_FAILURE). Например, предположим, что анализ вымышленной ошибки 0x9F проверка показал, что система зависла на IRP и ждала NDIS. В NDIS irPs из ОС преобразуются в OID, включая переходы питания, поэтому, выполнив **!ndiskd.oid** , можно увидеть, что в этом примере устройство в нижней части стека, возможно, цепляется к OID_PNP_SET_POWER и зависает остальную часть стека. Драйверы NDIS не должны перо OID в течение более одной секунды, поэтому вы могли бы изучить, почему это устройство держало OID в ожидании слишком долго, чтобы попытаться решить проблему.

## Примеры

Чтобы просмотреть пример ожидающих OIDS в системе, работающей обычно, задайте точку останова для подпрограммы обработчика запросов OID мини-порта (в соответствующем драйвере минипорта). Сначала выполните **команду !ndiskd.minidriver** без параметров, чтобы получить список драйверов минипорта в системе. В этом примере выходных данных найдите дескриптор kdnic minidriver, ffffdf801418d650..

Консоль

```
3: kd> !ndiskd.minidriver
    ffffdf8015a98380 - tunnel
    ffffdf801418d650 - kdnic
```

Щелкните дескриптор мини-driver, а затем щелкните ссылку "Обработчики" в нижней части страницы сведений, чтобы просмотреть список обработчиков. Можно также ввести **команду !ndiskd.minidriver -handle -handlers** . После получения списка обработчиков мини-driver найдите обработчик OidRequestHandler, дескриптор которого fffff80f1fd71c90 в этом примере.

Консоль

```
2: kd> !ndiskd.minidriver ffffdf801418d650 -handlers


HANDLERS

    NDIS Handler                            Function pointer   Symbol (if
available)
    InitializeHandlerEx                     fffff80f1fd78230  bp
    SetOptionsHandler                       fffff80f1fd72800  bp
    HaltHandlerEx                           fffff80f1fd78040  bp
    ShutdownHandlerEx                       fffff80f1fd722c0  bp
```

```
    CheckForHangHandlerEx                   fffff80f1fd72810  bp
    ResetHandlerEx                          fffff80f1fd72f70  bp

    PauseHandler                            fffff80f1fd78000  bp
    RestartHandler                          fffff80f1fd78940  bp

    OidRequestHandler                       fffff80f1fd71c90  bp
    CancelOidRequestHandler                 fffff80f1fd722c0  bp
    DirectOidRequestHandler                 [None]
    CancelDirectOidRequestHandler           [None]
    DevicePnPEventNotifyHandler             fffff80f1fd789a0  bp

    SendNetBufferListsHandler               fffff80f1fd71870  bp
    ReturnNetBufferListsHandler             fffff80f1fd71b50  bp
    CancelSendHandler                       fffff80f1fd722c0  bp
```

Теперь щелкните ссылку "bp" справа от OidRequestHandler или введите **команду bp -handle с его дескриптором** , чтобы задать точку останова для этой подпрограммы. Затем введите **команду g** , чтобы разрешить целевому компьютеру отладки запускаться и нажимать точку останова, которой вы только что задали.

Консоль

```
2: kd> bp fffff80f1fd71c90
2: kd> g
Breakpoint 1 hit
fffff80f`1fd71c90 448b4204        mov     r8d,dword ptr [rdx+4]
```

После активации точки останова в подпрограмме обработчика запросов OID мини-driver, как показано в предыдущем примере, можно запустить команду !ndiskd.oid, чтобы просмотреть список всех ожидающих OID в системе.

Консоль

```
1: kd> !ndiskd.oid


ALL PENDING OIDs

    NetAdapter         ffffdf80140c71a0 - Microsoft Kernel Debug Network
Adapter
        Current OID        OID_GEN_STATISTICS
    Filter             ffffdf8014950c70 - Microsoft Kernel Debug Network
Adapter-WFP Native MAC Layer LightWeight Filter-0000
        Current OID        OID_GEN_STATISTICS
    Filter             ffffdf801494dc70 - Microsoft Kernel Debug Network
Adapter-QoS Packet Scheduler-0000
        Current OID        OID_GEN_STATISTICS
```

В этом примере ожидается OID_GEN_STATISTICS OID. При просмотре результатов !ndiskd.oid, помните, что фильтры клонируют запросы OID и передают их по стеку, а OID обычно передаются из фильтра для фильтрации в мини-порт. Таким образом, хотя в этом примере может выглядеть три отдельных запроса OID с одинаковым именем, в этом примере происходит одна логическая операция, которая была физически распределена по 3 OID и на 3 драйверах.

## См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

0x9F ошибки проверка

OID_PNP_SET_POWER

**bp, bu, bm (Установка точки останова)**

OID_GEN_STATISTICS

NDIS OIDs

Интерфейс запроса NDIS OID

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.interfaces

16.08.2025

Расширение **!ndiskd.interfaces** отображает сведения о сетевом интерфейсе. Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех сетевых интерфейсов.

Дополнительные сведения о сетевых интерфейсах см. в разделе сетевых интерфейсов NDIS.

Консоль

```
!ndiskd.interfaces -handle <x> [-luid <x>]
```

## Параметры

*-ручка*
Необязательный дескриптор сетевого интерфейса.

*-luid*
NetLuid (локальный уникальный идентификатор) сетевого интерфейса.

## Библиотека dll

Ndiskd.dll

## Примеры

Запустите расширение **!ndiskd.interfaces** без параметров, чтобы просмотреть список всех сетевых интерфейсов в системе. В этом примере найдите интерфейс сетевого подключения Intel(R) 82579LM Gigabit. Его дескриптор — ffffdf80139f8a20.

Консоль

```
1: kd> !ndiskd.interfaces
    Interface
    ffffdf801494fa20 - Microsoft Kernel Debug Network Adapter-WFP Native MAC Layer
LightWeight Filter-0000
    ffffdf801494c010 - Microsoft Kernel Debug Network Adapter-QoS Packet
Scheduler-0000
    ffffdf801494ba20 - Microsoft Kernel Debug Network Adapter-WFP 802.3 MAC Layer
LightWeight Filter-0000
    ffffdf80139b3a20 - Microsoft Kernel Debug Network Adapter
```

```
    ffffdf80139f8a20 - Intel(R) 82579LM Gigabit Network Connection
    ffffdf80139baa20 - WAN Miniport (IP)
    ffffdf80139a4a20 - WAN Miniport (IPv6)
    ffffdf80131d0010 - WAN Miniport (Network Monitor)
    ffffdf80131cda20 - WAN Miniport (PPPOE)
    ffffdf80139b6a20 - Software Loopback Interface 1
    ffffdf80139b0a20 - Microsoft ISATAP Adapter
    ffffdf80139ada20 - WAN Miniport (SSTP)
    ffffdf80131cf010 - WAN Miniport (IKEv2)
    ffffdf80139fea20 - WAN Miniport (L2TP)
    ffffdf80139a7a20 - WAN Miniport (PPTP)
    ffffdf80139aaa20 - Microsoft ISATAP Adapter #2
    ffffdf80139fba20 - Teredo Tunneling Pseudo-Interface
```

Щелкнув дескриптор интерфейса или введя команду **!ndiskd.interfaces -handle** , вы можете просмотреть сведения об этом интерфейсе, включая сведения об идентификаторе и его текущем состоянии. В этом примере видно, что сетевое подключение Intel(R) 82579LM Gigabit является подключением Ethernet (его ifAlias) и находится в состоянии MediaConnectUnknown для подключения (так как оно зарезервировано для использования отладчиком ядра Windows).

Консоль

```
1: kd> !ndiskd.interfaces ffffdf80139f8a20


INTERFACE

    Ethernet

    Ndis handle        ffffdf80139f8a20
    IfProvider         ffffdf80131ca8d0 - The NDIS interface provider
    Provider context   ffffdf80139f8a20

    ifType             IF_TYPE_ETHERNET_CSMACD
    Media type         802.3
    Physical medium    802.3
    Access type        BROADCAST
    Direction type     SEND_AND_RECEIVE
    Connection type    DEDICATED

    ifConnectorPresent No

    Network            ffffdf80139b8900 - [Unnamed network]
    Compartment        ffffdf80139b9940 - Compartment #1


IDENTIFIERS

    ifAlias            Ethernet
    ifDescr            Intel(R) 82579LM Gigabit Network Connection
    ifName (NET_LUID)  06:8001
```

```
    ifPhysAddress      18-03-73-c1-e8-72

    ifIndex            0n14
    ifGuid             cbddfde1-5570-4c65-9d47-52d63abce00c


STATE

    Connected          MediaConnectUnknown
    ifOperStatus       NOT_PRESENT

    Link speed         0 [Speed is not applicable]
    ifMtu              0
    Duplex             UnknownDuplex

    Refer to RFC 2863 for definitions of many of these terms
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Сетевые интерфейсы NDIS

значение NET_LUID

# !ndiskd.ifprovider

16.08.2025

Расширение **!ndiskd.ifprovider** отображает сведения о поставщике интерфейса NDIS (IfProvider). Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех зарегистрированных поставщиков интерфейсов NDIS.

Консоль

```
!ndiskd.ifprovider [-handle <x>]
```

# Параметры

*-ручка*
Необязательный дескриптор IfProvider.

## Библиотека dll

Ndiskd.dll

## Примеры

Запустите расширение **!ndiskd.ifprovider** без параметров, чтобы получить список всех зарегистрированных IfProviders.

Консоль

```
1: kd> !ndiskd.ifprovider
    IfProvider
    ffffd20d14334180 - wanarp
    ffffd20d1264a950 - wfplwfs
    ffffd20d11deae00 - The NDIS loopback provider
    ffffd20d11deae70 - The NDIS interface provider
```

В предыдущем примере видно, что на компьютере отладки зарегистрировано четыре поставщика интерфейса. Два из них — поставщики интерфейсов NDIS.

**Заметка** Поставщики интерфейсов являются универсальной концепцией и не требуются для мини-портов драйверов. Хотя минипорт-драйвер может зарегистрировать себя в качестве поставщика интерфейса при необходимости, большинство драйверов минипорта не делают этого, так как NDIS имеет встроенный поставщик интерфейса.

Встроенный поставщик интерфейсов NDIS автоматически предоставляет интерфейсы для каждого мини-драйвера, каждого модуля Light-Weight filter (LWF) и интерфейса обратного цикла. Дополнительные сведения см. в разделе о поставщике интерфейса NDIS.

В следующем примере показаны сведения о поставщике интерфейса wanarp в предыдущем примере, дескриптор которого имеет значение ffffd20d14334180.

Консоль

```
1: kd> !ndiskd.ifprovider ffffd20d14334180


IF PROVIDER

    wanarp
    Ndis handle        ffffd20d14334180


INTERFACES

    Interface
    [No interfaces found]


HANDLERS

    Protocol handler                      Function pointer   Symbol (if
available)
    QueryObjectHandler                    fffff80d2f0414b0  bp
wanarp!WanNdisIfQueryHandler
    SetObjectHandler                      fffff80d2f04bd10  bp
wanarp!WanNdisIfSetHandler
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Регистрация в качестве поставщика интерфейса

# !ndiskd.ifstacktable

16.08.2025

Расширение **!ndiskd.ifstacktable** отображает таблицу стека сетевого интерфейса (ifStackTable).

Дополнительные сведения о таблице стека интерфейса см. в разделе "Обслуживание стека сетевых интерфейсов".

Консоль

```
!ndiskd.ifstacktable
```

## Параметры

Это расширение не имеет параметров.

## Библиотека dll

Ndiskd.dll

## Примеры

Выполните команду **!ndiskd.ifstacktable** , чтобы просмотреть ifStackTable.

Консоль

```
3: kd> !ndiskd.ifstacktable


INTERFACE STACK TABLE

    Lower interface    Lower IfIndex       Higher IfIndex     Higher interface
    ffffdf80139b3a20   6                   15                 ffffdf801494fa20
    ffffdf801494fa20   15                  16                 ffffdf801494c010
    ffffdf801494c010   16                  17                 ffffdf801494ba20
```

NDIS поддерживает таблицу стека для адаптеров минипорта NDIS, промежуточных драйверов NDIS 5.x и модулей фильтров NDIS, в то время как драйверы NDIS MUX промежуточного драйвера требуются для указания внутренней связи между интерфейсом виртуального минипорта и нижним интерфейсом протокола. Таким

образом, ifStackTable может быть полезным для просмотра связей стека интерфейса в системе с более сложными драйверами MUX, установленными.

Так как в этой примере системы нет драйверов NDIS MUX Промежуточного уровня, в ifStackTable отображаются только связи стека, предоставляемые NDIS. В следующем примере щелкните дескриптор нижнего интерфейса третьей строки (дескриптор ffffdf801494c010, Lower IfIndex 16) показывает интерфейс планировщика пакетов QoS.

Консоль

```
3: kd> !ndiskd.interface ffffdf801494c010


INTERFACE

    [Zero-length string]

    Ndis handle        ffffdf801494c010
    IfProvider         ffffdf80131ca8d0 - The NDIS interface provider
    NDIS filter        ffffdf801494dc70 - Microsoft Kernel Debug Network Adapter-
QoS Packet Scheduler-0000

    ifType             IF_TYPE_ETHERNET_CSMACD
    Media type         802.3
    Physical medium    NdisPhysicalMediumOther
    Access type        BROADCAST
    Direction type     SEND_AND_RECEIVE
    Connection type    DEDICATED

    ifConnectorPresent No

    Network            ffffdf80139b8900 - [Unnamed network]
    Compartment        ffffdf80139b9940 - Compartment #1


IDENTIFIERS

    ifAlias            [Zero-length string]
    ifDescr            Microsoft Kernel Debug Network Adapter-QoS Packet
Scheduler-0000
    ifName (NET_LUID)  06:01
    ifPhysAddress      18-03-73-c1-e8-72

    ifIndex            0n16
    ifGuid             fc2a0ae1-b103-11e6-b724-806e6f6e6963


STATE

    Connected          Connected
    ifOperStatus       DORMANT
    ifOperStatusFlags  DORMANT_PAUSED
```

```
    Link speed         1000000000 (1 Gbps)
    ifMtu              0n1500
    Duplex             FullDuplex

    Refer to RFC 2863 for definitions of many of these terms
```

Продолжая тот же пример, щелкнув дескриптор для интерфейса "Выше" третьей строки (дескриптор ffffdf801494ba20, Higher IfIndex 17) отображает интерфейс для фильтра полувеса УРОВНЯ MAC 802.3.

Консоль

```
3: kd> !ndiskd.interface ffffdf801494ba20


INTERFACE

    [Zero-length string]

    Ndis handle        ffffdf801494ba20    [type it]
    IfProvider         ffffdf80131ca8d0 - The NDIS interface provider
    NDIS filter        ffffdf801494c670 - Microsoft Kernel Debug Network Adapter-
WFP 802.3 MAC Layer LightWeight Filter-0000

    ifType             IF_TYPE_ETHERNET_CSMACD
    Media type         802.3
    Physical medium    NdisPhysicalMediumOther
    Access type        BROADCAST
    Direction type     SEND_AND_RECEIVE
    Connection type    DEDICATED

    ifConnectorPresent No

    Network            ffffdf80139b8900 - [Unnamed network]
    Compartment        ffffdf80139b9940 - Compartment #1


IDENTIFIERS

    ifAlias            [Zero-length string]
    ifDescr            Microsoft Kernel Debug Network Adapter-WFP 802.3 MAC Layer
LightWeight Filter-0000
    ifName (NET_LUID)  06:02
    ifPhysAddress      18-03-73-c1-e8-72

    ifIndex            0n17
    ifGuid             fc2a0ae0-b103-11e6-b724-806e6f6e6963


STATE

    Connected          Connected
```

```
    ifOperStatus       DORMANT
    ifOperStatusFlags  DORMANT_PAUSED

    Link speed         1000000000 (1 Gbps)
    ifMtu              0n1500
    Duplex             FullDuplex

    Refer to RFC 2863 for definitions of many of these terms
```

В этом примере показано, что фильтр MAC уровня MAC 802.3 находится над фильтром планировщика пакетов QoS в стеке сетевого интерфейса. Это можно подтвердить, запустив расширение **!ndiskd.netreport** , в котором показан визуальный стек сети.

## См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Обслуживание стека сетевых интерфейсов

Промежуточные драйверы NDIS MUX

**!ndiskd.netreport**

# !ndiskd.compartments

16.08.2025

Расширение **!ndiskd.compartments** отображает все сетевые отсеки.

Консоль

```
!ndiskd.compartments
```

## Параметры

Это расширение не имеет параметров.

## Библиотека dll

Ndiskd.dll

## Замечания

Секции — это способ управления интерфейсами NDIS. Сторонние поставщики интерфейсов используют только основной отсек, как описано в элементе **CompartmentId** структуры NDIS_BIND_PARAMETERS .

## Примеры

Запустите расширение **!ndiskd.compartments** , чтобы просмотреть список всех сетевых отсеков. В этом примере существует только один отсек (основной).

Консоль

```
3: kd> !ndiskd.compartments
    Compartment        ffffdf80139b9940
    ID                 1
    Loopback Network   ffffdf80139b8900
    Loopback Interface ffffdf80139b6a20
    Networks:
                       ffffdf80139b8900    [Unnamed network]
```

## См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**NDIS_BIND_PARAMETERS**

# !ndiskd.pkt

Статья • 08.07.2024

**Предупреждение** , что это расширение предназначено для устаревших драйверов NDIS 5.x. Не рекомендуется использовать структуру NDIS_PACKET и связанную с ней архитектуру.

Расширение **!ndiskd.pkt** отображает сведения о структуре NDIS_PACKET .

Консоль

```
!ndiskd.pkt [-packet] [-verbosity]
```

## Параметры

*Пакет*
Указывает адрес пакета.

*Многословие*
Указывает объем отображаемых сведений.

### DLL-библиотеки

Ndiskd.dll

## См. также

NDIS_PACKET

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.pktpools

Статья • 05.04.2024

**Предупреждение** , что это расширение предназначено для устаревших драйверов NDIS 5.x. Не рекомендуется использовать структуру NDIS_PACKET и связанную с ней архитектуру.

Расширение **!ndiskd.pktpools** отображает список всех выделенных пулов пакетов.

Консоль

```
!ndiskd.pktpools
```

## DLL-библиотеки

Ndiskd.dll

## Примеры

**Запустите расширение !ndiskd.pktpools**, чтобы просмотреть список всех выделенных пулов пакетов в системе. Обратите внимание, что дескрипторы пулов пакетов недоступны для щелчка, что означает, что вы не можете изучить дополнительные сведения о пуле пакетов. Это связано с тем, что NDIS не использует пулы пакетов, начиная с NDIS 6.0, поэтому эти пулы выделяются только для устаревших драйверов, которые по-прежнему могут находиться в старых системах. Компьютер отладки в этом примере не имеет устаревших драйверов NDIS 5.x, поэтому пулы пакетов не используются. Этот пример предназначен только для иллюстрирующих целей.

Консоль

```
3: kd> !ndiskd.pktpools
Pool      Allocator  BlocksAllocated  BlockSize  PktsPerBlock  PacketLength
ffffdf80131d58c0  fffff80f1fbe3e8f   0x1          0x1000     0xa
0x190   ndis!DriverEntry+6af
ffffdf80131d5940  fffff80f1fbe3e71   0x1          0x1000     0xa
0x180   ndis!DriverEntry+691
```

# См. также

Руководство по проектированию сети Windows 2000 и Windows XP

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.findpacket

Статья • 05.04.2024

**Предупреждение** , что это расширение предназначено для устаревших драйверов NDIS 5.x. Не рекомендуется использовать структуру NDIS_PACKET и связанную с ней архитектуру.

Расширение **!ndiskd.findpacket** находит указанные пакеты.

Консоль

```
!ndiskd.findpacket [-VirtualAddress] [-PoolAddress]
```

## Параметры

*VirtualAddress*
Указывает виртуальный адрес, содержащийся в требуемом пакете.

*PoolAddress*
Указывает адрес пула. Будут отображаться все незавершенные пакеты в этом пуле.

### DLL-библиотеки

Ndiskd.dll

## См. также

NDIS_PACKET

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.vc

16.08.2025

Расширение **!ndiskd.vc** отображает виртуальное подключение Connection-Oriented (CoNDIS).

Консоль

```
!ndiskd.vc -handle <x>
```

# Параметры

*-ручка*
Обязательное. Дескриптор указателя VC.

## Библиотека dll

Ndiskd.dll

## Замечания

Дополнительные сведения о CoNDIS см. в Connection-Oriented NDIS.

Дополнительные сведения о виртуальных подключениях CoNDIS см. в разделе "Виртуальные подключения".

## Примеры

CoNDIS используется в определенных ситуациях, таких как подключение к VPN, поэтому запуск **!ndiskd.vc** не будет отображать результаты, если мини-драйвер в вашей системе не создал и активировал виртуальное подключение CoNDIS. В следующем примере показаны результаты компьютера, подключенного к VPN-сети. Сначала запустите расширение **!ndiskd.netadapter** без параметров, чтобы просмотреть список минипортов и минипортов драйверов в системе. В следующих выходных данных найдите мини-драйвер для сетевого адаптера Marvell AVASTAR Wireless-AC сетевого контроллера. Его дескриптор ffffc804af2e3710.

Консоль

```
1: kd> !ndiskd.netadapter
    Driver             NetAdapter          Name
```

```
    ffffc804af2e3710   ffffc804b9e6f1a0    Marvell AVASTAR Wireless-AC Network
Controller
    ffffc804b99b9020   ffffc804b9c301a0    WAN Miniport (Network Monitor)
    ffffc804b99b9020   ffffc804b9c2a1a0    WAN Miniport (IPv6)
    ffffc804b99b9020   ffffc804b8a8a1a0    WAN Miniport (IP)
    ffffc804ae9d7020   ffffc804b9ceb1a0    WAN Miniport (PPPOE)
    ffffc804b9ca5900   ffffc804b9e601a0    WAN Miniport (PPTP)
    ffffc804b99dc720   ffffc804b99b01a0    WAN Miniport (L2TP)
    ffffc804b86581b0   ffffc804b9c6c1a0    WAN Miniport (IKEv2)
    ffffc804ad4a7250   ffffc804b99651a0    WAN Miniport (SSTP)
    ffffc804b11c4020   ffffc804b85821a0    Microsoft ISATAP Adapter
    ffffc804b11c4020   ffffc804b71731a0    Microsoft ISATAP Adapter #2
    ffffc804ad725020   ffffc804b05e71a0    Surface Ethernet Adapter #2
    ffffc804b0bf0020   ffffc804b0c011a0    Bluetooth Device (Personal Area
Network)
    ffffc804aef695e0   ffffc804aed331a0    TAP-Windows Adapter V9
```

Затем введите команду **!ndiskd.vc** с дескриптором драйвера минипорта, чтобы увидеть виртуальные подключения, открытые этим мини-драйвером.

Консоль

```
1: kd> !ndiskd.vc ffffc804af2e3710


VIRTUAL CALL

    [Zero-length string]
    Ndis handle        ffffc804af2e3710
    VC Index           0

    AF                 fffff80965fd5888
    Call Flags         [No flags set]
    VC Flags           [Unrecognized flags 04a80100] VC_ACTIVATE_PENDING

    Miniport           fffff80965ffaa20 - [Invalid NetAdapter]
    Miniport Context   fffff80965ffaad0

    Call Manager       fffff80965ff9b50 - [Invalid Open]
    Call Manager Context fffff80965ff9c70

    Client             ffffc804af96fd78 - [Invalid Open]
    Client Context     00003206
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Connection-Oriented NDIS

Виртуальные подключения

**!ndiskd.netadapter**

# !ndiskd.af

16.08.2025

Расширение **!ndiskd.af** отображает семейство адресов Connection-Oriented NDIS (CoNDIS).

Консоль

```
!ndiskd.af -handle <x>
```

## Параметры

*-ручка*
Обязательное. Дескриптор семейства адресов CoNDIS.

## Библиотека dll

Ndiskd.dll

### Замечания

Дополнительные сведения о CoNDIS см. в Connection-Oriented NDIS.

Дополнительные сведения о семействах адресов CoNDIS см. в разделе "Семьи адресов".

### Примеры

CoNDIS используется в определенных ситуациях, таких как подключение к VPN, поэтому запуск **!ndiskd.af** не будет отображать результаты, если в системе не создано и активировано виртуальное подключение CoNDIS. В следующем примере показаны результаты компьютера, подключенного к VPN-сети. Сначала запустите расширение **!ndiskd.netadapter** без параметров, чтобы просмотреть список минипортов и минипортов драйверов в системе. В следующих выходных данных найдите мини-драйвер для сетевого адаптера Marvell AVASTAR Wireless-AC сетевого контроллера. Его дескриптор ffffc804af2e3710.

Консоль

```
1: kd> !ndiskd.netadapter
    Driver             NetAdapter          Name
    ffffc804af2e3710   ffffc804b9e6f1a0    Marvell AVASTAR Wireless-AC Network
Controller
```

```
    ffffc804b99b9020   ffffc804b9c301a0    WAN Miniport (Network Monitor)
    ffffc804b99b9020   ffffc804b9c2a1a0    WAN Miniport (IPv6)
    ffffc804b99b9020   ffffc804b8a8a1a0    WAN Miniport (IP)
    ffffc804ae9d7020   ffffc804b9ceb1a0    WAN Miniport (PPPOE)
    ffffc804b9ca5900   ffffc804b9e601a0    WAN Miniport (PPTP)
    ffffc804b99dc720   ffffc804b99b01a0    WAN Miniport (L2TP)
    ffffc804b86581b0   ffffc804b9c6c1a0    WAN Miniport (IKEv2)
    ffffc804ad4a7250   ffffc804b99651a0    WAN Miniport (SSTP)
    ffffc804b11c4020   ffffc804b85821a0    Microsoft ISATAP Adapter
    ffffc804b11c4020   ffffc804b71731a0    Microsoft ISATAP Adapter #2
    ffffc804ad725020   ffffc804b05e71a0    Surface Ethernet Adapter #2
    ffffc804b0bf0020   ffffc804b0c011a0    Bluetooth Device (Personal Area
Network)
    ffffc804aef695e0   ffffc804aed331a0    TAP-Windows Adapter V9
```

Затем введите команду **!ndiskd.af** с дескриптором драйвера минипорта, чтобы просмотреть семейство адресов для этого минипорт-драйвера, который выступает в качестве клиента, ориентированного на подключение.

Консоль

```
1: kd> !ndiskd.af ffffc804af2e3710


ADDRESS FAMILY

    Ndis handle        ffffc804af2e3710
    Flags              [Unrecognized flags 399b9020] AF_CLOSING
    References         ffffc804
    Close Requested?   0

    Miniport           0 - [Unreadable NetAdapter]

    Call Manager       ffffc804b90a4ac0 - [Invalid Open]
    Call Manager Context 007a0078

    Client             00060000 - [Unreadable Open]
    Client Context     ffffc804af2e3888


CLIENT HANDLERS

    Client Handler                          Function pointer   Symbol (if
available)
    ClCreateVcHandler                       fffff80965fd5888
mrvlpcie8897!Globals+8
    ClDeleteVcHandler                       [None]
    ClRequestHandler                        ffffc804af96fd78
    ClRequestCompleteHandler                ffffc804af96fd78
    ClOpenAfCompleteHandler                 [None]
    ClCloseAfCompleteHandler                [None]
    ClRegisterSapCompleteHandler            000132060098028a
    ClDeregisterSapCompleteHandler          [None]
```

```
    ClMakeCallCompleteHandler              fffff80965ff9ec0
wdiwifi!MPWrapperSetOptions
    ClCloseCallCompleteHandler             fffff80965ff9c70
wdiwifi!MPWrapperHalt
    ClModifyCallQoSCompleteHandler         fffff80965ff9b50
wdiwifi!MPWrapperInitializeEx
    ClAddPartyCompleteHandler              fffff80965e71a08
mrvlpcie8897!MPUnload
    ClDropPartyCompleteHandler             fffff80965ffa070
wdiwifi!MPWrapperPause
    ClIncomingDropPartyHandler             fffff80965ffaa20
wdiwifi!MPWrapperReturnNetBufferLists
    ClIncomingCallHandler                  fffff80965ffa1e0
wdiwifi!MPWrapperRestart
    ClCallConnectedHandler                 fffff80965ffaad0
wdiwifi!MPWrapperCancelSendNetBufferLists
    ClIncomingCloseCallHandler             fffff80965ffa870
wdiwifi!MPWrapperSendNetBufferLists
    ClIncomingCallQoSChangeHandler         fffff80965ffa610
wdiwifi!MPWrapperOidRequest
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Connection-Oriented NDIS

Семейства адресов

# !ndiskd.ndisref

16.08.2025

Расширение **!ndiskd.ndisref** отображает журнал отладки отслеживаемого количества ссылок.

Консоль

```
!ndiskd.ndisref -handle <x> [-tagtype <str>] [-stacks] [-tag <str>] [-refdebug]
```

## Параметры

*-ручка*
Обязательное. Дескриптор блока ссылок.

*-tagtype*
Тип перечисления тегов.

*-Стеки*
Включает трассировки стека (если доступно).

*-ярлык*
Ограничения, отображаемые для одного тега.

*-refdebug*
Отображает подробный журнал отладки, если он доступен.

## Библиотека dll

Ndiskd.dll

## Примеры

В следующем примере дескриптор минипорта NDIS передается в расширение **!ndiskd.ndisref** для отображения блока ссылок для этого драйвера. Сначала запустите !ndiskd.minidriver без параметров, чтобы просмотреть список всех драйверов минипорта в системе. В приведенном ниже примере найдите дескриптор драйвера kdnic, ffffdf801418d650.

Консоль

```
3: kd> !ndiskd.minidriver
    ffffdf8015a98380 - tunnel
    ffffdf801418d650 - kdnic
```

С помощью дескриптора минипорта введите команду **!ndiskd.ndisref -handle** , чтобы просмотреть блок ссылок для этого мини-драйвера. В следующем примере посередине блока ссылок вырезается для краткости.

Консоль

```
3: kd> !ndiskd.ndisref ffffdf801418d650


REFCOUNT BLOCK

    Tag                                     Number of references
    0n0                                     0n6293              Show stacks
    0n1                                     0n1045              Show stacks
    0n2                                     0n4294966717        Show stacks
    0n5                                     0n25048             Show stacks
    0n18                                    0n4294967293        Show stacks
    0n19                                    0n4294967295        Show stacks
    0n21                                    0n4294967036        Show stacks
    0n23                                    0n30818             Show stacks
    0n24                                    0n24693             Show stacks
    0n25                                    0n25808             Show stacks

...

    0n153                                   0n7                 Show stacks
    0n154                                   0n3                 Show stacks
    0n156                                   0n29972             Show stacks
    0n159                                   0n4294959128        Show stacks
    0n160                                   0n30892             Show stacks
    0n161                                   0n136               Show stacks
    0n162                                   0n4294951910        Show stacks
    0n163                                   0n30892             Show stacks
    0n164                                   0n136               Show stacks
    0n167                                   0n4294965996        Show stacks

    Include inactive tags
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

## Расширения NDIS (Ndiskd.dll)

### !ndiskd.help

### !ndiskd.minidriver

# !ndiskd.ndisevent

Статья • 05.04.2024

**Обратите внимание** , что сторонние разработчики сетевых драйверов не должны вручную использовать эту команду расширения. Вы можете запустить его, чтобы просмотреть отображаемые сведения, но вы не можете повторно использовать сведения, которые он предоставляет в драйвере.

Расширение **!ndiskd.ndisevent** отображает журнал событий отладки NDIS.

Консоль

```
!ndiskd.ndisevent -handle <x> [-tagtype <str>]
```

# Параметры

*-Обрабатывать*
Обязательный. Дескриптор журнала событий.

*-tagtype*
Тип перечисления тегов.

## DLL-библиотеки

Ndiskd.dll

## Примеры

Чтобы просмотреть выходные данные журнала событий для сетевого адаптера, !ndiskd предоставляет ссылку на него в разделе State выходных **данных !ndiskd.netadapter** . Это проще, чем ручной метод поиска дескриптора журнала событий из мини-порта и использования этого для запуска **расширения !ndiskd.ndisevent** .

Во-первых, введите **команду !ndiskd.netadapter** без параметров, чтобы просмотреть список сетевых адаптеров и мини-портов драйверов в системе. В следующем примере найдите дескриптор сетевого контроллера Marvell AVASTAR Wireless-AC, ffffc804b9e6f1a0.

Консоль

```
1: kd> !ndiskd.netadapter
    Driver             NetAdapter          Name
    ffffc804af2e3710   ffffc804b9e6f1a0    Marvell AVASTAR Wireless-AC
Network Controller
    ffffc804b99b9020   ffffc804b9c301a0    WAN Miniport (Network Monitor)
    ffffc804b99b9020   ffffc804b9c2a1a0    WAN Miniport (IPv6)
    ffffc804b99b9020   ffffc804b8a8a1a0    WAN Miniport (IP)
    ffffc804ae9d7020   ffffc804b9ceb1a0    WAN Miniport (PPPOE)
    ffffc804b9ca5900   ffffc804b9e601a0    WAN Miniport (PPTP)
    ffffc804b99dc720   ffffc804b99b01a0    WAN Miniport (L2TP)
    ffffc804b86581b0   ffffc804b9c6c1a0    WAN Miniport (IKEv2)
    ffffc804ad4a7250   ffffc804b99651a0    WAN Miniport (SSTP)
    ffffc804b11c4020   ffffc804b85821a0    Microsoft ISATAP Adapter
    ffffc804b11c4020   ffffc804b71731a0    Microsoft ISATAP Adapter #2
    ffffc804ad725020   ffffc804b05e71a0    Surface Ethernet Adapter #2
    ffffc804b0bf0020   ffffc804b0c011a0    Bluetooth Device (Personal Area
Network)
    ffffc804aef695e0   ffffc804aed331a0    TAP-Windows Adapter V9
```

Теперь щелкните ссылку для этого NetAdapter или введите **команду !ndiskd.netadapter -handle** , чтобы просмотреть ее сведения. Найдите ссылку "Показать журнал состояний" справа от поля Device PnP в разделе "Состояние".

Консоль

```
1: kd> !ndiskd.netadapter ffffc804b9e6f1a0


MINIPORT

    Marvell AVASTAR Wireless-AC Network Controller

    Ndis handle        ffffc804b9e6f1a0
    Ndis API version   v6.50
    Adapter context    ffffc804af3b1100
    Driver             ffffc804af2e3710 - mrvlpcie8897  v1.0
    Network interface  ffffc804aea60a20

    Media type         802.3
    Physical medium    NdisPhysicalMediumUnspecified
    Device instance
PCI\VEN_11AB&DEV_2B38&SUBSYS_045E0001&REV_00\4&379f07b2&0&00E0
    Device object      ffffc804b9e6f050    More information
    MAC address        c0-33-5e-13-22-f7


STATE

    Miniport           INITIALIZING
    Device PnP         ADDED               Show state history
    Datapath           Normal
    Operational status DOWN
```

```
    Operational flags  [No flags set]
    Admin status       ADMIN_UP
    Media              MediaConnectUnknown
    Power              D0
    References         1                   Show detail
    Total resets       0
    Pending OID        None
    Flags              IN_INITIALIZE, NOT_BUS_MASTER,
DEFAULT_PORT_ACTIVATED,
                       NOT_SUPPORTS_MEDIA_SENSE, DOES_LOOPBACK,
MEDIA_CONNECTED
    PnP flags          PM_SUPPORTED, RECEIVED_START, HARDWARE_DEVICE


WDI

    This system supports WDI.
    Learn more about the associated WDI state


BINDINGS

    Protocol list      Driver              Open               Context
    No protocols are bound to this miniport

    Filter list        Driver              Module             Context
    No filters are bound to this miniport



MORE INFORMATION

    Driver handlers                        Task offloads
    Power management                       PM protocol offloads
    Pending OIDs                           Timers
    Pending NBLs                           Receive side throttling
    Wake-on-LAN (WoL)                      Packet filter
    Receive queues                         Receive filtering
    RSS                                    NIC switch
    Hardware resources                     Selective suspend
    NDIS ports                             WMI guids
    Diagnostic log
```

Теперь можно щелкнуть ссылку "Показать журнал состояний" или использовать дескриптор сетевого адаптера, чтобы ввести **команду !ndiskd.netadapter -handle -log** , которая будет отображать журнал событий PnP для мини-порта драйвера минипорта.

Консоль

```
1: kd> !ndiskd.netadapter ffffc804b9e6f1a0 -log
```

```
MINIPORT PM & PNP EVENTS

    Event               Timestamp           (most recent event at bottom)
    DeviceAdded
                        13 ms later
    DeviceStart
                        Mon Mar 20 21:27:07.106 2017 (UTC - 7:00) Now?

    Set a breakpoint on the next event
```

## См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**!ndiskd.netadapter**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.ndisstack

16.08.2025

**Заметка** Сторонние разработчики сетевых драйверов не должны вручную использовать эту команду расширения. Вы можете запустить его, чтобы просмотреть отображаемые сведения, но вы не можете повторно использовать сведения, которые он предоставляет в драйвере.

Расширение **!ndiskd.ndisstack** отображает трассировку стека отладки.

Консоль

```
!ndiskd.ndisstack -handle <x> [-statistics]
```

## Параметры

*-ручка*
Обязательное. Дескриптор блока стека.

*-статистика*
Отображает статистику отладки.

## Библиотека dll

Ndiskd.dll

## См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

# !ndiskd.wdiadapter

Статья • 05.04.2024

Расширение **!ndiskd.wdiadapter** отображает сведения о WDIWiFi! Структура CAdapter. Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех WDIWiFi! Структуры CAdapter.

Дополнительные сведения о драйверах минипорта WDI см. в руководстве по проектированию драйверов минипорта WDI.

Дополнительные сведения о справочнике по драйверу минипорта WDI см. в справочнике по драйверу минипорта WDI.

Консоль

```
!ndiskd.wdiadapter [-handle <x>] [-pm] [-rcvfilter]
```

# Параметры

*-Обрабатывать*
Необязательный дескриптор объекта CAdapter.

*-pm*
Показывает состояние и возможности управления питанием.

*-rcvfilter*
Отображает возможности фильтрации.

## DLL-библиотеки

Ndiskd.dll

## Примеры

**Запустите расширение !ndiskd.wdiadapter** без параметров, чтобы просмотреть список всех объектов CAdapter, а также подробные сведения для каждого из их адаптеров WDI. В следующем примере существует только одна структура CAdapter. Дескриптор связанного адаптера WDI — ffffc804af396000.

Консоль

```
1: kd> !ndiskd.wdiadapter
    CAdapter
    ffffc804af396000 - WDI Adapter


WDI ADAPTER

    Object             ffffc804af396000
    Miniport           ffffc804b9e6f1a0
    WDI driver         ffffc804b8ce7c40 - WDI MiniDriver
    NetGuid            47cb3aa5-e29c-4628-80bd-5023c53fcccb
    Power              D0

    CCtlPlane          ffffc804af396080
    CTxMgr             ffffc804af399790


DEVICE COMMANDS

    Scheduler state    DeviceCommandSchedulerStateInit
    [No queued device commands found]

Device Command Scheduler


SERIALIZED JOBS

    Job                State              Type               Child
    ffffc804b9f33000   StepPending        WiFiJobMiniportInitialize



ACTIVE JOBS

    Job                State              Type               Child
    ffffc804b9f33000   StepPending        WiFiJobMiniportInitialize



EVENTS

    Processing?        No

    Event              Status             Type
    [No events found]

Event Queue


MORE INFORMATION

    Power management                      Receive filtering
```

Теперь можно щелкнуть ссылки "Управление питанием" и "Получить фильтрацию" в нижней части описания структуры CAdapter или ввести **команду** **!ndiskd.wdiadapter -handle** с параметром *-pm* или *-rcvfilter* . В следующем примере показаны выходные данные для *параметра -rcvfilter* .

Консоль

```
1: kd> !ndiskd.wdiadapter ffffc804af396000 -rcvfilter


RECEIVE FILTER

    Current Configuration

    Revision                                0
    Flags                                   0
    Enabled filter types                    [No flags set]
    Enabled queue types                     [No flags set]
    Max Queues                              0
    Queue properties                        [No flags set]
    Filter tests                            [No flags set]
    Header types                            [No flags set]
    MAC header fields                       [No flags set]
    Max MAC header filters                  0
    Max queue groups                        0
    Max queues per group                    0
    Min lookahead split size                0
    Max lookahead split size                0
```

# См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Руководство по проектированию драйвера минипорта WDI

Справочник по драйверу минипорта WDI

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.wdiminidriver

Статья • 08.07.2024

Расширение **!ndiskd.wdiminidriver** отображает сведения о одной или нескольких структурах CMiniportDriver. Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех структур CMiniportDriver.

Дополнительные сведения о драйверах минипорта WDI см. в руководстве по проектированию драйверов минипорта WDI.

Дополнительные сведения о справочнике по драйверу минипорта WDI см. в справочнике по драйверу минипорта WDI.

Консоль

```
!ndiskd.wdiminidriver [-handle <x>] [-pm] [-rcvfilter]
```

# Параметры

*-ручка*
Необязательный дескриптор объекта CMiniportDriver.

*-основной*
Отображает основные сведения о драйвере минипорта.

*-Обработчики*
Отображает мини-обработчики этого драйвера.

## DLL-библиотеки

Ndiskd.dll

## Примеры

**Запустите расширение !ndiskd.wdiminidriver без параметров, чтобы просмотреть список всех объектов CMiniportDriver**. В следующем примере существует только один объект CMiniportDriver. Дескриптор WdiMiniDriver — ffffc804b8ce7c40.

Консоль

```
1: kd> !ndiskd.wdiminidriver
    WdiMiniDriver      Name
```

```
    ffffc804b8ce7c40 - WDI MiniDriver
```

Щелкните дескриптор WdiMiniDriver или введите **команду !ndiskd.wdiminidriver -handle** , чтобы просмотреть сведения об этом мини-драйвере WDI.

Консоль

```
1: kd> !ndiskd.wdiminidriver ffffc804b8ce7c40


WDI MINIPORT DRIVER

    Object             ffffc804b8ce7c40
    DRIVER_OBJECT      ffffc804b90a4ac0

    Ndis API version   v6.50
    NDIS driver        ffffc804af2e3710 - mrvlpcie8897
    Driver version     v1.0
    WDI API version    v1.0.1


    Handlers
```

Теперь можно щелкнуть ссылку "Обработчики" в нижней части сведений о драйвере минипорта WDI или использовать дескриптор для ввода **команды !ndiskd.wdiminidriver -handle-handlers, чтобы просмотреть список всех обработчиков** WDI Miniport Driver.

Консоль

```
1: kd> !ndiskd.wdiminidriver ffffc804b8ce7c40 -handlers


HANDLERS

    Miniport Handler                        Function pointer   Symbol (if
available)
    InitializeHandlerEx                     [None]
    SetOptionsHandler                       fffff80965e8cae0
mrvlpcie8897!wlan_cmd_queue_deinit
    UnloadHandler                           fffff80965e71a08
mrvlpcie8897!MPUnload
    HaltHandlerEx                           [None]
    ShutdownHandlerEx                       fffff80965e71974
mrvlpcie8897!MPShutdown

    CheckForHangHandlerEx                   [None]
    ResetHandlerEx                          [None]

    PauseHandler                            [None]
```

```
    RestartHandler                          [None]

    OidRequestHandler                       [None]
    SendNetBufferListsHandler               [None]
    ReturnNetBufferListsHandler             [None]
    CancelSendHandler                       [None]
    CancelOidRequestHandler                 [None]
    DirectOidRequestHandler                 [None]
    CancelDirectOidRequestHandler           [None]
    DevicePnPEventNotifyHandler             fffff80965e71868
mrvlpcie8897!MPPnPEventNotify

    AllocateAdapterHandler                  fffff80965e713ec
mrvlpcie8897!MPAllocateTalAdapter
    FreeAdapterHandler                      fffff80965e717e8
mrvlpcie8897!MPFreeTalAdapter
    OpenAdapterHandler                      fffff80965e719b8
mrvlpcie8897!MPTaskOpenHandler
    CloseAdapterHandler                     fffff80965e719ac
mrvlpcie8897!MPTaskCloseHandler
    StartOperationHandler                   [None]
    StopOperationHandler                    [None]
    PostPauseHandler                        [None]
    PostRestartHandler                      [None]
    HangDiagnoseHandler                     fffff80965e715d8
mrvlpcie8897!MPDiagnosticHandler
    TalTxRxInitializeHandler                fffff80965e752d4
mrvlpcie8897!TalDataPathInitialize
    TalTxRxDeinitializeHandler              fffff80965e7527c
mrvlpcie8897!TalDataPathDeinitialize

    OpenAdapterCompleteHandler              fffff80965ffab50
wdiwifi!WDIOpenAdapterCompleteHandler
    CloseAdapterCompleteHandler             fffff80965fface0
wdiwifi!WDICloseAdapterCompleteHandler
```

# См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Руководство по проектированию драйвера минипорта WDI

Справочник по драйверу минипорта WDI

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.nwadapter

Статья • 08.07.2024

Расширение **!ndiskd.nwadapter** отображает сведения об одном или нескольких nwifi! СТРУКТУРЫ ADAPT. Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех nwifi! СТРУКТУРЫ ADAPT.

Консоль

```
!ndiskd.nwadapter [-handle <x>]
```

## Параметры

*-ручка*
Адрес блока ADAPT.

## DLL-библиотеки

Ndiskd.dll

## См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.ndisrwlock

16.08.2025

Расширение **!ndiskd.ndisrwlock** отображает сведения о структуре блокировки NDIS_RW_LOCK_EX .

Консоль

```
!ndiskd.ndisrwlock -handle <x>
```

# Параметры

*-ручка*
Обязательное. Дескриптор структуры блокировки.

## Библиотека dll

Ndiskd.dll

## Примеры

Используйте расширение **!ndiskd.ndisrwlock** , если вы создаете собственную блокировку RW и хотите проверить ее. Чтобы получить дескриптор блокировки RW, используйте команду *poi* , чтобы разыменовать адрес блокировки драйвера. В следующем фрагменте кода показано, как просмотреть блокировку, используемую протоколом TCIPIP во время примера.

Консоль

```
0: kd> !ndiskd.ndisrwlock poi(tcpip!gAleHashtableLock)


NDIS READ-WRITE LOCK

    Allocated by       [NDIS generic object]
    Exclusive access   Not acquired
    Read-only access   0 references

    Set a breakpoint on acquire/release
```

Чтобы наблюдать за драйвером с помощью этой блокировки RW, щелкните ссылку "Задать точку останова при приобретении или выпуске" в нижней части сведений о

блокировке RW. После установки точки останова введите команду **g** , чтобы позволить отладочному компьютеру запускаться и нажать точку останова.

Консоль

```
0: kd> ba r4 ffffe00bc3fc22f8
0: kd> g
Breakpoint 0 hit
nt!KeTestSpinLock+0x3:
fffff802`0d69eb53 4885c0          test    rax,rax
```

Теперь вы можете повторно запустить ту же команду **!ndiskd.ndisrwlock** , чтобы увидеть, что эта блокировка RW содержит одну ссылку на доступ только для чтения.

Консоль

```
0: kd> !ndiskd.ndisrwlock poi(tcpip!gAleHashtableLock)


NDIS READ-WRITE LOCK

    Allocated by       [NDIS generic object]
    Exclusive access   Not acquired
    Read-only access   1 reference

    Set a breakpoint on acquire/release
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**NDIS_RW_LOCK_EX**

# !ndiskd.ndisslot

16.08.2025

**Заметка** Сторонние разработчики сетевых драйверов не должны вручную использовать эту команду расширения. Вы можете запустить его, чтобы просмотреть отображаемые сведения, но вы не можете повторно использовать сведения, которые он предоставляет в драйвере.

Расширение **!ndiskd.ndisslot** отображает содержимое переменной NDIS на процессор. Если вы запускаете это расширение без параметров, !ndiskd отобразит список всех переменных NDIS для каждого процессора в системе.

Консоль

```
!ndiskd.ndisslot [-handle <x>] [-itemtype <str>]
```

## Параметры

*-ручка*
Дескриптор слота.

*-itemtype*
Тип значения, хранящегося в слоте.

## Библиотека dll

Ndiskd.dll

## Примеры

Запустите расширение **!ndiskd.ndisslot** без параметров, чтобы просмотреть список всех переменных слота для каждого процессора. В следующем примере выходные данные обрезали среднюю часть списка для краткости.

Консоль

```
1: kd> !ndiskd.ndisslot
    Per-processor slot                    Summary of contents
    ffffc804ae060000 - NDrw               All values are zero
    ffffc804ae060008 - NDrw               All values are zero
    ffffc804ae060010 - NDrw               All values are zero
    ffffc804ae060018 - NDrw               All values are zero
```

```
    ffffc804ae060020 - NDrw                All values are zero
    ffffc804ae060028 - NDrw                All values are zero
    ffffc804ae060030 - NDrw                All values are zero
    ffffc804ae060038 - NDrw                All values are zero
    ffffc804ae060040 - NDrw                All values are zero
    ffffc804ae060048 - NDrw                All values are zero

...

    ffffc804ae060910 - NDtk                All values are zero
    ffffc804ae060918 - NDtk                All values are zero
    ffffc804ae060920 - tsR                 All values are non-zero
    ffffc804ae060928 - NDrw                All values are zero
    ffffc804ae060930 - NDtk                All values are zero
    ffffc804ae060938 - NDtk                All values are zero
    ffffc804ae060940 - NDtk                All values are zero
    ffffc804ae060948 - NDtk                All values are zero
    ffffc804ae060950 - NDrw                All values are zero
    ffffc804ae060958 - NDrw                All values are zero

    Statistics
    Descriptors        1
    Total slots        512
    Slots available    275
    Slots used         237
    Efficiency         46%
```

Щелкнув один из дескрипторов для переменных слота для каждого процессора, вы увидите сведения об этой переменной. В следующем примере для переменной tsR используется дескриптор ffffc804ae060920 из предыдущего примера.

Консоль

```
1: kd> !ndiskd.ndisslot ffffc804ae060920
    Processor          Slot value
    00                 00000006
    01                 00000006
    02                 00000006
    03                 00000006
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

# !ndiskd.help

# !ndiskd.ndis

Статья • 05.04.2024

Расширение **!ndiskd.ndis** отображает сведения о сборке ndis.sys.

Консоль

```
!ndiskd.ndis
```

## Параметры

Это расширение не имеет параметров.

## DLL-библиотеки

Ndiskd.dll

## Примеры

В следующем примере показано, что компьютер отладки имеет сборку Checked Ndis.

Консоль

```
0: kd> !ndiskd.ndis
Checked Ndis
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.dbglevel

16.08.2025

Расширение **!ndiskd.dbglevel** отображается и при необходимости изменяет текущий уровень отладки NDIS.

**Warning!ndiskd.dbglevel** заменен WPP (препроцессор трассировки программного обеспечения Windows) и средство проверки драйверов. !ndiskd даст вам следующее предупреждение, если целевая система не поддерживает **!ndiskd.dbglevel**.

Консоль

```
0: kd> !ndiskd.dbglevel
    This target does not support tracing through !ndiskd.dbglevel or
    !ndiskd.dbgsystems.
    Learn how to collect traces with WPP
```

Если щелкнуть ссылку в нижней части предупреждения, !ndiskd даст вам дополнительные сведения.

Консоль

```
0: kd> !ndiskd.help wpptracing
    WPP traces are fast, flexible, and detailed.  Plus, starting with Windows 8
    and Windows Server 2012, you can automatically decode NDIS traces using the
    symbol file.  Just point TraceView (or tracepdb.exe) at NDIS.PDB, and it
    will be able to get all the TMFs it needs to trace NDIS activity.

    If you would like traces to be printed in the debugger window, you use the
    !wmitrace extension.  For example, you might enable traces with this:

    !wmitrace.searchpath c:\path\to\TMF\files
    !wmitrace.start ndis -kd
    !wmitrace.enable ndis {DD7A21E6-A651-46D4-B7C2-66543067B869} -level 4 -flag
0x31f3
```

Дополнительные сведения о WPP см. в разделе "Трассировка программного обеспечения WPP".

Дополнительные сведения о средство проверки драйверов см. в разделе "Средство проверки драйверов".

Дополнительные сведения о трассировке WMI см. в разделе "Расширения трассировки WMI" (Wmitrace.dll).

Консоль

```
!ndiskd.dbglevel [-level <str>]
```

# Параметры

*-уровень*
Уровень детализации отладки. Возможны следующие значения:

- NONE — отключает трассировку отладки
- FATAL — позволяет распечатать неустранимые ошибки
- ERROR — позволяет распечатать ошибки
- WARN — позволяет выводить предупреждения
- INFO — позволяет распечатать информационные сообщения
- VERBOSE — позволяет распечатать все отладочные трассировки

## Библиотека dll

Ndiskd.dll

## Замечания

Это расширение применяется только к проверяемой NDIS.sys. Чтобы проверить сведения о сборке NDIS.sys, выполните расширение **!ndiskd.ndis** .

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**!ndiskd.ndis**

Трассировка программного обеспечения WPP

Средство проверки драйверов

Расширения трассировки WMI (Wmitrace.dll)

# !ndiskd.dbgsystems

16.08.2025

Расширение **!ndiskd.dbgsystems** отображает и при необходимости изменяет подсистемы NDIS с включенными трассировками отладки.

**Warning!ndiskd.dbgsystems** заменены WPP (препроцессор трассировки программного обеспечения Windows) и средство проверки драйверов. !ndiskd даст вам следующее предупреждение, если целевая система не поддерживает **!ndiskd.dbgsystems**.

Консоль

```
0: kd> !ndiskd.dbgsystems
    This target does not support tracing through !ndiskd.dbglevel or
    !ndiskd.dbgsystems.
    Learn how to collect traces with WPP
```

Если щелкнуть ссылку в нижней части предупреждения, !ndiskd даст вам дополнительные сведения.

Консоль

```
0: kd> !ndiskd.help wpptracing
    WPP traces are fast, flexible, and detailed.  Plus, starting with Windows 8
    and Windows Server 2012, you can automatically decode NDIS traces using the
    symbol file.  Just point TraceView (or tracepdb.exe) at NDIS.PDB, and it
    will be able to get all the TMFs it needs to trace NDIS activity.

    If you would like traces to be printed in the debugger window, you use the
    !wmitrace extension.  For example, you might enable traces with this:

    !wmitrace.searchpath c:\path\to\TMF\files
    !wmitrace.start ndis -kd
    !wmitrace.enable ndis {DD7A21E6-A651-46D4-B7C2-66543067B869} -level 4 -flag
0x31f3
```

Дополнительные сведения о WPP см. в разделе "Трассировка программного обеспечения WPP".

Дополнительные сведения о средство проверки драйверов см. в разделе "Средство проверки драйверов".

Дополнительные сведения о трассировке WMI см. в разделе "Расширения трассировки WMI" (Wmitrace.dll).

Консоль

```
!ndiskd.dbgsystems [-subsystem <any>]
```

# Параметры

*-подсистема*
Подсистема для переключения.

Если выбрано несколько компонентов, разделите их пробелами. Если ранее выбранный компонент повторяется, его мониторинг отладки будет отключен. Возможны следующие значения:

Развернуть таблицу

| Ценность | Значение |
|---|---|
| INIT | Инициализация адаптера трассировки. |
| Конфигурация | Конфигурация адаптера трассировки. |
| SEND | Трассировки, отправляя данные по сети. |
| RECV | Трассировки получения данных из сети. |
| ПРОТОКОЛ | Трассировки операций протокола. |
| СВЯЗЫВАТЬ | Трассировка операций привязки. |
| BUS_QUERY | Трассировка запросов шины. |
| РЕГИСТРАТУРА | Выполняет трассировку операций реестра. |
| ПАМЯТЬ | Отслеживает управление памятью. |
| ФИЛЬТР | Операции фильтрации трассировок. |
| ПРОСЬБА | Трассировка запросов. |
| WORK_ITEM | Трассировка операций рабочего элемента. |
| PNP | Трассировка операций Plug and Play. |
| премьер-министр | Трассировка операций управления питанием. |
| OPEN | Трассировка операций, открывающих эталонные объекты. |
| ЗАМКИ | Трассировка операций блокировки. |

| | |
|---|---|
| СБРОС | Трассировка выполняет сброс операций. |
| WMI | Трассирует операции инструментирования управления Windows. |
| NDIS_CO | Трассировки Connection-Oriented NDIS. |
| ССЫЛКА | Трассировки ссылочных операций. |

## Библиотека dll

Ndiskd.dll

## Замечания

Это расширение применяется только к проверяемой NDIS.sys. Чтобы проверить сведения о сборке NDIS.sys, выполните расширение **!ndiskd.ndis** .

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**!ndiskd.ndis**

Трассировка программного обеспечения WPP

Средство проверки драйверов

Расширения трассировки WMI (Wmitrace.dll)

# !ndiskd.ndiskdversion

16.08.2025

Расширение **!ndiskd.ndiskdversion** отображает сведения о самом !ndiskd.

Консоль

```
!ndiskd.ndiskdversion
```

## Параметры

Это расширение не имеет параметров.

## Библиотека dll

Ndiskd.dll

## Примеры

В следующем примере показаны выходные данные для **!ndiskd.ndiskdversion**.

Консоль

```
0: kd> !ndiskd.ndiskdversion
    NDISKD Version     17.08.00 (NDISKD codename "We'll deploy IPv6 real soon
now")
    Build              Release
    Debugger CPU       AMD64
    Hyperlinks (DML)   Enabled
    Unicode            Enabled
    Debug NDISKD       Disabled
```

## См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

# !ndiskd.help

# !ndiskd.netreport

Статья • 05.04.2024

Расширение **!ndiskd.netreport** создает визуальный отчет всего сетевого стека. **Отчет !ndiskd.netreport** создает HTML-файл, и он даст ссылку на его расположение. HTML-файл содержит подробные сведения о сетевом стеке, поэтому если вам нужно предоставить общий доступ к нему для анализа, вы можете по электронной почте вместо отправки большого файла дампа аварийного дампа.

Консоль

```
!ndiskd.netreport [-outputpath <str>] [-jsononly]
```

# Параметры

*-outputpath*
Указывает место записи файла отчета.

*-jsononly*
Записывает только необработанные данные без HTML.

## DLL-библиотеки

Ndiskd.dll

## Примеры

Запустите расширение !ndiskd.netreport**, чтобы нарисовать прямоугольную** схему сетевого стека.

Консоль

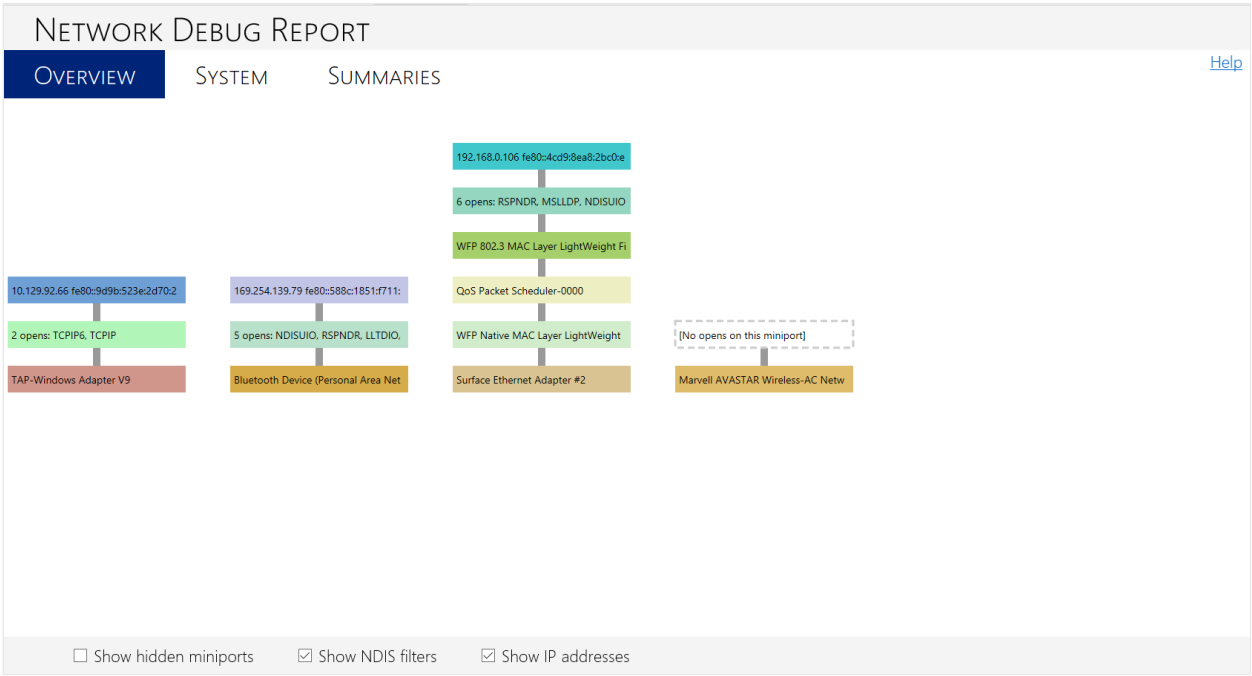
```
1: kd> !ndiskd.netreport


NETWORK STACK REPORT


    Want more stuff?  Rerun with the -verbose flag
```

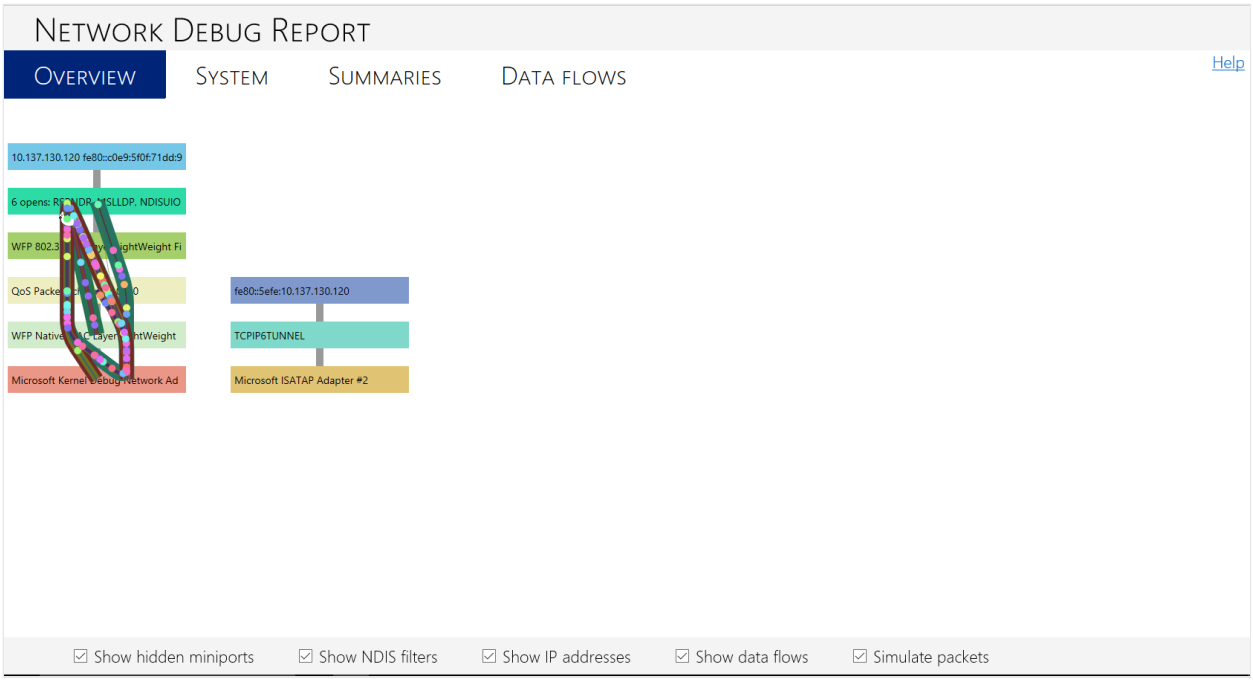
```
Report was saved to C:\Users\******\AppData\Local\Temp\NKDFE9F.html
View the report                          Send in email
```

Щелкните ссылку "Просмотреть отчет" внизу, чтобы просмотреть созданный отчет. На следующем рисунке показан чистый отчет, созданный из файла аварийного дампа. Каждый вертикальный стек — это сетевой адаптер, разделенный на слои, показывающие компоненты стека. Цвет каждого поля создается путем хэширования имени компонента, что означает, что одни и те же компоненты будут отображаться с одинаковым цветом при каждом запуске отчета. Это означает, что вы можете легко выбрать конкретный драйвер или адаптер при отладке проблемы с ним.




Как сравнение, на следующем рисунке показан чистый отчет, созданный из активной системы, а не файл аварийного дампа. Обратите внимание, что в нижней части HTML-страницы есть еще два варианта: "Показать потоки данных" и "Имитация пакетов", а в верхней части отчета — "Потоки данных". Эти параметры появились, так как в отладочном компьютере включена функция отслеживания NBL, что позволяет **!ndiskd.netreport** анализировать журнал отслеживания NBL для визуального отображения информации. Если отслеживание NBL не включено, эти параметры не будут отображаться. Дополнительные сведения об отслеживании NBL и журнале NBL см. в разделе !ndiskd.nbllog.

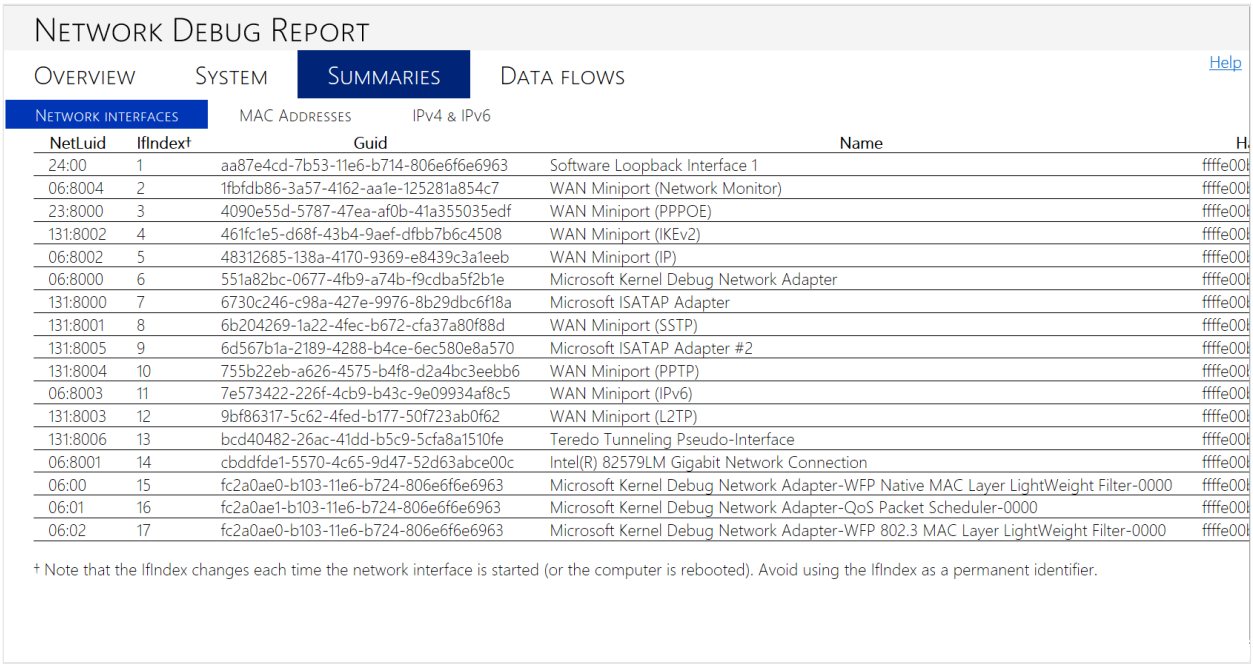
Проверка поле "Показать потоки данных" можно увидеть пути, по которым передаются данные. Проверка поле "Имитация пакетов" можно увидеть анимированные круги, перемещающиеся вверх и вниз по путям потока данных. Каждый круг представляет сетевой пакет.




Второй пример активной системы также показывает другое отличие от первого примера, в котором использовался файл аварийного дампа. Целевой компьютер отладки во втором примере был подготовлен для отладки ядра по сети, поэтому сетевой адаптер в стеке с потоками данных — это сетевой адаптер отладки ядра Майкрософт. Этот адаптер обычно скрыт, если отладка ядра не включена на компьютере отладчика. В действительности сетевой адаптер отладки ядра зарезервирован адаптер Ethernet компьютера для сеанса отладки, поэтому трафик передается через Ethernet.

Возможность визуализировать сетевой стек и увидеть, где происходит поток трафика, позволяет быстро определить, где может возникнуть проблема. Это может быть особенно полезно для виртуальных коммутаторов или серверов, которые имеют более сложные сетевые схемы, чем предыдущие примеры. Например, в Windows Server, использующем объединение сетевых карт, можно увидеть, пересекает ли несколько сетевых стеков друг друга, чтобы сбалансировать нагрузку трафика и определить, возникла ли проблема в нижней части одного стека, влияющего на другой стек. Пример отчета об отладке сети, который показывает это, см. в разделе "Отладка сетевого стека". Дополнительные сведения о команде сетевых адаптеров см. в разделе "Использование объединения сетевых адаптеров для производительности сетевой подсистемы".

**!ndiskd.netreport** также имеет другие вкладки в верхней части страницы для системных, сводок и Поток данных (если применимо). Эти вкладки содержат дополнительные полезные сведения о состоянии сетевого стека. На следующем рисунке показана вкладка "Сетевые интерфейсы" на вкладке "Сводки". Таблица на этой вкладке позволяет просмотреть дополнительные сведения об именах и идентификаторах сетевых интерфейсов в системе.



NETWORK DEBUG REPORT

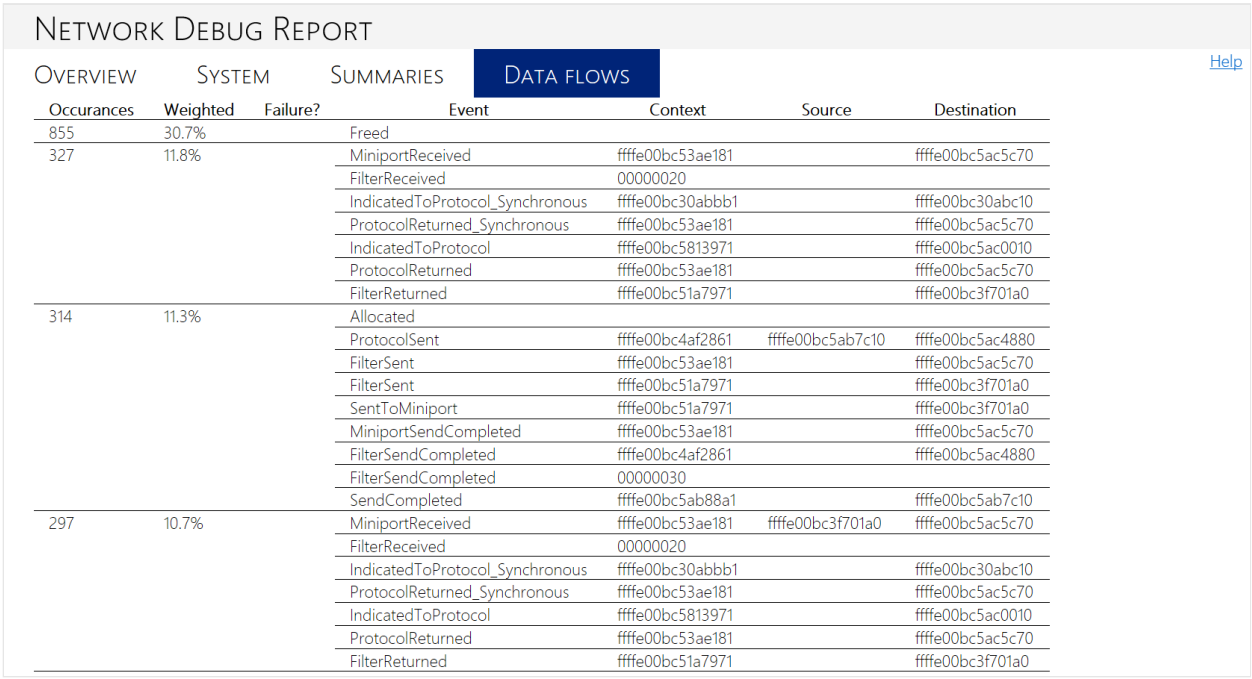
OVERVIEW SYSTEM SUMMARIES DATA FLOWS Help

NETWORK INTERFACES MAC ADDRESSES IPv4 & IPv6

| NetLuid | IfIndex† | Guid | Name | H |
|---|---|---|---|---|
| 24:00 | 1 | aa87e4cd-7b53-11e6-b714-806e6f6e6963 | Software Loopback Interface 1 | ffffe00 |
| 06:8004 | 2 | 1fbfdb86-3a57-4162-aa1e-125281a854c7 | WAN Miniport (Network Monitor) | ffffe00 |
| 23:8000 | 3 | 4090e55d-5787-47ea-af0b-41a355035edf | WAN Miniport (PPPOE) | ffffe00 |
| 131:8002 | 4 | 461fc1e5-d68f-43b4-9aef-dfbb7b6c4508 | WAN Miniport (IKEv2) | ffffe00 |
| 06:8002 | 5 | 48312685-138a-4170-9369-e8439c3a1eeb | WAN Miniport (IP) | ffffe00 |
| 06:8000 | 6 | 551a82bc-0677-4fb9-a74b-f9cdba5f2b1e | Microsoft Kernel Debug Network Adapter | ffffe00 |
| 131:8000 | 7 | 6730c246-c98a-427e-9976-8b29dbc6f18a | Microsoft ISATAP Adapter | ffffe00 |
| 131:8001 | 8 | 6b204269-1a22-4fec-b672-cfa37a80f88d | WAN Miniport (SSTP) | ffffe00 |
| 131:8005 | 9 | 6d567b1a-2189-4288-b4ce-6ec580e8a570 | Microsoft ISATAP Adapter #2 | ffffe00 |
| 131:8004 | 10 | 755b22eb-a626-4575-b4f8-d2a4bc3eebb6 | WAN Miniport (PPTP) | ffffe00 |
| 06:8003 | 11 | 7e573422-226f-4cb9-b43c-9e09934af8c5 | WAN Miniport (IPv6) | ffffe00 |
| 131:8003 | 12 | 9bf86317-5c62-4fed-b177-50f723ab0f62 | WAN Miniport (L2TP) | ffffe00 |
| 131:8006 | 13 | bcd40482-26ac-41dd-b5c9-5cfa8a1510fe | Teredo Tunneling Pseudo-Interface | ffffe00 |
| 06:8001 | 14 | cbddfde1-5570-4c65-9d47-52d63abce00c | Intel(R) 82579LM Gigabit Network Connection | ffffe00 |
| 06:00 | 15 | fc2a0ae0-b103-11e6-b724-806e6f6e6963 | Microsoft Kernel Debug Network Adapter-WFP Native MAC Layer LightWeight Filter-0000 | ffffe00 |
| 06:01 | 16 | fc2a0ae1-b103-11e6-b724-806e6f6e6963 | Microsoft Kernel Debug Network Adapter-QoS Packet Scheduler-0000 | ffffe00 |
| 06:02 | 17 | fc2a0ae0-b103-11e6-b724-806e6f6e6963 | Microsoft Kernel Debug Network Adapter-WFP 802.3 MAC Layer LightWeight Filter-0000 | ffffe00 |

† Note that the IfIndex changes each time the network interface is started (or the computer is rebooted). Avoid using the IfIndex as a permanent identifier.

Вкладка "Потоки данных", которая отображается, если отслеживание NBL было включено в целевой системе, отображает таблицу событий трафика и подробные сведения о каждом из них. На следующем рисунке показана вкладка "Потоки данных" из активной системы во втором примере отчета об отладке, описанном ранее.



NETWORK DEBUG REPORT

OVERVIEW SYSTEM SUMMARIES DATA FLOWS Help

| Occurances | Weighted | Failure? | Event | Context | Source | Destination |
|---|---|---|---|---|---|---|
| 855 | 30.7% | | Freed | | | |
| 327 | 11.8% | | MiniportReceived | ffffe00bc53ae181 | | ffffe00bc5ac5c70 |
| | | | FilterReceived | 00000020 | | |
| | | | IndicatedToProtocol_Synchronous | ffffe00bc30abbb1 | | ffffe00bc30abc10 |
| | | | ProtocolReturned_Synchronous | ffffe00bc53ae181 | | ffffe00bc5ac5c70 |
| | | | IndicatedToProtocol | ffffe00bc5813971 | | ffffe00bc5ac0010 |
| | | | ProtocolReturned | ffffe00bc53ae181 | | ffffe00bc5ac5c70 |
| | | | FilterReturned | ffffe00bc51a7971 | | ffffe00bc3f701a0 |
| 314 | 11.3% | | Allocated | | | |
| | | | ProtocolSent | ffffe00bc4af2861 | ffffe00bc5ab7c10 | ffffe00bc5ac4880 |
| | | | FilterSent | ffffe00bc53ae181 | | ffffe00bc5ac5c70 |
| | | | FilterSent | ffffe00bc51a7971 | | ffffe00bc3f701a0 |
| | | | SentToMiniport | ffffe00bc51a7971 | | ffffe00bc3f701a0 |
| | | | MiniportSendCompleted | ffffe00bc53ae181 | | ffffe00bc5ac5c70 |
| | | | FilterSendCompleted | ffffe00bc4af2861 | | ffffe00bc5ac4880 |
| | | | FilterSendCompleted | 00000030 | | |
| | | | SendCompleted | ffffe00bc5ab88a1 | | ffffe00bc5ab7c10 |
| 297 | 10.7% | | MiniportReceived | ffffe00bc53ae181 | ffffe00bc3f701a0 | ffffe00bc5ac5c70 |
| | | | FilterReceived | 00000020 | | |
| | | | IndicatedToProtocol_Synchronous | ffffe00bc30abbb1 | | ffffe00bc30abc10 |
| | | | ProtocolReturned_Synchronous | ffffe00bc53ae181 | | ffffe00bc5ac5c70 |
| | | | IndicatedToProtocol | ffffe00bc5813971 | | ffffe00bc5ac0010 |
| | | | ProtocolReturned | ffffe00bc53ae181 | | ffffe00bc5ac5c70 |
| | | | FilterReturned | ffffe00bc51a7971 | | ffffe00bc3f701a0 |

# См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

**!ndiskd.nbllog**

Использование объединения сетевых карт для производительности сетевой подсистемы

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.cxadapter

Статья • 05.04.2024

Расширение **!ndiskd.cxadapter** отображает сведения об объекте NETADAPTER.

Дополнительные сведения о расширении класса WDF сетевого адаптера (NetAdapterCx) см. в разделе Расширения класса WDF сетевого адаптера (Cx).

Консоль

```
!ndiskd.cxadapter [-handle <x>] [-basic] [-power] [-datapath]
```

## Параметры

*-Обрабатывать*
Обязательный. Дескриптор NETADAPTER.

*-Основные*
Отображает основные сведения.

*-Мощность*
Отображает сведения об объекте NETPOWERSETTINGS netADAPTER.

*-datapath*
Отображает сведения о очередях datapath.

## DLL-библиотеки

Ndiskd.dll

## Примеры

Чтобы получить дескриптор для объекта NETADAPTER, сначала выполните **команду !ndiskd.netadapter** , чтобы просмотреть список всех драйверов сетевых адаптеров и NetAdapters в системе. В следующем примере найдите дескриптор netAdapter с именем Realtek PCIe GBE Family Controller NetAdapter Sample Driver #2. Его дескриптор — ffffd1022d048030.

Консоль

```
0: kd> !ndiskd.netadapter
    Driver             NetAdapter          Name
    ffffd1022e8ecae0   ffffd1022d048030    Realtek PCIe GBE Family
Controller NetAdapter Sample Driver #2
    ffffd1022ed908e0   ffffd1022e8611a0    Microsoft Kernel Debug Network
Adapter
```

Щелкнув этот дескриптор NetAdapter или введя **команду !ndiskd.netadapter -handle с его дескриптором** в командной строке, можно просмотреть сведения об этом NetAdapter, включая его объект NETADAPTER. Дескриптор NetAdapter для контроллера семейства NetAdapter в Realtek PCIe GBE — это дескриптор NETADAPTER 0002efd0e5f988.

Консоль

```
0: kd> !ndiskd.netadapter ffffd1022d048030


NETADAPTER

    Realtek PCIe GBE Family Controller NetAdapter Sample Driver #2

    Ndis handle        ffffd1022d048030
    NETADAPTER         00002efdd0e5f988    More information
    WDFDEVICE          00002efdcf45f2f8
    Driver             ffffd1022e8ecae0 - RtEthSample  v0.0
    Network interface  ffffd1022e395a20

    Media type         802.3
    Device instance
PCI\VEN_10EC&DEV_8168&SUBSYS_816810EC&REV_03\4&22bb23f1&0&0038
    Device object      ffffd1022de127f0    More information
    MAC address        00-e0-4c-68-00-8b


STATE

    Miniport           Running
    Device PnP         Started             Show state history
    Datapath           Normal
    Interface          Up
    Media              Connected
    Power              D0
    References         0n10                Show detail
    Total resets       0
    Pending OID        None
    Flags              NOT_BUS_MASTER, WDF, DEFAULT_PORT_ACTIVATED,
                       SUPPORTS_MEDIA_SENSE, DOES_NOT_DO_LOOPBACK,
                       MEDIA_CONNECTED
    PnP flags          PM_SUPPORTED, DEVICE_POWER_ENABLED,
                       DEVICE_POWER_WAKE_ENABLE, HARDWARE_DEVICE,
```

```
                        NDIS_WDM_DRIVER, WAKE_CAPABLE


IP ADDRESS SUMMARY

    10.137.188.169                          See all IP addresses on this
adapter
    fe80::3cad:81bb:5dad:1066


BINDINGS

    Protocol list      Driver              Open               Context
    MSLLDP             ffffd1023043f6a0    ffffd1022e786a90
ffffd102307465c0
    LLTDIO             ffffd1022c6b7830    ffffd1022ef8cc00
ffffd1022f1e5730
    TCPIP6             ffffd1022e2c7c10    ffffd10230b98310
ffffd102304d9010
    (RASPPPOE)         Not running
    (RDMANDK)          ffffd1022d574a70    Declined with
NDIS_STATUS_NOT_RECOGNIZED
    RSPNDR             ffffd1022c71a830    ffffd1022de0cc00
ffffd1022d03f6a0
    TCPIP              ffffd1022e2cbc10    ffffd1022de067f0
ffffd1022d03f010
    NDISUIO            ffffd1022de07670    ffffd1022cd648d0
ffffd10231131970

    Filter list        Driver              Module             Context
    WFP 802.3 MAC Layer LightWeight Filter-0000
                       ffffd1022e384d70    ffffd1022f271660
ffffd10230cfe2c0
    QoS Packet Scheduler-0000
                       ffffd1022d56f220    ffffd1022f26d660
ffffd10231778700
    WFP Native MAC Layer LightWeight Filter-0000
                       ffffd1022e384ad0    ffffd1022f26b660
ffffd1022ed59c20



MORE INFORMATION

    Driver handlers                        Task offloads
    Power management                       PM protocol offloads
    Pending OIDs
    Pending NBLs                           Receive side throttling
    Wake-on-LAN (WoL)                      Packet filter
    Receive queues                         Receive filtering
    RSS                                    NIC switch
                                           Selective suspend
    NDIS ports                             WMI guids
    Diagnostic log
```

Так как объект NETADAPTER является объектом WDF, щелкнув его дескриптор, отладчик **запустит команду !wdfkd.wdfhandle** , которая даст вам дополнительные сведения об этом с точки зрения WDF. Чтобы просмотреть более подробные сведения о NETADAPTER с точки зрения сети, щелкните ссылку "Дополнительные сведения" справа от дескриптора NETADAPTER, чтобы запустить **команду !ndiskd.cxadapter** с его дескриптором.

Консоль

```
0: kd> !ndiskd.cxadapter ffffd1022f1a0720


NETADAPTER

    Miniport           ffffd1022d048030 - Realtek PCIe GBE Family Controller
NetAdapter Sample Driver #2
    NETADAPTER         00002efdd0e5f988
    WDFDEVICE          00002efdcf45f2f8

    Event Callbacks                         Function pointer   Symbol (if
available)
    EvtAdapterCreateTxQueue                 fffff80034151508
RtEthSample+1508
    EvtAdapterCreateRxQueue                 fffff800341510ec
RtEthSample+10ec

    Show datapath info
    Show NETPOWERSETTINGS info
```

Вы также можете объединить эту команду с другими параметрами, такими как *-datapath* , чтобы просмотреть дополнительные сведения об этом NETADAPTER.

Консоль

```
0: kd> !ndiskd.cxadapter ffffd1022f1a0720 -basic -datapath


NETADAPTER

    Miniport           ffffd1022d048030 - Realtek PCIe GBE Family Controller
NetAdapter Sample Driver #2
    NETADAPTER         00002efdd0e5f988
    WDFDEVICE          00002efdcf45f2f8

    Event Callbacks                         Function pointer   Symbol (if
available)
    EvtAdapterCreateTxQueue                 fffff80034151508
RtEthSample+1508
    EvtAdapterCreateRxQueue                 fffff800341510ec
RtEthSample+10ec
```

```
DATAPATH QUEUES

    NETTXQUEUE          ffffd1022f512700
    NETRXQUEUE          ffffd1022cc7b0d0
```

## См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Расширение класса WDF сетевого адаптера (Cx)

**!ndiskd.netadapter**

**!wdfkd.wdfhandle**

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.netqueue

Статья • 05.04.2024

Расширение **!ndiskd.netqueue** отображает сведения о объекте NETTXQUEUE или NETRXQUEUE.

Дополнительные сведения о расширении класса WDF сетевого адаптера (NetAdapterCx) см. в разделе Расширения класса WDF сетевого адаптера (Cx).

Консоль

```
!ndiskd.netqueue -handle <x> [-basic]
```

# Параметры

*-Обрабатывать*
Обязательный. Дескриптор NETTXQUEUE или NETRXQUEUE.

*-Основные*
Отображает основные сведения.

## DLL-библиотеки

Ndiskd.dll

## Примеры

**Примечание. См.** сводку объектов, чтобы увидеть схему, объясняющую связь объектов NETTXQUEUE и NETRXQUEUE с другими объектами в NetAdapterCx.

Чтобы получить дескриптор для NETTXQUEUE или NETRXQUEUE, выполните следующие действия.

1. **Запустите расширение !ndiskd.netadapter**.
2. Щелкните дескриптор для NetAdapter, на котором установлен драйвер NetAdapterCx.
3. Щелкните ссылку "Дополнительные сведения" справа от объекта NETAdapter NETADAPTER, чтобы запустить расширение !ndiskd.cxadapter.
4. **Введите команду !ndiskd.cxadapter** с параметром *-datapath*, чтобы увидеть очереди пути к данным NETADAPTER.

Дополнительные сведения об этой процедуре см. в примерах **статьи !ndiskd.cxadapter** . В следующем примере найдите дескриптор для этого NETADAPTER NETTXQUEUE, ffffd1022f512700.

Консоль

```
0: kd> !ndiskd.cxadapter ffffd1022f1a0720 -basic -datapath


NETADAPTER

    Miniport           ffffd1022d048030 - Realtek PCIe GBE Family Controller
NetAdapter Sample Driver #2
    NETADAPTER         00002efdd0e5f988
    WDFDEVICE          00002efdcf45f2f8

    Event Callbacks                         Function pointer   Symbol (if
available)
    EvtAdapterCreateTxQueue                 fffff80034151508
RtEthSample+1508
    EvtAdapterCreateRxQueue                 fffff800341510ec
RtEthSample+10ec


DATAPATH QUEUES

    NETTXQUEUE         ffffd1022f512700
    NETRXQUEUE         ffffd1022cc7b0d0
```

Щелкнув дескриптор NETTXQUEUE или введя **команду !ndiskd.netqueue -handle в командной строке, вы можете просмотреть сведения об этой очереди, включая дескриптор объекта WDF-компаньона, дескриптор** в его кольцевой буфер и указатели функций для зарегистрированных обратных вызовов.

Консоль

```
0: kd> !ndiskd.netqueue ffffd1022f512700

    NETTXQUEUE         00002efdd0aed9a8
    Ring buffer        ffffd1022d000000

    Switch to EC thread

    Event Callbacks                         Function pointer   Symbol (if
available)
    EvtQueueAdvance                         fffff80034152af8
RtEthSample+2af8
    EvtQueueArmNotification                 fffff80034159a94
RtEthSample+9a94
```

```
    EvtQueueCancel                          fffff800341598d8
RtEthSample+98d8
```

## См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Расширение класса WDF сетевого адаптера (Cx)

Сводка объектов

**!ndiskd.netadapter**

**!ndiskd.cxadapter**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.netrb

Статья • 05.04.2024

Расширение **!ndiskd.netrb** отображает сведения о структуре NET_RING_BUFFER .

Дополнительные сведения о расширении класса WDF сетевого адаптера (NetAdapterCx) см. в разделе Расширения класса WDF сетевого адаптера (Cx).

Консоль

```
!ndiskd.netrb -handle <x> [-basic] [-dump] [-elementtype <str>]
```

## Параметры

*-Обрабатывать*
Обязательный. Адрес NET_RING_BUFFER.

*-Основные*
Отображает основные сведения.

*-Дампа*
Отображает сведения о каждом элементе в NET_RING_BUFFER.

*-elementtype*
Строка типа данных, используемая при обращении к элементу кольцевого буфера.

## DLL-библиотеки

Ndiskd.dll

## Примеры

**Примечание. См.** сводку объектов, чтобы увидеть схему, объясняющую связь объекта NET_RING_BUFFER с другими объектами в NetAdapterCx.

Чтобы получить дескриптор для NET_RING_BUFFER, выполните следующие действия.

1. **Запустите расширение !ndiskd.netadapter**.
2. Щелкните дескриптор для NetAdapter, на котором установлен драйвер NetAdapterCx.

3. Щелкните ссылку "Дополнительные сведения" справа от объекта NETAdapter NETADAPTER, чтобы запустить расширение !ndiskd.cxadapter.
4. **Введите команду !ndiskd.cxadapter** с параметром *-datapath*, чтобы увидеть очереди пути к данным NETADAPTER.
5. Щелкните дескриптор для одной из очередей datapath.

Дополнительные сведения о шагах 1-4 этой процедуры см. в примерах **статьи !ndiskd.cxadapter** . Дополнительные сведения о шаге 5 этой процедуры см. в примерах **статьи !ndiskd.netqueue** . В следующем примере найдите дескриптор для этого кольцевого буфера NETTXQUEUE, ffffd1022d0000000.

Консоль

```
0: kd> !ndiskd.netqueue ffffd1022f512700

    NETTXQUEUE          00002efdd0aed9a8
    Ring buffer         ffffd1022d000000

    Switch to EC thread

    Event Callbacks                         Function pointer   Symbol (if
available)
    EvtQueueAdvance                         fffff80034152af8
RtEthSample+2af8
    EvtQueueArmNotification                 fffff80034159a94
RtEthSample+9a94
    EvtQueueCancel                          fffff800341598d8
RtEthSample+98d8
```

Щелкнув дескриптор для кольцевого буфера или введя **команду !ndiskd.netrb -handle** в командной строке, можно просмотреть сведения об этом NET_RING_BUFFER, включая количество элементов, содержащихся в нем, и адрес его индексов Begin и End.

Консоль

```
0: kd> !ndiskd.netrb ffffd1022d000000

    NET_RING    ffffd1022d000000

    Number of elements 0x080
    Owned by OS        0x080
    Owned by Client    00000

    Begin Index        0x078 (ffffd1022d003c40 - NET_PACKET)
    Next Index         0x078 (ffffd1022d003c40 - NET_PACKET)
    End Index          0x078 (ffffd1022d003c40 - NET_PACKET)
```

```
    List all elements
```

Чтобы просмотреть элементы этого NET_RING_BUFFER, щелкните ссылку "Перечислить все элементы" в нижней части его сведений или введите **команду !ndiskd.netrb -dump** в командной строке. В следующем примере средние элементы были изъяты для краткости.

Консоль

```
0: kd> !ndiskd.netrb ffffd1022d000000 -dump

    [000] ffffd1022d000040 - NET_PACKET
    [001] ffffd1022d0000c0 - NET_PACKET
    [002] ffffd1022d000140 - NET_PACKET
    [003] ffffd1022d0001c0 - NET_PACKET
    [004] ffffd1022d000240 - NET_PACKET
    [005] ffffd1022d0002c0 - NET_PACKET

    ...

    [07b] ffffd1022d003dc0 - NET_PACKET
    [07c] ffffd1022d003e40 - NET_PACKET
    [07d] ffffd1022d003ec0 - NET_PACKET
    [07e] ffffd1022d003f40 - NET_PACKET
    [07f] ffffd1022d003fc0 - NET_PACKET
```

# См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Расширение класса WDF сетевого адаптера (Cx)

Сводка объектов

NET_RING_BUFFER

**!ndiskd.netadapter**

**!ndiskd.cxadapter**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.netpacket

16.08.2025

Расширение **!ndiskd.netpacket** отображает сведения о структуре NET_PACKET .

Дополнительные сведения о расширении класса WDF сетевого адаптера (NetAdapterCx) см. в разделе Расширения класса WDF сетевого адаптера (Cx).

Консоль

```
!ndiskd.netpacket -handle <x> [-basic] [-layout] [-checksum] [-data]
```

## Параметры

*-ручка*
Обязательное. Адрес NET_PACKET.

*-основной*
Отображает основные сведения.

*-схема*
Отображает макет протокола пакетов.

*-контрольная сумма*
Отображает сведения о контрольной сумме пакетов.

*-данные*
Дамп памяти полезных данных.

## Библиотека dll

Ndiskd.dll

## Примеры

**Заметка** См . сводку объектов , чтобы увидеть схему, объясняющую связь объекта NET_PACKET с другими объектами в NetAdapterCx.

Чтобы получить дескриптор для NET_PACKET, выполните следующие действия.

1. Запустите расширение **!ndiskd.netadapter** .

2. Щелкните дескриптор для NetAdapter, на котором установлен драйвер NetAdapterCx.
3. Щелкните ссылку "Дополнительные сведения" справа от объекта NETAdapter NETADAPTER, чтобы запустить расширение **!ndiskd.cxadapter** .
4. Введите команду **!ndiskd.cxadapter** с параметром *-datapath* , чтобы увидеть очереди пути к данным NETADAPTER.
5. Щелкните дескриптор для одной из очередей datapath.
6. Щелкните дескриптор для кольцевого буфера очереди datapath.
7. Щелкните ссылку "Перечислить все элементы" в нижней части кольцевого буфера, чтобы просмотреть элементы, которые он содержит.

Дополнительные сведения о шагах 1-4 этой процедуры см. в примерах статьи **!ndiskd.cxadapter** . Дополнительные сведения о шаге 5 этой процедуры см. в примерах статьи **!ndiskd.netqueue** . Дополнительные сведения о шагах 6-7 этой процедуры см. в примерах статьи **!ndiskd.netrb** . В следующем примере найдите дескриптор для первого NET_PACKET ffffd1022d000040.

Консоль

```
0: kd> !ndiskd.netrb ffffd1022d000000 -dump

    [000] ffffd1022d000040 - NET_PACKET
    [001] ffffd1022d0000c0 - NET_PACKET
    [002] ffffd1022d000140 - NET_PACKET
    [003] ffffd1022d0001c0 - NET_PACKET
    [004] ffffd1022d000240 - NET_PACKET
    [005] ffffd1022d0002c0 - NET_PACKET

    ...

    [07b] ffffd1022d003dc0 - NET_PACKET
    [07c] ffffd1022d003e40 - NET_PACKET
    [07d] ffffd1022d003ec0 - NET_PACKET
    [07e] ffffd1022d003f40 - NET_PACKET
    [07f] ffffd1022d003fc0 - NET_PACKET
```

Щелкнув дескриптор для этого NET_PACKET или введя **!ndiskd.netpacket -handle** в командной строке, вы можете просмотреть сведения об этом NET_PACKET, включая кольцевой буфер, содержащий его, очередь пути данных, содержащую его кольцевой буфер, и дескриптор для своего первого фрагмента.

Консоль

```
0: kd> !ndiskd.netpacket ffffd1022d000040


    NET_PACKET         ffffd1022d000040    Ring Buffer        ffffd1022d000000
```

```
    First fragment     ffffd1022d000040    NETTXQUEUE         ffffd1022f512700

    Client Context     ffffd1022d000090

    Show protocol layout
    Show checksum information
    Dump data payload
```

Теперь вы можете объединить базовое описание с любым из других параметров **!ndiskd.netpacket** или всех из них, чтобы просмотреть конкретные сведения для этого фрагмента. В следующем примере используются все параметры.

Консоль

```
0: kd> !ndiskd.netpacket ffffd1022d000040 -basic -layout -checksum -data

    NET_PACKET         ffffd1022d000040    Ring Buffer        ffffd1022d000000
    First fragment     ffffd1022d000040    NETTXQUEUE         ffffd1022f512700

    Client Context     ffffd1022d000090


    Protocol Layout

    Layer 2 Type       ETHERNET
    Header Length      0n14

    Layer 3 Type       IPV4_NO_OPTIONS
    Header Length      0n20

    Layer 4 Type       UDP
    Header Length      8


    Checksum Information

    Layer 2            TX_PASSTHROUGH
    Layer 3            TX_REQUIRED
    Layer 4            TX_PASSTHROUGH


    Payload data

    Fragment           ffffd1022d000040
    ffffd102303e8332  00 00 01 02 71 68 0a 89-be 39 e0 00 00 16 94 04
....qh...9......
    ffffd102303e8342  00 00 22 00 fa 01 00 00-00 01 03 00 00 00 e0 00
..".............
    ffffd102303e8352  00 fc
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Расширение класса WDF сетевого адаптера (Cx)

Сводка объектов

NET_PACKET

**!ndiskd.netadapter**

**!ndiskd.cxadapter**

**!ndiskd.netqueue**

**!ndiskd.netrb**

# !ndiskd.netfragment

Статья • 05.04.2024

Расширение **!ndiskd.netfragment** отображает сведения о структуре NET_PACKET_FRAGMENT .

Дополнительные сведения о расширении класса WDF сетевого адаптера (NetAdapterCx) см. в разделе Расширения класса WDF сетевого адаптера (Cx).

Консоль

```
!ndiskd.netfragment -handle <x>
```

## Параметры

*-Обрабатывать*
Обязательный. Адрес NET_PACKET_FRAGMENT.

## DLL-библиотеки

Ndiskd.dll

## Примеры

**Примечание. См.** сводку объектов, чтобы увидеть схему, объясняющую связь объекта NET_PACKET с другими объектами в NetAdapterCx.

Чтобы получить дескриптор для NET_PACKET, выполните следующие действия.

1. **Запустите расширение !ndiskd.netadapter**.
2. Щелкните дескриптор для NetAdapter, на котором установлен драйвер NetAdapterCx.
3. Щелкните ссылку "Дополнительные сведения" справа от объекта NETAdapter NETADAPTER, чтобы запустить расширение !ndiskd.cxadapter.
4. **Введите команду !ndiskd.cxadapter** с параметром *-datapath*, чтобы увидеть очереди пути к данным NETADAPTER.
5. Щелкните дескриптор для одной из очередей datapath.
6. Щелкните дескриптор для кольцевого буфера очереди datapath.
7. Щелкните ссылку "Перечислить все элементы" в нижней части кольцевого буфера, чтобы просмотреть элементы, которые он содержит.

8. Щелкните один из объектов NET_PACKET в списке элементов буфера кольца.

Дополнительные сведения о шагах 1-4 этой процедуры см. в примерах **статьи !ndiskd.cxadapter** . Дополнительные сведения о шаге 5 этой процедуры см. в примерах **статьи !ndiskd.netqueue** . Дополнительные сведения о шагах 6-7 этой процедуры см. в примерах **статьи !ndiskd.netrb** . Дополнительные сведения о шаге 8 этой процедуры см. в примерах **статьи !ndiskd.netpacket** . В следующем примере найдите дескриптор для первого фрагмента этого NET_PACKET, ffffd1022d000040.

Консоль

```
0: kd> !ndiskd.netpacket ffffd1022d000040


    NET_PACKET         ffffd1022d000040    Ring Buffer
ffffd1022d000000
    First fragment     ffffd1022d000040    NETTXQUEUE
ffffd1022f512700

    Client Context     ffffd1022d000090

    Show protocol layout
    Show checksum information
    Dump data payload
```

Щелкнув дескриптор первого фрагмента или введя **команду !ndiskd.netfragment -handle** в командной строке, вы можете просмотреть сведения об этом NET_PACKET_FRAGMENT, включая его виртуальный адрес, емкость и независимо от того, является ли он последним пакетом в цепочке фрагментов NET_PACKET.

Консоль

```
0: kd> !ndiskd.netfragment ffffd1022d000040

    NET_PACKET_FRAGMENT ffffd1022d000040

    Virtual Address    ffffd102303e82f8
    Capacity           0n92
    Valid Length       0n34
    Offset             0n58

    Last packet of chain
```

# См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Расширение класса WDF сетевого адаптера (Cx)

Сводка объектов

NET_PACKET_FRAGMENT

NET_PACKET

**!ndiskd.netadapter**

**!ndiskd.cxadapter**

**!ndiskd.netqueue**

**!ndiskd.netrb**

**!ndiskd.netpacket**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.nrc

Статья • 05.04.2024

Расширение **!ndiskd.nrc** отображает сведения о структуре NET_RING_COLLECTION .

Дополнительные сведения о расширении класса WDF сетевого адаптера (NetAdapterCx) см. в разделе "Расширение класса WDF сетевого адаптера" (Cx) и введение в круги сети.

Консоль

```
!ndiskd.nrc -handle <x> [-basic] [-packet] [-fragment] [-dump]
```

## Параметры

*-Обрабатывать*
Обязательный. Адрес NET_RING_COLLECTION

*-Основные*
Отображает ссылки на кольцо пакета и кольцо фрагмента.

*-Пакет*
Отображает только содержимое кольца пакетов.

*-Фрагмент*
Отображает только содержимое кольца фрагмента.

*-Дампа*
Отображает сведения о каждом элементе (пакете или фрагменте).

## DLL-библиотеки

Ndiskd.dll

## Примеры

**Примечание. См.** сводку объектов, чтобы увидеть схему, объясняющую связь объекта NET_PACKET с другими объектами в NetAdapterCx.

Чтобы получить дескриптор для NET_PACKET, выполните следующие действия.

1. **Запустите расширение !ndiskd.netadapter**.

2. Щелкните дескриптор для NetAdapter, на котором установлен драйвер NetAdapterCx.
3. Щелкните ссылку "Дополнительные сведения" справа от объекта NETAdapter NETADAPTER, чтобы запустить расширение !ndiskd.cxadapter.
4. **Введите команду !ndiskd.cxadapter** с параметром *-datapath*, чтобы увидеть очереди пути к данным NETADAPTER.
5. Щелкните дескриптор для одной из очередей datapath.
6. Щелкните дескриптор для кольцевого буфера очереди datapath.
7. Щелкните ссылку "Перечислить все элементы" в нижней части кольцевого буфера, чтобы просмотреть элементы, которые он содержит.
8. Щелкните один из объектов NET RING COLLECTION.

Дополнительные сведения о шагах 1-4 этой процедуры см. в примерах **статьи !ndiskd.cxadapter** . Дополнительные сведения о шаге 5 этой процедуры см. в примерах **статьи !ndiskd.netqueue** . Дополнительные сведения о шагах 6-7 этой процедуры см. в примерах **статьи !ndiskd.netrb** .

В следующем примере дескриптор для NET_RING_COLLECTION используется с командой nrc ffff8b82fbcf94b0.

dbgcmd

```
0: kd> !ndiskd.nrc fffff8b82fbcf94b0

NET RING COLLECTION ffff8b82fbcf94b0

Packet Ring         ffff8b82f8e75000
Fragment Ring       ffff8b8300961000

List only packets
List only fragments
List all elements
```

# См. также

Руководство по проектированию сетевого драйвера

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Расширение класса WDF сетевого адаптера (Cx)

Сводка объектов

NET_RING_COLLECTION

**!ndiskd.netadapter**

**!ndiskd.cxadapter**

**!ndiskd.netqueue**

**!ndiskd.netrb**

**!ndiskd.netpacket**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ndiskd.netring

16.08.2025

Расширение **!ndiskd.netring** отображает сведения о структуре NET_RING .

Дополнительные сведения о расширении класса WDF сетевого адаптера (NetAdapterCx) см. в разделе "Расширение класса WDF сетевого адаптера" (Cx) и введение в круги сети.

Консоль

```
!ndiskd.netring -handle <x> [-basic] [-dump]
```

## Параметры

*-ручка*
Обязательное. Адрес NET_RING

*-основной*
Отображение основных сведений

*-свалка*
Отображает сведения о каждом элементе

## Библиотека dll

Ndiskd.dll

## Примеры

**Заметка** См . сводку объектов , чтобы увидеть схему, объясняющую связь объекта NET_PACKET с другими объектами в NetAdapterCx.

Чтобы получить дескриптор для NET_PACKET, выполните следующие действия.

1. Запустите расширение **!ndiskd.netadapter** .
2. Щелкните дескриптор для NetAdapter, на котором установлен драйвер NetAdapterCx.
3. Щелкните ссылку "Дополнительные сведения" справа от объекта NETAdapter NETADAPTER, чтобы запустить расширение **!ndiskd.cxadapter** .
4. Введите команду **!ndiskd.cxadapter** с параметром *-datapath* , чтобы увидеть очереди пути к данным NETADAPTER.

5. Щелкните дескриптор для одной из очередей datapath.
6. Щелкните дескриптор для кольцевого буфера очереди datapath.
7. Щелкните ссылку "Перечислить все элементы" в нижней части кольцевого буфера, чтобы просмотреть элементы, которые он содержит.
8. Щелкните один из объектов Net Ring Collection.

Дополнительные сведения о шагах 1-4 этой процедуры см. в примерах статьи **!ndiskd.cxadapter** . Дополнительные сведения о шаге 5 этой процедуры см. в примерах статьи **!ndiskd.netqueue** . Дополнительные сведения о шагах 6-7 этой процедуры см. в примерах статьи **!ndiskd.netrb** .

В этом примере показано использование сетки.

Консоль

```
0: kd> !ndiskd.netring ffff8b82f8e75000

NET RING               ffff8b82f8e75000

Begin index            000
Next index             000
End index              0x2
Reserved by OS         000
Element stride         0x40
Element index mask     0x7f

Number of elements     0x80
Owned by OS            0x7e
Owned by Client        0x2
```

# См. также

Руководство по проектированию сетевого драйвера)

Справочник по сети Windows Vista и более поздних версий

Отладка сетевого стека

**Расширения NDIS (Ndiskd.dll)**

**!ndiskd.help**

Расширение класса WDF сетевого адаптера (Cx)

Сводка объектов

NET_RING

!ndiskd.netadapter

!ndiskd.cxadapter

!ndiskd.netqueue

!ndiskd.netrb

!ndiskd.netpacket

# Расширения RPC (Rpcexts.dll)

18.07.2025

Команды расширения, полезные для отладки вызова удаленной процедуры (RPC) Майкрософт, можно найти в Rpcexts.dll.

Библиотека DLL для Windows XP и более поздней версии этого расширения отображается в каталоге winxp.

Дополнительные сведения об использовании этих расширений см. в разделе "Использование расширений отладчика RPC".

# !rpcexts.eeinfo

Статья • 05.04.2024

Расширение **!rpcexts.eeinfo** отображает расширенную цепочку сведений об ошибках.

```
dbgcmd

!rpcexts.eeinfo EEInfoAddress
```

## Параметры

*EEInfoAddress*
Указывает адрес расширенной информации об ошибке.

## DLL-библиотеки

Rpcexts.dll

## Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

## Замечания

Это расширение отображает содержимое всех записей в расширенной цепочке сведений об ошибках.

Записи отображаются в порядке, при этом последние записи сначала отображаются. Записи разделены линией дефисов.

Ниже приведен пример (в котором существует только одна запись):

```
dbgcmd

0:001> !rpcexts.eeinfo 0xb015f0
Computer Name: (null)
ProcessID: 708 (0x2C4)
System Time is: 3/21/2000 4:3:0:264
Generating component: 8
```

```
Status: 14
Detection Location: 311
Flags:
Parameter 0:(Long value) : -30976 (0xFFFF8700)
Parameter 1:(Long value) : 16777343 (0x100007F)
```

Если цепочка очень длинна, и вы хотите увидеть только одну запись, используйте **!rpcexts.eerecord** .

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.eerecord

Статья • 05.04.2024

Расширение **!rpcexts.eerecord** отображает содержимое расширенной записи сведений об ошибках.

dbgcmd

```
!rpcexts.eerecord EERecordAddress
```

## Параметры

*EERecordAddress*
Указывает адрес расширенной записи об ошибке.

## DLL-библиотеки

Rpcexts.dll

## Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

## Замечания

Это расширение отображает содержимое одной расширенной записи сведений об ошибках в отладчике. В большинстве случаев проще использовать **!rpcexts.eeinfo**, который отображает всю цепочку. Если цепочка очень длинна, и вы хотите увидеть только одну запись, используйте **!eerecord** вместо этого.

Рассмотрим пример:

dbgcmd

```
0:001> !rpcexts.eerecord 0xb015f0
Computer Name: (null)
ProcessID: 708 (0x2C4)
System Time is: 3/21/2000 4:3:0:264
Generating component: 8
Status: 14
```

```
Detection Location: 311
Flags:
Parameter 0:(Long value) : -30976 (0xFFFF8700)
Parameter 1:(Long value) : 16777343 (0x100007F)
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.getcallinfo

Статья • 05.04.2024

Расширение **!rpcexts.getcallinfo** выполняет поиск сведений о состоянии RPC системы для сведений о состоянии вызова на стороне сервера (SCALL).

dbgcmd

```
!rpcexts.getcallinfo [ CallID | 0 [ IfStart | 0 [ ProcNum | 0xFFFF
[ProcessID|0] ] ] ]
!rpcexts.getcallinfo -?
```

## Параметры

*CallID*
Указывает идентификатор вызова. Этот параметр является необязательным; включите его, если требуется отображать только вызовы, соответствующие определенному *значению CallID* .

*IfStart*
Указывает первый DWORD интерфейса UUID, на котором был выполнен вызов. Этот параметр является необязательным; включите его, если требуется отображать только вызовы, соответствующие определенному *значению IfStart* .

*ProcNum*
Указывает номер процедуры этого вызова. (Время выполнения RPC определяет отдельные подпрограммы из интерфейса путем нумерирования их по расположению в файле IDL - первая подпрограмма в интерфейсе — 0, вторая 1 и т. д.)

*ProcessID*
Указывает идентификатор процесса (PID) сервера, который владеет вызовами, которые необходимо отобразить. Этот параметр является необязательным; опустить его, если требуется отобразить вызовы, принадлежащие нескольким процессам.

**-?**
Отображает краткий текст справки для этого расширения в окне командной строки.

## DLL-библиотеки

# Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

# Замечания

Это расширение можно использовать только с CDB или с WinDbg в пользовательском режиме.

Параметры анализируются слева направо. Чтобы пропустить параметр, укажите значение 0. Существует одно исключение из этого правила: *параметр ProcNum* пропускается путем предоставления значения 0xFFFF.

Рассмотрим пример:

dbgcmd

```
0:002> !rpcexts.getcallinfo
Searching for call info ...
## PID  CELL ID   ST PNO IFSTART  TIDNUMBER CALLFLAG CALLID   LASTTIME
CONN/CLN
-------------------------------------------------------------------------------
00c4 0000.0006 00 009 367abb81 0000.0007 00000001 0000004a 00018b41
0000.0005
00c4 0000.000a 00 007 367abb81 0000.002d 00000001 0000009f 000134ff
0000.0009
00c4 0000.000d 00 00f 82273fdc 0000.002d 00000001 00000002 00036cd8
0000.0042
00c4 0000.0010 00 00f 367abb81 0000.002d 00000001 00000078 00011636
0000.000f
00c4 0000.0012 00 00d 8d9f4e40 0000.0007 00000001 0000004f 000097bd
0000.0011
00c4 0000.0015 00 000 367abb81 0000.0004 00000001 0000004c 0002cccf
0000.0014
00c4 0000.0017 00 007 367abb81 0000.0004 00000001 00000006 0000cf5e
0000.0016
00c4 0000.0018 00 000 367abb81 0000.002d 00000001 0000000b 0001236f
0000.002a
00c4 0000.0019 01 00b 82273fdc 0000.0002 00000009 00000000 00018b19
00d0.0104
00c4 0000.001b 00 009 65a93890 0000.0007 00000001 000000ea 0003cd14
0000.001a
00c4 0000.0021 00 03b 8d9f4e40 0000.0013 00000001 0000000b 0001162c
0000.0020
00c4 0000.0022 01 008 82273fdc 0000.001f 00000009 00000000 00013405
00c4.02e8
```

```
00c4 0000.0024 00 007 367abb81 0000.0004 00000001 00000006 0000f198
0000.0023
00c4 0000.0026 00 000 367abb81 0000.0036 00000001 000000ab 00038049
0000.0025
00c4 0000.0027 01 00b 82273fdc 0000.001f 00000009 00000000 00020b7c
00a8.0228
00c4 0000.0028 01 008 82273fdc 0000.003e 00000009 00000000 0003a949
0294.02f0
00c4 0000.0029 00 00d 8d9f4e40 0000.002d 00000001 0000033f 0003831a
0000.0031
00c4 0000.0030 00 03b 8d9f4e40 0000.0013 00000001 00000002 00024e43
0000.002f
00c4 0000.0032 01 008 82273fdc 0000.001f 00000009 00000000 000118f3
022c.019c
00c4 0000.0035 00 007 367abb81 0000.0033 00000001 00000074 0001042d
0000.0034
00c4 0000.0038 00 007 367abb81 0000.002d 00000001 0000000a 0002a3e4
0000.0037
00c4 0000.003a 00 007 367abb81 0000.0036 00000001 00000063 0003b7b8
0000.0039
00c4 0000.003b 00 004 3ba0ffc0 0000.002d 00000001 00000005 0002dd79
0000.002e
00c4 0000.003f 01 008 82273fdc 0000.0002 00000009 00000000 000245c6
01c0.037c
00c4 0000.0043 01 008 82273fdc 0000.0002 00000009 00000000 00037d50
020c.0394
00c4 0000.0049 00 008 8d9f4e40 0000.0007 00000001 000002b1 0004e900
0000.0048
0170 0000.0009 01 002 e60c73e6 0000.0002 00000009 baadf00d 0004ad30
020c.03a4
0170 0000.000a 01 002 0b0a6584 0000.0008 00000009 baadf00d 0001187b
00c4.012c
0170 0000.000c 01 002 0b0a6584 0000.0008 00000009 baadf00d 00011cdc
022c.019c
0170 0000.000d 01 003 00000136 0000.0011 00000009 baadf00d 00034845
020c.02b4
0170 0000.000e 01 000 412f241e 0000.0002 00000009 baadf00d 00012491
0294.02b8
0170 0000.000f 01 002 0b0a6584 0000.0011 00000009 baadf00d 000492e7
026c.0118
0170 0000.0010 01 002 e60c73e6 0000.0013 00000009 baadf00d 0004ab78
0378.038c
0170 0000.0014 01 004 e60c73e6 0000.0011 00000001 baadf00d 0002bc25
0378.024c
0170 0000.0015 01 003 00000136 0000.0013 00000009 00000003 00031d8d
0378.00b8
0170 0000.0018 01 004 00000136 0000.0002 00000001 baadf00d 00032e05
020c.026c
020c 0000.0004 01 003 00000132 0000.000b 00000009 00000000 00034953
0170.0240
020c 0000.000e 01 001 2f5f6520 0000.001e 00000009 00120006 00035bac
020c.03b4
020c 0000.0010 01 000 629b9f66 0000.000f 00000009 00000000 000279ff
00a8.0194
020c 0000.0011 01 004 faedcf59 0000.0003 00000009 00000012 0003836b
```

```
0378.024c
020c 0000.0012 01 001 629b9f66 0000.000f 00000009 00000000 0003657e
020c.02ec
020c 0000.0017 01 005 00000134 0000.0002 00000001 00000016 0003836b
0378.024c
020c 0000.001d 01 001 2f5f6520 0000.0014 00000001 0020007d 000351b2
020c.0258
0294 0000.0004 01 004 00000132 0000.0002 00000009 00000000 0003b786
0170.01ac
0378 0000.0004 01 003 00000134 0000.0003 0000000b 00300038 0002d896
020c.021c
026c 0000.0004 02 000 19bb5061 0000.0002 00000001 00000001 0004caa5
0000.0003
```

Аналогичный пример использования средства DbgRpc см. в разделе "Получение сведений о вызове RPC".

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.getclientcallinfo

Статья • 05.04.2024

Расширение **!rpcexts.getclientcallinfo** выполняет поиск сведений о состоянии RPC системы для получения сведений о вызове клиента (CCALL).

dbgcmd

```
!rpcexts.getclientcallinfo [ CallID | 0 [ IfStart | 0 [ ProcNum | 0xFFFF
[ProcessID|0] ] ] ]
!rpcexts.getclientcallinfo -?
```

## Параметры

*CallID*
Указывает идентификатор вызова. Этот параметр является необязательным; включите его, если требуется отображать только вызовы, соответствующие определенному *значению CallID* .

*IfStart*
Указывает первый DWORD интерфейса UUID, на котором был выполнен вызов. Этот параметр является необязательным; включите его, если требуется отображать только вызовы, соответствующие определенному *значению IfStart* .

*ProcNum*
Указывает номер процедуры этого вызова. (Время выполнения RPC определяет отдельные подпрограммы из интерфейса путем нумерирования их по расположению в файле IDL - первая подпрограмма в интерфейсе — 0, вторая 1 и т. д.) Этот параметр является необязательным; включите его, если требуется отображать только вызовы, соответствующие определенному *значению ProcNum* .

*ProcessID*
Указывает идентификатор процесса (PID) клиентского процесса, который владеет вызовами, которые необходимо отобразить. Этот параметр является необязательным; опустить его, если требуется отобразить вызовы, принадлежащие нескольким процессам.

**-?**
Отображает краткий текст справки для этого расширения в окне командной строки.

# DLL-библиотеки

Rpcexts.dll

# Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

# Замечания

Это расширение можно использовать только с CDB или с WinDbg в пользовательском режиме. Он доступен только в том случае, если собираются полные сведения о состоянии RPC.

Рассмотрим пример:

dbgcmd

```
0:002> !rpcexts.getclientcallinfo
Searching for call info ...
## PID  CELL ID   PNO  IFSTART  TIDNUMBER CALLID   LASTTIME PS CLTNUMBER
ENDPOINT
------------------------------------------------------------------------------
--
03d4 0000.0001 0000 19bb5061 0000.0000 00000001 0004ca9b 07 0000.0002 1118
```

Аналогичный пример использования средства DbgRpc см. в разделе "Получение сведений о вызове клиента RPC".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.getdbgcell

Статья • 05.04.2024

Расширение **!rpcexts.getdbgcell** отображает сведения о состоянии RPC для указанной ячейки.

```
dbgcmd
```

```
!rpcexts.getdbgcell ProcessID CellID1.CellID2
!rpcexts.getdbgcell -?
```

## Параметры

*ProcessID*
Указывает идентификатор процесса (PID) процесса, сервер которого содержит нужную ячейку.

*CellID1.CellID2*
Указывает число отображаемой ячейки.

**-?**
Отображает краткий текст справки для этого расширения в окне командной строки.

## DLL-библиотеки

Rpcexts.dll

## Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

## Замечания

Это расширение можно использовать только с CDB или с WinDbg в пользовательском режиме.

Рассмотрим пример:

```
dbgcmd
```

```
0:002> !rpcexts.getdbgcell c4 0.19
Getting cell info ...
Call
Status: Active
Procedure Number: 11
Interface UUID start (first DWORD only): 82273FDC
Call ID: 0x0 (0)
Servicing thread identifier: 0x0.3E
Call Flags: cached, LRPC
Last update time (in seconds since boot):1453.459 (0x5AD.1CB)
Caller (PID/TID) is: d0.1ac (208.428)
```

Аналогичный пример использования средства DbgRpc см. в разделе "Получение сведений о ячейках RPC".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.getendpointinfo

Статья • 05.04.2024

Расширение **!rpcexts.getendpointinfo** ищет сведения о состоянии RPC системы для получения сведений о конечной точке.

dbgcmd

```
    !rpcexts.getendpointinfo [EndpointName]
!rpcexts.getendpointinfo -?
```

## Параметры

*EndpointName*
Указывает число отображаемых конечных точек. Если опущено, отображаются конечные точки для всех процессов в системе.

**-?**
Отображает краткий текст справки для этого расширения в окне командной строки.

## DLL-библиотеки

Rpcexts.dll

## Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

## Замечания

Это расширение можно использовать только с CDB или с WinDbg в пользовательском режиме.

Рассмотрим пример:

dbgcmd

```
0:002> !rpcexts.getendpointinfo
Searching for endpoint info ...
```

```
## PID  CELL ID   ST PROTSEQ        ENDPOINT
-------------------------------------------------------------
00a8 0000.0001 01            NMP \PIPE\InitShutdown
00a8 0000.0003 01            NMP \PIPE\SfcApi
00a8 0000.0004 01            NMP \PIPE\ProfMapApi
00a8 0000.0005 01           LRPC OLE5
00a8 0000.0007 01            NMP \pipe\winlogonrpc
00c4 0000.0001 01           LRPC ntsvcs
00c4 0000.0003 01            NMP \PIPE\ntsvcs
00c4 0000.0008 01            NMP \PIPE\scerpc
00d0 0000.0001 01            NMP \PIPE\lsass
00d0 0000.0003 01            NMP \pipe\WMIEP_d0
00d0 0000.0006 01           LRPC policyagent
00d0 0000.0007 01            NMP \PIPE\POLICYAGENT
0170 0000.0001 01           LRPC epmapper
0170 0000.0003 01            TCP 135
0170 0000.0005 01            SPX 34280
0170 0000.0006 01             NB 135
0170 0000.0007 01             NB 135
0170 0000.000b 01            NMP \pipe\epmapper
01c0 0000.0001 01            NMP \pipe\spoolss
01c0 0000.0003 01           LRPC spoolss
01c0 0000.0007 01           LRPC OLE7
020c 0000.0001 01           LRPC OLE2
020c 0000.0005 01           LRPC senssvc
020c 0000.0007 01            NMP \pipe\tapsrv
020c 0000.0009 01           LRPC tapsrvlpc
020c 0000.000c 01            NMP \PIPE\ROUTER
020c 0000.0016 01            NMP \pipe\WMIEP_20c
0218 0000.0001 01            NMP \PIPE\winreg
022c 0000.0001 01           LRPC LRPC0000022c.00000001
022c 0000.0003 01            TCP 1041
022c 0000.0005 01            SPX 24576
022c 0000.0006 01            NMP \PIPE\atsvc
0294 0000.0001 01           LRPC OLE3
0378 0000.0001 01           LRPC OLE9
026c 0000.0001 01            TCP 1118
0344 0000.0001 01           LRPC OLE12
```

Аналогичный пример использования средства DbgRpc см. в разделе "Получение сведений о конечной точке RPC".

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.getthreadinfo

Статья • 05.04.2024

Расширение **!rpcexts.getthreadinfo** ищет сведения о состоянии RPC системы для получения сведений о потоке.

dbgcmd

```
!rpcexts.getthreadinfo ProcessID [ThreadID]
!rpcexts.getthreadinfo -?
```

## Параметры

*ProcessID*
Указывает идентификатор процесса (PID) процесса, содержащего нужный поток.

*ThreadID*
Указывает идентификатор потока для отображения. Если опущено, будут отображаться все потоки в указанном процессе.

**-?**
Отображает краткий текст справки для этого расширения в окне командной строки.

## DLL-библиотеки

Rpcexts.dll

## Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

## Замечания

Это расширение можно использовать только с CDB или с WinDbg в пользовательском режиме.

Рассмотрим пример:

dbgcmd

```
0:002> !rpcexts.getthreadinfo 26c
Searching for thread info ...
## PID  CELL ID   ST TID      LASTTIME
-----------------------------------
026c 0000.0002 01 000003c4 0004caa5
026c 0000.0005 03 00000254 0004ca9b
```

Аналогичный пример использования средства DbgRpc см. в разделе "Получение сведений о потоке RPC".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.help

Статья • 05.04.2024

Расширение **!rpcexts.help** отображает текст справки в окне командной строки со всеми командами расширения Rpcexts.dll.

dbgcmd

```
!rpcexts.help
```

## DLL-библиотеки

Rpcexts.dll

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.rpcreadstack

Статья • 05.04.2024

Расширение **!rpcexts.rpcreadstack** считывает стек на стороне клиента RPC и извлекает сведения о вызове.

dbgcmd

```
!rpcexts.rpcreadstack ThreadStackPointer
```

## Параметры

*ThreadStackPointer*
Указывает указатель на стек потоков.

## DLL-библиотеки

Rpcexts.dll

## Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

## Замечания

Общие сведения об использовании этого расширения см. в разделе "Анализ проблемы с застрявших вызовов".

Рассмотрим пример:

dbgcmd

```
0:001> !rpcexts.rpcreadstack 68fba4
CallID: 1
IfStart: 19bb5061
ProcNum: 0
        Protocol Sequence:      "ncacn_ip_tcp"  (Address: 00692ED8)
        NetworkAddress: ""      (Address: 00692F38)
        Endpoint:       "1120"  (Address: 00693988)
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.rpctime

Статья • 05.04.2024

Расширение !rpcexts.rpctime **отображает текущее** системное время.

```
dbgcmd
```

```
!rpcexts.rpctime
```

## DLL-библиотеки

Rpcexts.dll

## Замечания

Это расширение можно использовать только с CDB или с WinDbg в пользовательском режиме.

Рассмотрим пример:

```
dbgcmd
```

```
0:001> !rpcexts.rpctime
Current time is: 059931.126  (0x00ea1b.07e)
```

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rpcexts.thread

Статья • 05.04.2024

Расширение **!rpcexts.thread** отображает сведения О RPC для каждого потока.

Эта команда расширения не должна путаться с командой **.thread (Set Register Context)** или **расширением !thread** (!kdextx86.thread и !kdexts.thread).

dbgcmd

```
!rpcexts.thread TEB
```

## Параметры

*TEB*
Указывает адрес блока среды потока (TEB).

## DLL-библиотеки

Rpcexts.dll

## Дополнительная информация

Дополнительные сведения об отладке вызова удаленной процедуры Майкрософт (RPC) см. в разделе "Отладка RPC".

## Замечания

Это расширение отображает сведения О RPC для каждого потока. Поле в сведениях RPC для каждого потока — это расширенные сведения об ошибках для этого потока.

Рассмотрим пример:

dbgcmd

```
0:001> !rpcexts.thread 7ffdd000
RPC TLS at 692e70

HandleToThread - 0x6c
SavedProcedure - 0x0
```

```
SavedParameter - 0x0
ActiveCall - 0x0
Context - 0x0
CancelTimeout - 0xffffffff
SecurityContext - 0x0
ExtendedStatus - 0x0
ThreadEEInfo - 0xb015f0
ThreadEvent at - 0x00692E78
fCallCancelled - 0x0
buffer cache array at - 0x00692E84
fAsync - 0x0
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения ACPI (Acpikd.dll и Kdexts.dll)

18.07.2025

Команды расширения, которые полезны для отладки кода BIOS ACPI (advanced Configuration and Power Interface) можно найти в Acpikd.dll и Kdexts.dll.

Некоторые из расширений отладки ACPI можно найти в\Acpikd.dll, while others can be found in Winxp\Kdexts.dllWinxp.

# !acpicache

Статья • 05.04.2024

Расширение **!acpicache** отображает все таблицы расширенной конфигурации и интерфейса управления питанием (ACPI), кэшированные hal.

dbgcmd

```
!acpicache [DisplayLevel]
```

## Параметры

*DisplayLevel*
Указывает уровень детализации дисплея. Это значение равно 0 для сокращенного дисплея или 1 для более подробного отображения. Значение по умолчанию равно 0.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения об ACPI см. в документации по пакету драйверов Microsoft Windows (WDK), документации по пакету SDK для Windows и *внутренним* службам Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном. Также см . отладку ACPI для получения сведений о других расширениях, связанных с ACPI.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !acpiinf

Статья • 05.04.2024

Расширение **!acpiinf** отображает сведения о конфигурации расширенной конфигурации и интерфейса питания (ACPI), включая расположение системных таблиц и содержимое оборудования фиксированной функции ACPI.

dbgcmd

```
!acpiinf
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения об ACPI см. в документации по пакету драйверов Microsoft Windows (WDK), документации по пакету SDK для Windows и *внутренним* службам Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном. Также см . отладку ACPI для получения сведений о других расширениях, связанных с ACPI.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !acpiirqarb

Статья • 05.04.2024

Расширение **!acpiirqarb** отображает содержимое структуры arbiter IRQ advanced Configuration and Power Interface (ACPI), которая содержит конфигурацию устройств ввода-вывода для системных входных данных контроллера прерываний и записей таблицы отправки прерываний процессора (IDT).

dbgcmd

```
!acpiirqarb
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения об ACPI см. в документации по пакету драйверов Microsoft Windows (WDK), документации по пакету SDK для Windows и *внутренним* службам Microsoft Windows Марком Руссиновичем и Дэвидом Соломоном. Также см . отладку ACPI для получения сведений о других расширениях, связанных с ACPI.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !acpikd.help

Статья • 07.02.2024

Расширение **!acpikd.help** отображает текст справки в окне Команда отладчика, в котором отображаются все команды расширения Acpikd.dll.

dbgcmd

```
!acpikd.help
```

# DLL

Acpikd.dll

## Дополнительные сведения

Дополнительные сведения см. в разделе Отладка ACPI.

# !amli ?

Статья • 08.07.2024

Расширение **!amli ?** отображает текст справки в окне команды команд отладчика для **команд расширения !amli** .

Синтаксис

```
dbgcmd

    !amli ? [Command]
```

## Параметры

*Команда*
Указывает **команду !amli** , справка которой должна отображаться. Например, **набор !amli ? отображает справку по команде !amli set.** Если *команда* опущена, отображается список всех команд.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli bc

Статья • 08.07.2024

Расширение **!amli bc** окончательно очищает точку останова AML.

Синтаксис

dbgcmd

```
!amli bc Breakpoint
!amli bc *
```

## Параметры

*Контрольная точка* Указывает число точек останова, которые необходимо очистить.

* Указывает, что все точки останова должны быть очищены.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Чтобы определить номер точки останова точки останова, используйте **расширение !amli bl** .

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli bd

Статья • 08.07.2024

Расширение **!amli bd** временно отключает точку останова AML.

Синтаксис

dbgcmd

```
    !amli bd Breakpoint!amli bd *
```

## Параметры

*Контрольная точка* Указывает количество отключаемой точки останова.

* Указывает, что все точки останова должны быть отключены.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Отключенную точку останова можно повторно включить с помощью **расширения !amli** .

Чтобы определить номер точки останова точки останова, используйте **расширение !amli bl** .

Ниже приведен пример этой команды:

Консоль

```
kd> !amli bl
 0:  c29accf5 [\_WAK]
 1:  c29c20a5 [\_SB.PCI0.ISA.BAT1._BST]
```

```
kd> !amli bd 1
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli be

Статья • 08.07.2024

Расширение **!amli включает** точку останова AML.

Синтаксис

```
dbgcmd

    !amli be Breakpoint!amli be *
```

## Параметры

*Контрольная точка* Указывает номер точки останова для включения точки останова.

* Указывает, что все точки останова должны быть включены.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Все точки останова включены при создании. Точки останова отключены только в том случае, если вы использовали **расширение !amli bd** .

Чтобы определить номер точки останова точки останова, используйте **расширение !amli bl** .

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli bl

Статья • 01.02.2024

Расширение **!amli bl** отображает список всех точек останова AML.

Синтаксис

dbgcmd

```
!amli bl
```

# DLL

Kdexts.dll

## Дополнительные сведения

Сведения о связанных командах и их использовании см. в разделе Отладчик amli.

# Комментарии

Отладчик !amli поддерживает не более десяти точек останова.

Ниже приведен пример расширения **!amli bl** :

Консоль

```
kd> !amli bl
 0: <e> ffffffff80e5e2f1:[\_SB.LNKD._SRS]
 1: <e> ffffffff80e5d969:[\_SB.LNKB._STA]
 2: <d> ffffffff80e630c9:[\_WAK]
 3: <e> ffffffff80e612c9:[\_SB.MBRD._CRS]
```

Первый столбец дает номер точки останова. Знаки **<e>** и **<d>** указывают, включена или отключена точка останова. Адрес точки останова указан в следующем столбце. Наконец, отображается метод, содержащий точку останова, со смещением точки останова, если она не задана в начале метода.

# !amli bp

Статья • 08.07.2024

Расширение **!amli bp** помещает точку останова в код AML.

Синтаксис

dbgcmd

```
!amli bp { MethodName | CodeAddress }
```

## Параметры

*Имя метода*
Указывает полный путь к имени метода, в котором будет задана точка останова.

*CodeAddress*
Указывает адрес кода AML, в котором будет задана точка останова. Если *CodeAddress префиксируется* двумя знаками (%%), он интерпретируется как физический адрес. В противном случае он интерпретируется как виртуальный адрес.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Отладчик AMLI поддерживает не более 10 точек останова.

Ниже приведен пример. Следующая команда устанавливает точку останова в методе _DCK:

Консоль

```
kd> !amli bp \_sb.pci0.dock._dck
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli cl

Статья • 11.11.2023

Расширение **!amli cl** очищает журнал событий интерпретатора AML.

Синтаксис

```
dbgcmd

    !amli cl
```

## DLL

Kdexts.dll

## Дополнительные сведения

Сведения о связанных командах и их использовании см. в разделе Отладчик AMLI.

# Отладчик !amli

Статья • 07.02.2024

Расширение **отладчика !amli** разрывается на отладчик AMLI.

Синтаксис

```
dbgcmd

    !amli debugger
```

# DLL

Kdexts.dll

## Дополнительные сведения

Сведения о связанных командах и их использовании см. в разделе Отладчик AMLI.

# Комментарии

При выполнении этой команды в интерпретатор AML отправляется уведомление. При следующей активности интерпретатора он немедленно войдет в отладчик AMLI.

Расширение **отладчика !amli** вызывает только один перерыв. Если вы хотите снова прервать работу, необходимо снова использовать это расширение или установить точку останова.

# !amli dh

Статья • 08.07.2024

Расширение **!amli dh** отображает внутренний блок кучи интерпретатора AML.

Синтаксис

```
dbgcmd

!amli dh [HeapAddress]
```

## Параметры

*КучиAddress*
Указывает адрес блока кучи. Если это опущено, отображается глобальная куча.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli dl

Статья • 11.11.2023

Расширение **!amli dl** отображает часть журнала событий интерпретатора AML.

Синтаксис

dbgcmd

```
    !amli dl
```

## DLL

Kdexts.dll

## Дополнительные сведения

Сведения о связанных командах и их использовании см. в разделе Отладчик AMLI.

# Комментарии

В журнале событий перечислены последние 150 событий, произошедших в интерпретаторе.

Ниже приведен пример отображения журнала:

Консоль

```
kd> !amli dl
RUN!: [c15a6618]QTh=00000000,QCt=00000000,QFg=00000000: Ctx=c18b4000,rc=0
KICK: [c15a6618]QTh=00000000,QCt=00000000,QFg=00000000: rc=0
SYNC:
[c15a6618]QTh=00000000,QCt=00000000,QFg=00000002,LockPhase=0,Locked=0,IRQL=0
0: Obj=\_WAK
ASYN:
[c15a6618]QTh=00000000,QCt=00000000,QFg=00000002,LockPhase=0,Locked=0,IRQL=0
0: Obj=\_WAK
REST: [c15a6618]QTh=00000000,QCt=00000000,QFg=00000002:
Ctx=c18b4000,Obj=\_WAK
INSQ: [c15a6618]QTh=00000000,QCt=00000000,QFg=00000002:
Ctx=c18b4000,Obj=\_WAK
EVAL: [c15a6618]QTh=00000000,QCt=00000000,QFg=00000002:
Ctx=c18b4000,Obj=\_WAK
```

```
RUNC: [c15a6618]QTh=c15a6618,QCt=c18b4000,QFg=00000002:
Ctx=c18b4000,Obj=\_WAK
```

# !amli dns

Статья • 08.07.2024

Расширение **!amli dns** отображает объект пространства имен ACPI.

Синтаксис

```
dbgcmd

    !amli dns [/s] [Name | Address]
```

## Параметры

**/s**
Вызывает рекурсивное отображение всего поддерев пространства имен в указанном объекте.

*Имя*
Указывает путь к пространству имен.

*Адрес*
Указывает адрес узла пространства имен.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Если ни имя, *ни адрес* не указан, отображается все дерево пространства имен ACPI рекурсивно. Параметр **/s** всегда предполагается в этом случае, даже если он не указан.

Эта команда полезна для определения того, какой объект пространства имен является методом, единицей поля, устройством или другим типом объекта.

**Без параметра /s** это расширение эквивалентно расширению **!nsobj**. **При использовании параметра /s** он эквивалентен расширению **!nstree**.

Ниже приведено несколько примеров. Следующая команда отображает пространство имен для bios **объекта**:

Консоль

```
AMLI(? for help)-> dns \bios

ACPI Name Space: \BIOS (80E5F378)
OpRegion(BIOS:RegionSpace=SystemMemory,Offset=0xfcb07500,Len=2816)
```

Следующая команда отображает пространство имен для объекта _BST и дерево, подчиненное ему:

Консоль

```
kd> !amli dns /s \_sb.pci0.isa.bat1._bst

ACPI Name Space: \_SB.PCI0.ISA.BAT1._BST (c29c2044)
Method(_BST:Flags=0x0,CodeBuff=c29c20a5,Len=103)
```

Чтобы отобразить пространство имен для устройства BAT1, введите следующее:

Консоль

```
kd> !amli dns /s \_sb.pci0.isa.bat1
```

Чтобы отобразить пространство имен всех подчиненных устройству DOCK, введите следующее:

Консоль

```
kd> !amli dns /s \_sb.pci0.dock
```

Чтобы отобразить пространство имен, подчиненное методу _DCK, введите:

Консоль

```
kd> !amli dns /s \_sb.pci0.dock._dck
```

Чтобы отобразить все пространство имен, введите следующее:

Консоль

```
kd> !amli dns
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli do

Статья • 08.07.2024

Расширение **!amli do** отображает объект данных AML.

Синтаксис

```
dbgcmd

    !amli do Address
```

## Параметры

*Адрес*
Указывает адрес объекта данных.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli ds

Статья • 08.07.2024

Расширение **!amli ds** отображает стек AML.

Синтаксис

dbgcmd

```
!amli ds [/v] [Address]
```

## Параметры

**/v**
Приводит к детализации отображения.

*Адрес*
Указывает адрес блока контекста, стек которого необходим. Если *адрес* опущен, используется текущий контекст.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli find

Статья • 08.07.2024

Расширение **!amli find** находит объект пространства имен ACPI.

Синтаксис

dbgcmd

```
    !amli find Name
```

## Параметры

*Имя*
Указывает имя объекта пространства имен (без пути).

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Команда **!amli find** принимает имя объекта и возвращает полный путь и имя. Параметр *Name* должен быть окончательным сегментом полного пути и имени.

Ниже приведено несколько примеров. Следующая команда найдет все объявления объекта _SRS:

Консоль

```
kd> !amli find _srs
\_SB.LNKA._SRS
\_SB.LNKB._SRS
\_SB.LNKC._SRS
\_SB.LNKD._SRS
```

Это не просто текстовый поиск. Команда **!amli find srs** не отображает никаких попаданий, так как окончательный сегмент каждого из этих объявлений является "_SRS", а не "SRS". Команда **!amli найти LNK** аналогично не возвращает попадания. **Команда !amli find LNKB** будет отображать один узел, завершающийся в LNKB, а не четыре дочерних элемента этого узла, показанные на предыдущем экране:

Консоль

```
kd> !amli find lnkb
\_SB.LNKB.
```

Если нужно просмотреть дочерние элементы узла, используйте команду !amli dns **с параметром /s.**

Ниже приведен еще один пример, выданный из запроса отладчика AMLI. Здесь показаны все объявления объекта, _BST в пространстве имен:

Консоль

```
AMLI(? for help)-> find _bst
\_SB.PCI0.ISA.BAT1._BST
\_SB.PCI0.ISA.BAT2._BST
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli lc

Статья • 11.11.2023

Расширение **!amli lc** выводит список всех активных контекстов ACPI.

Синтаксис

dbgcmd

```
!amli lc
```

## DLL

Kdexts.dll

## Дополнительные сведения

Сведения о связанных командах и их использовании см. в разделе Отладчик AMLI.

# Комментарии

Каждый контекст соответствует методу, который в настоящее время выполняется в интерпретаторе AML.

Вот пример:

Консоль

```
AMLI(? for help)-> lc
 Ctxt=80e3f000, ThID=00000000, Flgs=A--C-----, pbOp=00000000,
Obj=\_SB.LNKA._STA
 Ctxt=80e41000, ThID=00000000, Flgs=A--C-----, pbOp=00000000,
Obj=\_SB.LNKB._STA
 Ctxt=80e9a000, ThID=00000000, Flgs=A--C-----, pbOp=00000000,
Obj=\_SB.LNKC._STA
 Ctxt=80ea8000, ThID=00000000, Flgs=A--C-----, pbOp=00000000,
Obj=\_SB.LNKD._STA
*Ctxt=80e12000, ThID=80e6eda8, Flgs=---CR----, pbOp=80e5d5ac,
Obj=\_SB.LNKA._STA
```

В поле **Obj** указан полный путь и имя метода, которые отображаются в таблицах ACPI.

Поле **Ctxt** предоставляет адрес блока контекста. Звездочка (\) *указывает на *текущий контекст*. Это контекст, который выполнялся интерпретатором при прерывании.

Аббревиатура **pbOp** обозначает указатель инструкции (указатель на двоичные коды операций).

В разделе Flgs можно отобразить девять **флагов** . Если флаг не задан, вместо него отображается дефис. Полный список флагов выглядит следующим образом:

| **Flag** | **Значение** |
|---|---|
| A | Асинхронная оценка |
| Нет | Вложенная оценка |
| Q | В очереди готовности |
| C | Требуется обратный вызов |
| R | Запущен |
| W | Ready |
| T | Время ожидания |
| D | Диспетчеризация таймера |
| P | Ожидание таймера |

# !amli ln

Статья • 08.07.2024

Расширение **!amli ln** отображает указанный метод или метод, содержащий заданный адрес.

Синтаксис

dbgcmd

```
!amli ln [ MethodName | CodeAddress ]
```

## Параметры

*Имя метода*
Указывает полный путь к имени метода. Если *MethodName* указывает объект, который на самом деле не является методом, результаты ошибки.

*CodeAddress*
Указывает адрес кода AML, содержащегося в нужном методе. Если *CodeAddress* *префиксируется* двумя знаками (%%), он интерпретируется как физический адрес. В противном случае он интерпретируется как виртуальный адрес.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Если параметр MethodName *или CodeAddress не* указан, отображается метод, связанный с текущим контекстом.

Следующая команда показывает, что в данный момент выполняется метод:

Консоль

```
kd> !amli ln
c29accf5: \_WAK
```

Показан метод _WAK с адресом 0xC29ACCF5.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli r

Статья • 08.07.2024

Расширение **!amli r** отображает сведения о текущем контексте или указанном контексте.

Синтаксис

dbgcmd

```
!amli r [ContextAddress]
```

## Параметры

*ContextAddress*
Указывает адрес отображаемого блока контекста. Адрес блока контекста можно определить из **поля Ctxt** в **дисплее !amli lc**. Если *ContextAddress префиксируется* с двумя знаками процента (%%), он интерпретируется как физический адрес. В противном случае он интерпретируется как виртуальный адрес. Если этот параметр опущен, отображается текущий контекст.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Если внезапно появится строка отладчика AMLI, это полезная команда для использования.

Например, следующая команда отобразит текущий контекст интерпретатора:

Консоль

```
AMLI(? for help)-> r

Context=c18b4000*, Queue=00000000, ResList=00000000
ThreadID=c15a6618, Flags=00000010
StackTop=c18b5eec, UsedStackSize=276 bytes, FreeStackSize=7636 bytes
LocalHeap=c18b40c0, CurrentHeap=c18b40c0, UsedHeapSize=88 bytes
Object=\_WAK, Scope=\_WAK, ObjectOwner=c18b4108, SyncLevel=0
AsyncCallBack=ff06b5d0, CallBackData=0, CallBackContext=c99efddc

MethodObject=\_WAK
80e0ff5c: Local0=Unknown()
80e0ff70: Local1=Unknown()
80e0ff84: Local2=Unknown()
80e0ff98: Local3=Unknown()
80e0ffac: Local4=Unknown()
80e0ffc0: Local5=Unknown()
80e0ffd4: Local6=Unknown()
80e0ffe8: Local7=Unknown()
80e0e040: RetObj=Unknown()

Next AML Pointer: ffffffff80e630df:[\_WAK+16]

ffffffff80e630df : If(S4BW
ffffffff80e630e5 : {
ffffffff80e630e5 : | Store(Zero, S4BW)
ffffffff80e630eb : }
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !amli set

Статья • 11.11.2023

Расширение **!amli set** задает или отображает параметры отладчика AMLI.

```
dbgcmd

    !amli set Options
```

# Параметры

*Параметры* Задает один или несколько параметров, которые необходимо задать. Разделите несколько параметров пробелами. Возможные значения:

**spewon**
Вызывает отправку полных выходных данных отладки с целевого компьютера. Этот параметр следует постоянно включать для эффективной отладки AML. Подробные сведения см. в разделе "Заметки".

**spewoff**
Подавляет выходные данные отладки.

**verboseon**
Включает подробный режим. Это приводит к тому, что отладчик AMLI отображает имена методов AML по мере их оценки.

**verboseoff**
Отключает подробный режим.

**Traceon**
Активирует трассировку ACPI. Это дает гораздо больше выходных данных, чем параметр **verboseon** . Этот параметр очень полезен для отслеживания жестких зависаний, связанных с SMI.

**traceoff**
Деактивирует трассировку ACPI.

**nesttraceon**
Активирует трассировку вложенных файлов. Этот параметр действует только в том случае, если выбран параметр **traceon** .

**dbgbrkon**
Включает возможность взлома отладчика AMLI.

**dbgbrkoff**
Деактивирует параметр dbgbrkon.

**nesttraceoff**
Деактивирует трассировку вложенных вложений.

**lbrkon**
Прерывает работу отладчика AMLI после завершения загрузки DDB.

**lbrkoff**
Деактивирует параметр **lbrkon** .

**errbrkon**
Прерывает работу отладчика AMLI всякий раз, когда интерпретатору возникают проблемы при оценке кода AML.

**errbrkoff**
Деактивирует параметр **errbrkon** .

**Входа**
Включает ведение журнала событий.

**Выхода**
Отключает ведение журнала событий.

**logmuton**
Включает ведение журнала событий мьютекса.

**logmutoff**
Отключает ведение журнала событий мьютекса.

## DLL

Kdexts.dll

## Дополнительные сведения

Сведения о связанных командах и их использовании см. в разделе Отладчик AMLI.

# Комментарии

Если параметры не указаны, отображается текущее состояние всех параметров.

По умолчанию многие сообщения отфильтровываются. Может потребоваться включить эти выходные данные с **параметром !amli set spewon**. В противном случае будут потеряны многочисленные сообщения отладчика AMLI.

Если интерпретатор AML врывается в отладчик AMLI, эти выходные данные будут автоматически включены.

Дополнительные сведения об этой фильтрации выходных данных см. в разделе **DbgPrintEx** и **KdPrintEx** в документации по комплекту драйверов Windows (WDK).

# !amli u

Статья • 08.07.2024

Код **AML расширения !amli u** unassembles.

Синтаксис

```
dbgcmd

    !amli u [ MethodName | CodeAddress ]
```

## Параметры

*Имя метода*
Указывает полный путь к имени метода, который необходимо расчленить.

*CodeAddress*
Указывает адрес кода AML, в котором начнется дизассембли. Если *CodeAddress* *префиксируется* двумя знаками (%%), он интерпретируется как физический адрес. В противном случае он интерпретируется как виртуальный адрес.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Сведения о связанных командах и их использовании см. в разделе "Отладчик AMLI".

## Замечания

Если параметр MethodName *или CodeAddress не* указан, и вы выдаете эту команду из AMLI.

Отображение дизассембли будет продолжаться до конца метода.

**Обратите внимание** , что команда standard **u (Unassemble)** не даст правильные результаты с кодом AML.

Ниже приведено несколько примеров. Чтобы дизассемблировать объект по адресу 0x80E5D701, используйте следующую команду:

Консоль

```
kd> !amli u 80e5d701

ffffffff80e5d701 : CreateWordField(CRES, 0x1, IRQW)
ffffffff80e5d70c : And(\_SB_.PCI0.LPC_.PIRA, 0xf, Local0)
ffffffff80e5d723 : Store(One, Local1)
ffffffff80e5d726 : ShiftLeft(Local1, Local0, IRQW)
ffffffff80e5d72d : Return(CRES)
```

Следующая команда будет дизассемблировать метод _DCK:

Консоль

```
kd> u \_sb.pci0.dock._dck
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !facs

Статья • 05.04.2024

Расширение **!facs** отображает структуру управления ACPI встроенного ПО (FACS).

Синтаксис

dbgcmd

```
!facs Address
```

## Параметры

*Адрес*
Указывает адрес FACS.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка ACPI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !fadt

Статья • 05.04.2024

Расширение **!fadt** отображает фиксированную таблицу описания ACPI (FADT).

Синтаксис

dbgcmd

```
!fadt Address
```

## Параметры

*Адрес*
Указывает адрес FADT.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка ACPI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !Mapic

Статья • 05.04.2024

Расширение **!mapic** отображает таблицу ACPI Multiple APIC.

Синтаксис

```
dbgcmd

!mapic Address
```

## Параметры

*Адрес*
Задает адрес таблицы APIC.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка ACPI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !nsobj

Статья • 05.04.2024

Расширение **!nsobj** отображает объект пространства имен ACPI.

Синтаксис

```
dbgcmd

!nsobj [Address]
```

## Параметры

*Адрес*
Указывает адрес объекта пространства имен. Если это не указано, используется корень дерева пространства имен.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка ACPI".

## Замечания

Это расширение эквивалентно **!amli dns**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !nstree

Статья • 05.04.2024

Расширение **!nstree** отображает объект пространства имен ACPI и его дочерние элементы в дереве пространства имен.

Синтаксис

```
dbgcmd

!nstree [Address]
```

## Параметры

*Адрес*
Указывает адрес объекта пространства имен. Этот объект и все дерево пространства имен, подчиненные ему, будут отображаться. Если *адрес* опущен, отображается все дерево пространства имен.

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка ACPI".

## Замечания

Это расширение эквивалентно **!amli dns /s**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !rsdt

Статья • 05.04.2024

Расширение **!rsdt** отображает таблицу описания корневой системы ACPI.

Синтаксис

```
dbgcmd
```

```
!rsdt
```

## DLL-библиотеки

Kdexts.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка ACPI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения потоковой передачи ядра (Ks.dll)

18.07.2025

Команды расширения, которые полезны для отладки драйверов потоковой передачи ядра и драйверов AVStream, можно найти в Ks.dll.

Большая часть выходных данных примера на этих эталонных страницах была создана путем отладки примера, ориентированного на фильтр, Avssamp.sys. Этот пример включен в комплект драйверов Windows.

Для использования этого расширения требуются специальные символы.

Дополнительные сведения для многих команд расширения в этом разделе можно получить, просто введя команду в отладчик без аргументов.

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

# !ks.help

Статья • 05.04.2024

Расширение **!ks.help** отображает текст справки, в котором отображаются все команды расширения, Ks.dll относящиеся к AVStream.

dbgcmd

```
!ks.help
```

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.kshelp

Статья • 05.04.2024

Расширение **!ks.kshelp** отображает текст справки с исходными командами расширения KS 1.Ks.dll 0.

dbgcmd

```
!ks.kshelp
```

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.pchelp

Статья • 05.04.2024

Расширение **!ks.pchelp** отображает текст справки с командами расширения portCls для конкретных Ks.dll.

dbgcmd

```
!ks.pchelp
```

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.allstreams

Статья • 05.04.2024

Расширение **!ks.allstreams** проходит весь дерево устройств и находит каждое устройство потоковой передачи ядра в системе.

dbgcmd

```
!ks.allstreams [Flags] [Level]
```

## Параметры

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x1:

Bit 0 (0x1)
Приводит к тому, что отображение будет включать потоки.

Бит 2 (0x4)
Приводит к включению экземпляров прокси-сервера.

Бит 3 (0x8)
Приводит к тому, что отображение включает в себя очереди IRP.

Бит 4 (0x10)
Приводит к включению неформатированного отображения всех потоков.

Бит 5 (0x20)
Приводит к включению неформатированного отображения всех форматов потока.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |

| | |
|---|---|
| **Windows XP и более поздние версии** | Ks.dll |

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

# Замечания

Эта команда может занять некоторое время для выполнения (минута не является необычной).

Ниже приведен пример **отображения !ks.allstreams** :

dbgcmd

```
kd> !allstreams
6 Kernel Streaming FDOs found:
    Functional Device 82a17690 [\Driver\smwdm]
    Functional Device 8296eb08 [\Driver\wdmaud]
    Functional Device 82490388 [\Driver\sysaudio]
    Functional Device 82970cb8 [\Driver\MSPQM]
    Functional Device 824661b8 [\Driver\MSPCLOCK]
    Functional Device 8241c020 [\Driver\avssamp]
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.automation

Статья • 05.04.2024

Расширение **!ks.automation отображает все элементы автоматизации** , связанные с заданным объектом.

dbgcmd

```
!ks.automation Object
```

## Параметры

*Объект*
Указывает указатель на объект, для которого будут отображаться элементы автоматизации. (Элементы автоматизации — это свойства, методы и события.) *Объект* должен быть одним из следующих типов: PKSPIN, PKSFILTER, CKsPin*, CKsFilter*, PIRP. Если *объект* является указателем на IRP службы автоматизации, команда возвращает сведения о свойствах и обработчики.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Эту команду можно использовать с адресом фильтра, полученным из **!ks.enumdevobj**.

Ниже приведен пример **отображения !ks.automation** . Аргумент — это адрес фильтра:

dbgcmd

```
kd> !automation 829493c4
Filter 829493c4 has the following automation items:
    Property Items:
        Set KSPROPSETID_Pin
            Item ID = KSPROPERTY_PIN_CINSTANCES
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000008
            Item ID = KSPROPERTY_PIN_CTYPES
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000018
                MinData = 00000004
            Item ID = KSPROPERTY_PIN_DATAFLOW
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000004
            Item ID = KSPROPERTY_PIN_DATARANGES
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000000
            Item ID = KSPROPERTY_PIN_DATAINTERSECTION
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000028
                MinData = 00000000
            Item ID = KSPROPERTY_PIN_INTERFACES
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000000
            Item ID = KSPROPERTY_PIN_MEDIUMS
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000000
            Item ID = KSPROPERTY_PIN_COMMUNICATION
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000004
            Item ID = KSPROPERTY_PIN_NECESSARYINSTANCES
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000004
            Item ID = KSPROPERTY_PIN_CATEGORY
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
```

```
                MinProperty = 00000020
                MinData = 00000010
            Item ID = KSPROPERTY_PIN_NAME
                Get Handler = ks!CKsFilter::Property_Pin
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000000
        Set KSPROPSETID_Topology
            Item ID = KSPROPERTY_TOPOLOGY_CATEGORIES
                Get Handler = ks!CKsFilter::Property_Topology
                Set Handler = NULL
                MinProperty = 00000018
                MinData = 00000000
            Item ID = KSPROPERTY_TOPOLOGY_NODES
                Get Handler = ks!CKsFilter::Property_Topology
                Set Handler = NULL
                MinProperty = 00000018
                MinData = 00000000
            Item ID = KSPROPERTY_TOPOLOGY_CONNECTIONS
                Get Handler = ks!CKsFilter::Property_Topology
                Set Handler = NULL
                MinProperty = 00000018
                MinData = 00000000
            Item ID = KSPROPERTY_TOPOLOGY_NAME
                Get Handler = ks!CKsFilter::Property_Topology
                Set Handler = NULL
                MinProperty = 00000020
                MinData = 00000000
        Set KSPROPSETID_General
            Item ID = KSPROPERTY_GENERAL_COMPONENTID
                Get Handler = ks!CKsFilter::Property_General_ComponentId
                Set Handler = NULL
                MinProperty = 00000018
                MinData = 00000048
        Set [ks!KSPROPSETID_Frame]  a60d8368-5324-4893-b020-c431a50bcbe3
            Item ID = 0
                Get Handler = ks!CKsFilter::Property_Frame_Holding
                Set Handler = ks!CKsFilter::Property_Frame_Holding
                MinProperty = 00000018
                MinData = 00000004
    Method Items:
        NO SETS FOUND!
    Event Items:
        NO SETS FOUND!
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.devhdr

Статья • 05.04.2024

Расширение **!ks.devhdr** отображает заголовок устройства потоковой передачи ядра, связанный с заданным объектом WDM.

```
dbgcmd
```

```
!ks.devhdr DeviceObject
```

## Параметры

*DeviceObject*
Этот параметр указывает указатель на объект устройства WDM. Если *DeviceObject* недопустим, команда возвращает ошибку.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Выходные данные из !ks.allstreams **можно использовать в качестве входных данных для !ks.devhdr.**

Ниже приведен пример **отображения !ks.devhdr** :

```
dbgcmd
```

```
kd> !devhdr 827aedf0 7
Device Header 824ca1e0
    Child Create Handler List:
```

```
        Create Item eb3a7284
            CreateFunction =
sysaudio!CFilterInstance::FilterDispatchCreate+0x00
            ObjectClass = NULL
            Flags = 0
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.dump

Статья • 05.04.2024

Расширение **!ks.dump** отображает указанный объект.

```
dbgcmd
```

```
!ks.dump Object [Level] [Flags]
```

# Параметры

*Объект*
Указывает указатель на структуру AVStream, объект класса AVStream или объект PortCls. Можно также указать указатель на IRP или объект файла.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7. Дополнительные сведения об уровнях можно просмотреть, выполнив **команду !ks.dump** без аргументов.

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов.

Bit 0 (0x1)
Отображение всех очередей IRP.

Бит 1 (0x2)
Отображение всех ожидающих IRP.

Бит 2 (0x4)
Анализ остановленного графа для подозреваемых.

Бит 3 (0x8)
Отображение всех состояний закреплений.

# DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

# Замечания

Команда **!ks.dump** распознает большинство объектов AVStream, включая закрепления, фильтры, фабрики, устройства, каналы и указатели потоков. Эта команда также распознает некоторые структуры классов потоков, включая объекты потока, экземпляры фильтров, расширения устройств и SRB.

Ниже приведен пример **отображения !ks.dump** для фильтра:

dbgcmd

```
kd> !dump 829493c4
Filter object 829493c4 [CKsFilter = 82949350]
    Descriptor     f7a233c8:
    Context        829dce28
```

Ниже приведен пример **отображения !ks.dump** для пин-кода:

dbgcmd

```
kd> !dump 8160DDE0 7
Pin object 8160DDE0 [CKsPin = 8160DD50]
    DeviceState    KSSTATE_RUN
    ClientState    KSSTATE_RUN
    ResetState     KSRESET_END
    CKsPin object 8160DD50 [KSPIN = 8160DDE0]
        State                   KSSTATE_RUN
        Processing Mutex        8160DFD0 is not held
        And Gate &              8160DF88
        And Gate Count          1
```

Некоторые важные части этого дисплея включены в следующую таблицу.

Развернуть таблицу

| Параметр | Значение |
| --- | --- |
| DeviceState | Состояние ввода пин-кода. Если отличается от ClientState, это состояние, на которое будет переходить мини-driver. |
| ClientState | Состояние, в которое на самом деле находится мини-driver. Это отражает состояние канала. |
| ResetState | Указывает, находится ли объект в середине очистки.<br><br>KSRESET_BEGIN указывает на смыв.<br><br>KSRESET_END указывает на отсутствие очистки. |
| State | Внутреннее состояние транспорта пин-кода в фильтры, отличные от AVStream. |

Ниже приведен пример **отображения !ks.dump** для драйвера класса потока:

dbgcmd

```
kd> !dump 81a0a170 7
Device Extension 81a0a228:
    Device Object          81a0a170 [\Driver\TESTCAP]
    Next Device Object     81bd56d8 [\Driver\PnpManager]
    Physical Device Object 81bd56d8 [\Driver\PnpManager]
    REGISTRY FLAGS:
        Page out driver when closed
        No suspend if running
    MINIDRIVER Data:
        Device Extension       81a0a44c
        Interrupt Routine      00000000
        Synchronize Routine    STREAM!StreamClassSynchronizeExecution
        Receive Device SRB     testcap!AdapterReceivePacket
        Cancel Packet          testcap!AdapterCancelPacket
        Timeout Packet         testcap!AdapterTimeoutPacket
        Size (d / r / s / f)   1a0(416), 14(20), 978(2424), 0(0)
        Sync Mode              Driver Synchronizes
    Filter Type 0:
        Symbolic Links:
            Information Paged Out
        Instances:
            816b7bd8
```

Обратите внимание, что размеры перечислены как в шестнадцатеричных числах, так и в десятичном эквиваленте. Сокращенные значения размера в этом отображении перечислены в следующей таблице.

Развернуть таблицу

| Размер | Описание |
| --- | --- |
| d | Устройство |
| r | Запросить |
| s | Stream |
| f | Фильтр. Если размер фильтра равен 0, фильтр является одним экземпляром. Если оно больше 0, оно является несколькими экземплярами. |

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.dumpbag

Статья • 05.04.2024

Расширение **!ks.dumpbag** отображает содержимое контейнера объектов для указанного объекта.

```
dbgcmd
```

```
!ks.dumpbag Object [Level]
```

## Параметры

*Объект*
Указывает указатель на допустимую структуру объекта, доступную для просмотра клиента, или объект частного класса.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Ниже приведен пример **отображения !ks.dumpbag** для фильтра:

```
dbgcmd
```

```
kd> !dumpbag 829493c4
Filter 829493c4 [CKsFilter = 82949350]:
    Object Bag 829493d0:
        Object Bag Item 829dce28:
            Reference Count        : 1
            Item Cleanup Handler   : f7a21730
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.dumpcircuit

Статья • 05.04.2024

Расширение **!ks.dumpcircuit** содержит сведения о канале транспорта, связанном с заданным объектом.

dbgcmd

```
!ks.dumpcircuitextension Object [Level]
```

## Параметры

*Объект*
Указывает указатель на объект, для которого будет отображаться канал транспорта. Для AVStream *объект* должен быть одним из следующих типов: CKsPin*, CKsQueue*, CKsRequestor*, CKsSplitter*, CKsSplitterBranch*.

Для PortCls объект должен быть одним из следующих типов: CPortPin*, CKsShellRequestor*или CIrpStream*.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Обратите внимание, что **!ks.dumpcircuit** начинает ходить по каналу по указанному объекту, который не всегда соответствует источнику данных.

Сначала можно использовать !ks.graph **с адресом фильтра для перечисления пин-адресов, а затем использовать эти адреса с !ks.dumpcircuit.**

Ниже приведен пример **отображения !ks.dumpcircuit** :

dbgcmd

```
kd> !dumpcircuit 8293f4f0
Pin8293f4f0 0 (snk, out)
Queue82990e20 r/w/c=2489/2/0
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.dumplog

Статья • 05.04.2024

Расширение **!ks.dumplog** отображает внутренний журнал отладки потоковой передачи ядра.

dbgcmd

```
!ks.dumplog [Entries]
```

## Параметры

*Записи*
Необязательно. Указывает количество отображаемых записей журнала. Если *записи* равны нулю или опущены, отображается весь журнал.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Вы можете остановить отображение журнала, нажав клавиши **CTRL+C**.

Для этого расширения требуется, чтобы целевой компьютер работал с проверка (отладочной) версией Ks.sys.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.dumpqueue

Статья • 05.04.2024

Расширение **!ks.dumpqueue** отображает сведения о очередях, связанных с заданным объектом AVStream, или потоком, связанным с объектом класса порта.

dbgcmd

```
!ks.dumpqueue Object [Level]
```

## Параметры

*Объект*
Указывает указатель на объект, для которого отображается очередь. *Объект* должен иметь тип PKSPIN, PKSFILTER, CKsPin*, CKsFilter*, CKsQueue*, CPortPin*или CPortFilter*.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

*Объект* должен быть фильтром или закреплением. Для пин-кода отображается одна очередь. Для фильтра отображаются несколько очередей.

Эта команда может занять некоторое время.

Ниже приведен пример **отображения !ks.dumpqueue** :

dbgcmd

```
kd> !dumpqueue 829493c4
Filter 829493c4: Output Queue 82990e20:
 Queue 82990e20:
        Frames Received  : 1889
        Frames Waiting   : 3
        Frames Cancelled : 0
        And Gate 82949464 : count = 1, next = 00000000
    Frame Gate NULL
        Frame Header 82aaef78:
 NextFrameHeaderInIrp = 00000000
            OriginalIrp = 82169e48
            Mdl = 8292e358
            Irp = 82169e48
 StreamHeader = 8298dea0
            FrameBuffer = edba3000
            StreamHeaderSize = 00000000
            FrameBufferSize = 00025800
            Context = 00000000
            Refcount = 1
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.enumdevobj

Статья • 05.04.2024

Расширение **!ks.enumdevobj** отображает KSDEVICE, связанное с заданным объектом устройства WDM, и перечисляет типы фильтров и фильтры, созданные в настоящее время на этом устройстве.

dbgcmd

```
!ks.enumdevobj DeviceObject
```

## Параметры

*DeviceObject*
Указывает указатель на объект устройства WDM.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Выходные данные из !ks.allstreams **можно использовать в качестве входных данных для !ks.enumdevobj.**

Ниже приведен пример **отображения !ks.enumdevobj** :

dbgcmd

```
kd> !enumdevobj 8241c020
WDM device object 8241c020:
    Corresponding KSDEVICE        823b8430
```

```
    Factory 829782dc [Descriptor f7a233c8] instances:
        829493c4
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.enumdrvobj

Статья • 05.04.2024

Расширение **!ks.enumdrvobj** отображает все структуры KSDEVICE, связанные с заданным объектом драйвера WDM, и перечисляет типы фильтров и фильтры, созданные на этих устройствах.

dbgcmd

```
!ks.enumdrvobj DriverObject
```

## Параметры

*DriverObject*
Указывает указатель на объект драйвера WDM.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Так как !ks.enumdrvobj **перечисляет каждое устройство, подключенное от объекта драйвера WDM, это эквивалентно вызову !ks.enumdevobj на каждом устройстве, подключенном от заданного драйвера.**

---

## Обратная связь

**Были ли сведения на этой странице полезными?**

Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.eval

Статья • 05.04.2024

Расширение **!ks.eval** вычисляет выражение с помощью вычислителя выражений для конкретного расширения.

```
dbgcmd
```

```
!ks.eval Expression
```

## Параметры

*Выражение*
Указывает выражение для вычисления. *Выражение* может включать любые операторы MASM, символы или числовый синтаксис, а также операторы, относящиеся к расширению, описанные ниже. Дополнительные сведения о выражениях MASM см. в разделе "Номера и операторы MASM".

### DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Модуль расширения включает два оператора расширения, которые можно использовать в параметрах адреса для команд расширения:

**fdo**(*x*)
Возвращает функциональный объект устройства, связанный с объектом по адресу **x**.

**driver**(*x*)
Возвращает объект драйвера, связанный с **fdo**(*x*).

Команду !ks.eval можно использовать **для анализа выражений, содержащих эти операторы расширения, а также** номера и операторы **MASM.**

Обратите внимание, что все операторы, поддерживаемые **!ks.eval** , также поддерживаются всеми другими командами расширения в модуле расширения Ks.dll.

Ниже приведен пример **расширения !ks.eval** , используемого с адресом фильтра. Обратите внимание на наличие адреса 0x8241C020 в выходных данных !ks.allstreams:

dbgcmd

```
kd> !eval fdo(829493c4)
Resulting Evaluation: 8241c020

kd> !allstreams
6 Kernel Streaming FDOs found:
    Functional Device 82a17690 [\Driver\smwdm]
    Functional Device 8296eb08 [\Driver\wdmaud]
    Functional Device 82490388 [\Driver\sysaudio]
    Functional Device 82970cb8 [\Driver\MSPQM]
    Functional Device 824661b8 [\Driver\MSPCLOCK]
    Functional Device 8241c020 [\Driver\avssamp]
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.findlive

Статья • 05.04.2024

**Расширение !ks.findlive** выполняет поиск внутреннего Ks.sys журнала отладки, чтобы попытаться найти динамические объекты указанного типа.

```
dbgcmd
```

```
!ks.findlive Type [Entries] [Level]
```

## Параметры

*Тип*
Указывает тип объекта, для которого выполняется поиск. Введите одно из следующих значений: **Queue, Requestor, Pin, Filter или Irp**.

Записи
Если этот параметр равен нулю или опущен, выполняется поиск по всему журналу.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Команда **!ks.findlive** может не найти все возможные указанные динамические объекты.

Для этого расширения требуется, чтобы целевой компьютер работал с проверка (отладочной) версией Ks.sys.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.forcedump

Статья • 05.04.2024

Команда **!ks.forcedump** отображает сведения о содержимом памяти по указанному вызывающим адресу.

dbgcmd

```
!ks.forcedump Object Type [Level]
```

## Параметры

*Объект*
Указывает указатель на объект, для которого будут отображаться сведения.

*Тип*
Указывает тип объекта.

Для объектов *AVStream/KS тип* должен быть одним из следующих значений: CKsQueue, CKsDevice, CKsFilterFactory, CKsFilterFactory, CKsFilter, CKsPin, CKsRequestor, CKsSplitter, CKsSplitterBranch, CKsPipeSection, KSPIN, KSFILTER, KSFILTERFACTORY, KSDEVICE, KSSTREAM_POINTER, KSPFRAME_HEADER, KSPFRAME_HEADER, KSPDO_EXTENSION, KSIDEVICE_HEADER KSIOBJECT_HEADER, KSSTREAM_HEADER, KSPIN_DESCRIPTOR_EX, CKsProxy, CKsInputPin, CKsOutputPin, CasyncItemHandler.

Для объектов *класса портов Тип* должен быть одним из следующих значений: DEVICE_CONTEXT, CPortWaveCyclic, CPortPinWaveCyclic, CPortTopology, CPortDMus, CIrpStream, CKsShellRequestor, CPortFilterWaveCyclic, CDmaChannel, CPortWavePci, CportPinWavePci.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

# Замечания

Как правило, для отображения структур данных можно использовать !ks.dump.

Однако если символы загружены неправильно или слишком много информации выгружаются, логика идентификации типов в **команде !ks.dump** может не определить тип структуры по указанному адресу.

В этом случае попробуйте использовать **команду !ks.forcedump** . Эта команда работает так же, как **!ks.dump** , за исключением того, что пользователь указывает тип объекта.

**Обратите внимание**, что команда !ks.forcedump **не проверяет правильностьтипа структуры, найденной по адресу, указанному в *объекте***. Команда предполагает, что это тип структуры, найденной в *объекте* , и отображает данные соответствующим образом.

Список всех поддерживаемых объектов можно получить путем выдачи **команды !ks.forcedump** без аргументов.

Ниже приведены два примера выходных данных из **!ks.forcedump**, используя адрес фильтра для *аргумента object* , но с различными уровнями детализации:

dbgcmd

```
kd> !forcedump 829493c4 KSFILTER
WARNING: I am dumping 829493c4 as a KSFILTER.
         No checking has been performed to ensure that it is this type!!!

Filter object 829493c4 [CKsFilter = 82949350]
    Descriptor     f7a233c8:
    Context        829dce28

kd> !forcedump 829493c4 KSFILTER 7
WARNING: I am dumping 829493c4 as a KSFILTER.
         No checking has been performed to ensure that it is this type!!!
```

```
Filter object 829493c4 [CKsFilter = 82949350]
    Descriptor     f7a233c8:
    Filter Category GUIDs:
        Video
        Capture
    Context        829dce28
    INTERNAL INFORMATION:
        Public Parent Factory   829782dc
        Aggregated Unknown      00000000
        Device Interface        823b83c8
        Control Mutex           829493f8 is not held
    Object Event List:
        None
    CKsFilter object 82949350 [KSFILTER = 829493c4]
        Processing Mutex         82949484 is not held
        Gate &                   82949464
        Gate.Count               1
        Pin Factories:
            Pin ID 0 [Video/General Capture Out]:
                Child Count        1
                Bound Child Count  1
                Necessary Count    0
                Specific Instances:
                    8293f580
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.graph

Статья • 05.04.2024

Команда **расширения !ks.graph отображает текстовое описание графа** режима ядра в топологическом порядке.

```
dbgcmd
```

```
!ks.graph Object [Level] [Flags]
```

## Параметры

*Объект*
Указывает указатель на объект, используемый в качестве отправной точки для графа. Должен быть указателем на один из следующих объектов: файловый объект, IRP, закрепление или фильтр.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7. Уровни для !ks.graph совпадают с уровнями **для !ks.dump.**

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов.

Bit 0 (0x1)
Отображение списка ip-адресов irps в очереди для каждого экземпляра пин-кода в графе.

Бит 1 (0x2)
Отображение списка поставщиков irps, ожидающих от каждого экземпляра пин-кода в графе. Отображаются только irps, которые пин-код знает, что он ожидает.

Бит 4 (0x10)
Анализ остановленного графа для подозрительных фильтров.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

# Замечания

Эта команда может занять некоторое время для обработки.

**Выполните команду !ks.graph** без аргументов для справки.

Ниже приведен пример **отображения !ks.graph** с адресом объекта фильтра:

dbgcmd

```
kd> !graph 829493c4
Attempting a graph build on 829493c4...  Please be patient...

Graph With Starting Point 829493c4:

"avssamp" Filter 82949350, Child Factories 1
    Output Factory 0 [Video/General Capture]:
        Pin 8293f4f0 (File 82503498) Irps(q/p) = 2, 0
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.libexts

Статья • 05.04.2024

Расширение **!ks.libexts** предоставляет доступ к расширениям библиотеки, предоставленным Корпорацией Майкрософт, которые статически связаны с модулем расширения.

dbgcmd

```
!ks.libexts [Command] [Libext]
```

# Параметры

*Command*
Необязательно. Указывает одно из следующих значений. Если этот аргумент опущен, **!ks.libexts** возвращает сведения справки.

**disableall**
Отключите все расширения библиотеки. Если это используется, опустите *параметр Libext* .

**disable**
Отключите определенное расширение библиотеки по имени. При использовании укажите имя в параметре *Libext* .

**enableall**
Включите все расширения библиотеки. Включены только загруженные компоненты с правильными символами. Если это используется, опустите *параметр Libext* .

**enable**
Включите определенное расширение библиотеки по имени. При использовании укажите имя в параметре *Libext* . Можно включить только загруженные компоненты с правильными символами.

**details**
Отображение сведений обо всех расширениях связанной библиотеки. Если это используется, опустите *параметр Libext* .

*Libext*
Указывает имя расширения библиотеки. Требуется только для значений команд *включения* или **отключения**.

# DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Модуль расширения содержит платформу расширяемости, которая позволяет создавать и связывать отдельные компоненты в Ks.dll. Эти дополнительные компоненты называются расширениями библиотеки.

Команда **!ks.libexts** позволяет просматривать статистику об этих расширениях библиотеки, а также контролировать их. Дополнительные сведения см. в статье **!ks.libexts** без аргументов.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.objhdr

Статья • 08.07.2024

Расширение **!ks.objhdr** отображает заголовок объекта потоковой передачи ядра, связанный с указанным объектом файла.

dbgcmd

```
!ks.objhdr FileObject [Level] [Flags]
```

## Параметры

*FileObject*
Этот параметр указывает указатель на объект WDM-файла. Если *FileObject* недопустим, команда возвращает ошибку.

*Уровень*
Необязательно. Значения совпадают с значениями для **!ks.dump**.

*Флаги*
Необязательно. Значения совпадают с значениями для **!ks.dump**.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Уровни и флаги для **!ks.objhdr** идентичны тем, которые описаны в **!ks.dump**.

Выходные данные из !ks.allstreams **и** **!ks.enumdevobj** **можно использовать в качестве входных данных для !ks.objhdr.** Для этого с примером *avssamp* , например, выполните следующие команды:

dbgcmd

```
kd> !ks.allstreams
6 Kernel Streaming FDOs found:
    Functional Device 8299be18 [\Driver\smwdm]
    Functional Device 827c86d8 [\Driver\wdmaud]
    Functional Device 827c0f08 [\Driver\sysaudio]
    Functional Device 82424590 [\Driver\avssamp]
    Functional Device 82423720 [\Driver\MSPQM]
    Functional Device 82b91a88 [\Driver\MSPCLOCK]
kd> !ks.enumdevobj 82424590
WDM device object 82424590:
    Corresponding KSDEVICE        82427540
    Factory 8285baa4 [Descriptor f7a333c8] instances:
        82837a34
kd> !ks.objhdr 82837a34 7
```

Результаты этой команды могут быть длительными. Чтобы остановить выходные данные, выполните ctrl-BREAK (WinDbg) или CTRL-C (NTSD, CDB, KD).

Ниже приведен отдельный пример:

dbgcmd

```
kd> !ks.objhdr 81D828B8 7
Adjusting file object 81D828B8 to object header 81BC1008

Object Header 81BC1008
    Associated Create Items:
        Create Item F9F77E98
            CreateFunction = ks!CKsFilter::DispatchCreatePin+0x00
            ObjectClass = {146F1A80-4791-11D0-A5D6-28DB04C10000}
            Flags = 0
    Child Create Handler List:
        Create Item F9F85AA0
            CreateFunction = ks!CKsPin::DispatchCreateAllocator+0x00
            ObjectClass = {642F5D00-4791-11D0-A5D6-28DB04C10000}
            Flags = 0
        Create Item F9F85AB8
            CreateFunction = ks!CKsPin::DispatchCreateClock+0x00
            ObjectClass = {53172480-4791-11D0-A5D6-28DB04C10000}
            Flags = 0
        Create Item F9F85AD0
            CreateFunction = ks!CKsPin::DispatchCreateNode+0x00
            ObjectClass = {0621061A-EE75-11D0-B915-00A0C9223196}
            Flags = 0
    DispatchTable:
```

```
        Dispatch Table F9F85AE8
            DeviceIoControl = ks!CKsPin::DispatchDeviceIoControl+0x00
            Read = ks!KsDispatchInvalidDeviceRequest+0x00
            Write = ks!KsDispatchInvalidDeviceRequest+0x00
            Flush = ks!KsDispatchInvalidDeviceRequest+0x00
            Close = ks!CKsPin::DispatchClose+0x00
            QuerySecurity = ks!KsDispatchQuerySecurity+0x00
            SetSecurity = ks!KsDispatchSetSecurity+0x00
            FastDeviceIoControl =
ks!KsDispatchFastIoDeviceControlFailure+0x00
            FastRead = ks!KsDispatchFastReadFailure+0x00
            FastWrite = ks!KsDispatchFastReadFailure+0x00
    TargetState: KSTARGET_STATE_ENABLED
    TargetDevice: 00000000
    BaseDevice  : 81BBDF10
    Stack Depth : 1
    Corresponding AVStream object = 81A971B0
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.ohdr

Статья • 05.04.2024

Расширение **!ks.ohdr** отображает сведения о заголовке объекта потоковой передачи ядра.

dbgcmd

```
!ks.ohdr Object [Level] [Flags]
```

## Параметры

*Объект*
Этот параметр задает указатель на заголовок объекта KS. Если *объект* недопустим, команда возвращает ошибку.

*Уровень*
Необязательно. Значения совпадают с значениями для **!ks.dump**.

*Flags*
Необязательно. Значения совпадают с значениями для **!ks.dump**.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Команда **!ks.ohdr работает аналогично команде !ks.objhdr**, в которой отображаются сведения о заголовке объекта KS. Разница заключается в том, что

вызывающий объект предоставляет прямой адрес заголовка объекта KS вместо адреса связанного объекта файла.

Уровни и флаги для **!ks.ohdr** идентичны тем, которые описаны в **!ks.dump**.

Если запрашиваемые данные не отображаются, рассмотрите возможность использования !ks.dump **вместо !ks.ohdr.**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.pciaudio

Статья • 05.04.2024

Расширение **!ks.pciaudio** отображает список FDOS, подключенных к PortCls.

dbgcmd

```
!ks.pciaudio [Options] [Level]
```

## Параметры

*Необязательные параметры* . Указывает тип отображаемой информации. *Параметры* могут быть любым сочетанием следующих битов.

Bit 0 (0x1)
Отображение списка выполняемых потоков.

Бит 1 (0x2)
Отображение списка всех потоков.

Бит 3 (0x4)
Выходные отображаемые потоки. *Уровень* имеет значение только в том случае, если этот бит задан.

*Уровень*
Необязательный и применимый, только если в параметрах *задан* бит 3. Уровни совпадают с уровнями для **!ks.dump**.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

# Замечания

Ниже приведен пример выходных данных из **!ks.pciaudio**:

dbgcmd

```
kd> !ks.pciaudio
1 Audio FDOs found:
 Functional Device 8299be18 [\Driver\smwdm]
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.pciks

Статья • 08.07.2024

Расширение **!ks.pciks** содержит функциональные устройства для устройств потоковой передачи ядра, подключенных к шине PCI. При необходимости он может отображать сведения о активных потоках на этих функциональных устройствах.

dbgcmd

```
!ks.pciks [Flags] [Level]
```

## Параметры

*Флаги*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов.

Bit 0 (0x1)
Вывод списка всех запущенных потоков.

Бит 1 (0x2)
Рекурсивные графы для поиска устройств, отличных от PCI.

Бит 2 (0x4)
Отображение списка экземпляров прокси-сервера.

Бит 3 (0x8)
Отображение в настоящее время в очереди irps.

Бит 4 (0x10)
Отображение сведений обо всех потоках.

Бит 5 (0x20)
Отображение форматов активных потоков.

*Уровень*
Необязательный и применимый только к сочетаниям флагов, которые приводят к отображению данных. Уровни совпадают с уровнями для **!ks.dump**.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

# Замечания

Эта команда может занять время выполнения, особенно если драйвер фильтра ACPI загружен, или если средство проверки драйверов включено, а имена драйверов выгружаются.

Ниже приведен пример **отображения !ks.pciks** :

dbgcmd

```
kd> !pciks
1 Kernel Streaming FDOs found:
    Functional Device 82a17690 [\Driver\smwdm]
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !ks.topology

Статья • 08.07.2024

Расширение **!ks.topology отображает отсортированный граф внутренней топологии** фильтра, ближайшего к объекту.

dbgcmd

```
!ks.topology Object [Level] [Flags]
```

## Параметры

*Объект*
Указывает указатель на объект, используемый в качестве основы для графа. Может быть указателем на объект файла, IRP, закрепить, фильтр или другой объект KS.

*Уровень*
Необязательно. Задает уровень детализации для отображения на шкале 0-7 с постепенно большей информацией, отображаемой для более высоких значений. Чтобы отобразить все доступные сведения, укажите значение 7.

*Флаги*
В настоящее время недоступно.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | winxp\Ks.dll |
| **Windows XP и более поздние версии** | Ks.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра".

## Замечания

Чтобы помочь, выполните **команду !ks.topology** без аргументов.

Обратите внимание, что эта команда может занять несколько минут.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !acxkd

21.06.2024

Расширение **!acxkd** отображает сведения о драйверах расширения аудиокласса (ACX). Дополнительные сведения о расширениях класса ACX см. в обзоре расширений аудиоклассов ACX.

# Синтаксис

dbgcmd

```
!acxkd.[Options]
```

## DLL-библиотеки

Acxkd.dll

## Версия отладчика

Расширение !acxkd доступно в WinDbg версии 1.2402.24001.0 и более поздних версиях.

## Отладка ACX 1.0

Расширение отладчика !acxkd предлагает только частичные функции в ACX 1.0. Рекомендуется обновить до ACX 1.1.

# Параметры

*Параметры* . Указывает тип отображаемой информации.

Развернуть таблицу

| Вариант | Описание | Параметр |
|---|---|---|
| !Справка | Отображает сведения о доступных командах расширения. | `[<command name>]` |
| !acxcircuit | Дамп объекта ACXCIRCUIT. | `<circuit>` — дескриптор WDF ACXCIRCUIT |

| Вариант | Описание | Параметр |
|---|---|---|
| !acxdataformat | Дамп объекта ACXDATAFORMAT. | `<dataformat>` — дескриптор WDF ACXDATAFORMAT |
| !acxdataformatlist | Дамп объекта ACXDATAFORMATLIST. | `<dataformatlist>` — дескриптор WDF ACXDATAFORMATLIST |
| !acxdevice | Дампа объекта ACXDEVICE. | `<device>` — дескриптор WDF ACXDEVICE |
| !acxelement | Дамп объекта ACXELEMENT. | `<element>` — дескриптор WDF ACXELEMENT |
| !acxevents | Дамп событий объекта ACXOBJECT. | `<events>` — дескриптор WDF ACXOBJECT |
| !acxfactory | Дамп объекта ACXFACTORYCIRCUT. | `<factory>` — дескриптор WDF ACXFACTORYCIRCUIT |
| !acxmanager | Дамп объекта ACXMANAGER. | нет |
| !acxmethods | Методы дампа объекта ACXOBJECT. | `<object>` — дескриптор WDF ACXOBJECT |
| !acxobjbag | Дампа объекта ACXOBJECTBAG. | `<objbag>` — дескриптор WDF ACXOBJECTBAG |
| !acxobject | Дамп объекта ACXOBJECT. | `<object>` — дескриптор WDF ACXOBJECT |
| !acxpin | Дампа объекта ACXPIN. | `<pin>` — дескриптор WDF ACXPIN |
| !acxproperties | Свойства дампа объекта ACXOBJECT. | `<properties>` — дескриптор WDF ACXOBJECT |
| !acxstream | Дамп объекта ACXSTREAM. | `<stream>` — дескриптор ACXSTREAM |
| !acxstreambridge | Дамп объекта ACXSTREAMBRIDGE. | `<bridge>` — дескриптор ACXSTREAMBRIDGE |
| !acxtarget | Дампа объекта ACXTARGET. | `<target>` — дескриптор WDF ACXTARGET |
| !acxtemplate | Дампа объекта ACXTEMPLATE. | `<tmpt>` — дескриптор ACXTEMPLATE |

# Замечания

## !acxkd.help

Чтобы получить список всех доступных команд, используйте команду acxkd `!help` .

dbgcmd

```
0: kd> !acxkd.help
Commands for C:\Debugger\acxkd.dll:
  !acxcircuit        - Dump a ACXCIRCUT object.
  !acxdataformat     - Dump a ACXDATAFORMAT object.
  !acxdataformatlist - Dump a ACXDATAFORMATLIST object.
  !acxdevice         - Dump a ACXDEVICE object.
  !acxelement        - Dump a ACXELEMENT object.
  !acxevents         - Dump events of a ACXOBJECT object.
  !acxfactory        - Dump a ACXFACTORYCIRCUT object.
  !acxmanager        - Dump a ACXMANAGER object.
  !acxmethods        - Dump methods of a ACXOBJECT object.
  !acxobjbag         - Dump a ACXOBJECTBAG object.
  !acxobject         - Dump a ACXOBJECT* object.
  !acxpin            - Dump a ACXPIN object.
  !acxproperties     - Dump properties of a ACXOBJECT object.
  !acxstream         - Dump a ACXSTREAM object.
  !acxstreambridge   - Dump a ACXSTREAMBRIDGE object.
  !acxtarget         - Dump a ACXTARGET* object.
  !acxtemplate       - Dump a ACX*TEMPLATE object.
  !help              - Displays information on available extension commands
!help <cmd> will give more information for a particular command
```

Используйте команду acxkd `!help` , чтобы узнать больше о любой из команд, например о команде `!acxdevice` .

dbgcmd

```
0: kd> !acxkd.help acxdevice
!acxdevice <device>
  <device> - ACXDEVICE handle
Dump a ACXDEVICE object.
```

Используйте команду !acxdevice в качестве отправной точки для проверки драйвера ACX.

dbgcmd

```
3: kd> !acxdevice 0x00007dfadb0a5358
Dumping info for ACXDEVICE 0x00007dfadb0a5358

    In connected standby: FALSE

    State: AfxDeviceStateInitialized
    State history:
        0 : AfxDeviceStateInvalid
        1 : AfxDeviceStateInvalid
        2 : AfxDeviceStateInvalid
```

```
        3 : AfxDeviceStateInvalid
        4 : AfxDeviceStateInvalid
        5 : AfxDeviceStateCreated
        6 : AfxDeviceStateInitializing
        7 : AfxDeviceStateInitialized

    Create dispatch list:
        Create name: eHDMIOutTopo
        Dispatch routine: fffff80393179918
        Dispatch context: ffff82052513b960

    Circuits:
        ---------------------------------

        [Circuit 0]

        Name: eHDMIOutTopo
        Type: AcxCircuitTypeRender
        ComponentId: {BFCA9AD9-4EED-46C2-9323-B5D4400761A5}

        State: AfxCircuitStatePoweredUp

        Interface is enabled
        SymolicLinkName: \??
\HDAUDIO#SUBFUNC_01&VEN_8086&DEV_281F&NID_0001&SUBSYS_00000000&REV_1000#6&4948348&
0&0002&00000025#{2c6bb644-e1ae-47f8-9a2b-1d1fa750f2fa}\eHDMIOutTopo

        !acxproperties 00007dfad9ccccb8
        !acxmethods 00007dfad9ccccb8
        !acxevents 00007dfad9ccccb8

        # Pins: 2
            !acxpin 00007dfadf996dd8
            !acxpin 00007dfad4697238

        # Elements: 1
            !acxelement 00007dfadf997a18

        # Streams: 0

        !acxcircuit 00007dfad9ccccb8

    !wdfqueue 00007dfade9beaf8
    !wdfdevice 00007dfadb0a5358

    !wdfhandle 00007dfadb0a5358
    dt Acx01000!Acx::AfxDevice ffff8205256ab420
```

Щелкните ссылки в выходных данных, чтобы отобразить информацию с помощью команд !acxproperties, !acxmethods и !acxevents.

Сведения о поиске wdfhandle для объекта ACXDEVICE см . в пошаговом руководстве по примеру драйвера ACX в этой статье.

## Команды WDF — !wdfkd.wdfldr

Так как драйверы ACX являются драйверами WDF, используйте любую из команд отладчика ядра WDF. Например, используйте !wdfkd.wdfldr для отображения сведений о версии и привязке ACX с WDF.

dbgcmd

```
0: kd> !wdfldr acx01000
WDF Driver: Acx01000
---------------------------------
CLIENT_MODULE   0xffff82052abdcbc0
  WDF Version   v1.31
  ImageName     Acx01000.sys
  ImageAddress  0xfffff80393150000
  ImageSize     0xb3000
  BindingList   0xffff82052abdcc08

  ImageName        WdfVer Ver   WdfGlobals         BindInfo           ImageAddress
ImageSize
  Wdf01000.sys     v1.33  v1.33 0xffff8205218d8fb0 0xffff8205218d8df0
0xfffff80356a00000 0x000c7000
---------------------------------
CLASS_MODULE    0xffff820525b3dc90
  WDF Version   v1.31
  Version       v1.1
  Service       \Registry\Machine\System\CurrentControlSet\Services\acx01000
  ImageName     Acx01000.sys
  ImageAddress  0xfffff80393150000
  ImageSize     0xb3000
  ClientsList   0xffff820525b3dcf8
  Associated Clients: 1

  ImageName        WdfVer Ver   WdfGlobals         BindInfo           ImageAddress
ImageSize
  AcxHdAudio.sys   v1.25  v1.0  0xffff820527f70ae0 0xfffff803930df3e8
0xfffff803930c0000 0x0008a000
---------------------------------
```

## !acxkd.acxmanager

Используйте команду для отображения сведений об объекте `!acxmanager` ACXMANAGER. Это обеспечивает хорошую отправную точку для изучения драйверов ACX.

В этом примере показана первая часть обширных `!acxmanager` выходных данных, предоставляемых для конфигурации ACX mulitcircuit.

dbgcmd

```
10: kd> !acxmanager
Dumping info for ACXMANAGER 0x000054f94d1d4378

    Delete pending: No

    # singleton composites: 8
        ---------------------------------

        [Composite 0]

        State: AfxCompositeStateActive
        !acxobjbag 000054f94c8e61c8
        !acxtemplate 000054f94c014d28
        !acxobject 000054f94c0141c8

        # circuits: 3
            ---------------------------------
            [Circuit 0 CORE]
…
```

В этом примере выходных данных показан один канал ACX.

dbgcmd

```
0: kd> !acxmanager
Dumping info for ACXMANAGER 0x000049f6c3c769f8

    Delete pending: No

    # singleton composites: 0
    # Composite factories: 0
    !wdfhandle 000049f6c3c769f8
    dt Acx01000!Acx::AfxManager ffffb6093c3896b0
```

## !acxkd.acxobject

В выходных данных для !acxmanager указан адрес для wdfhandle. Используйте адрес wdfhandle с командой `!acxobject` для отображения сведений об ACXMANAGER или любом другом объекте ACX.

dbgcmd

```
0: kd> !acxobject 000049f6c3c769f8
Dumping info for ACXMANAGER 0x000049f6c3c769f8

    Delete pending: No

    # singleton composites: 0
    # Composite factories: 0
```

```
    !wdfhandle 000049f6c3c769f8
    dt Acx01000!Acx::AfxManager ffffb6093c3896b0
```

Щелкните ссылку `dt` в выходных данных, показанных выше, чтобы просмотреть дополнительные сведения о внутренних структурах объектов ACX.

dbgcmd

```
0: kd> dt Acx01000!Acx::AfxManager ffffb6093c3896b0
   +0x000 m_Object         : 0x000049f6`c3c769f8 ACXMANAGER__
   =fffff803`8a478ad8 s_AfxManager     : 0xffffb609`3c3896b0 Acx::AfxManager
   +0x008 m_AcxGlobals     : 0xffffb609`403cae24 _ACX_DRIVER_GLOBALS
   +0x010 m_Flags          : 0
   +0x010 m_Unloading      : 0y0
   +0x018 m_CompositesMutex : _FAST_MUTEX
   +0x050 m_CompositeFactoriesList : _LIST_ENTRY [ 0xffffb609`3c389700 -
0xffffb609`3c389700 ]
   +0x060 m_CompositeFactoriesCount : 0n0
   +0x068 m_CompositesList : _LIST_ENTRY [ 0xffffb609`3c389718 -
0xffffb609`3c389718 ]
   +0x078 m_CompositesCount : 0n0
   +0x080 m_FactoriesList  : _LIST_ENTRY [ 0xffffb609`3c389730 -
0xffffb609`3c389730 ]
   +0x090 m_CircuitsList   : _LIST_ENTRY [ 0xffffb609`40a27068 -
0xffffb609`40a27068 ]
   +0x0a0 m_FiltersMutex   : _FAST_MUTEX
   +0x0d8 m_FactoryFiltersList : _LIST_ENTRY [ 0xffffb609`3c389788 -
0xffffb609`3c389788 ]
   +0x0e8 m_CircuitFiltersList : _LIST_ENTRY [ 0xffffb609`3c389798 -
0xffffb609`3c389798 ]
   +0x0f8 m_FactoriesNotificationHandle : 0xfffffc70c`b2356790 Void
   +0x100 m_CircuitsNotificationHandle : 0xfffffc70c`b2357ec0 Void
   +0x108 m_CommandQueue   : 0xffffb609`3ffc5400 Acx::AfxWorkQueue
```

Обратите внимание, что выходные данные отладчика могут содержать сочетание общедоступных объектов ACX, таких как ACXMANAGER, ACXCIRCFACTORY и ACXCIRCUIT и внутренние структуры, определенные для непрозрачности. Внутренние типы не гарантируют, что они остаются неизменными или доступны в разных выпусках ACX и не должны вызываться или использоваться напрямую.

Так как объекты ACX являются объектами WDF, можно использовать команду !wdfkd.wdfhandle для отображения дополнительных сведений об объекте ACXMANAGER.

dbgcmd

```
0: kd> !wdfhandle 000049f6c3c769f8
Treating handle as a KMDF handle!

Dumping WDFHANDLE 0x000049f6c3c769f8
```

```
============================
Handle type is WDFOBJECT [ACXMANAGER]
Refcount: 2
Contexts:
    context:  dt 0xffffb6093c3896b0 Acx01000!AfxManager (size is 0x110 bytes)
    EvtCleanupCallback fffff8038a453070
Acx01000!Acx::WdfCpp::ObjectContext<ACXMANAGER__
*,Acx::AfxManager>::EvtObjectContextCleanupThunk
    EvtDestroyCallback fffff8038a453000
Acx01000!Acx::WdfCpp::ObjectContext<ACXMANAGER__
*,Acx::AfxManager>::EvtObjectContextDestroyThunk

    context:  dt 0xffffb609404c2280 Acx01000!WdfCustomType_ACXMANAGER (size is
0x10 bytes)
    <no associated attribute callbacks>

Parent: !wdfhandle 0x000049f6d4e2d5c8, type is WDFDEVICE
Owning device: !wdfdevice 0x000049f6d4e2d5c8

!wdfobject 0xffffb6093c389600
```

## !acxkd.acxpin

Команды, отображающие другие сведения ACX, например `!acxpin` требовать дескриптор WDF для объекта. Сведения о поиске дескриптора WDF для объекта ACX см . в пошаговом руководстве по примеру драйвера ACX в этой статье.

dbgcmd

```
0: kd> !acxpin 0x000049f6befeee38
Dumping info for ACXPIN 0x000049f6befeee38

    ID: 0
    Type: AcxPinTypeSink
    Type: AcxPinCommunicationSink

    Category: KSCATEGORY_AUDIO
    Name: {00000000-0000-0000-0000-000000000000}
```

В зависимости от состояния объекта ACX не все сведения могут быть доступны для отображения.

## !acxkd.acxdataformatlist

Как и в !acxpin, `!acxdataformatlist` отображаются сведения о списках данных ACX.

dbgcmd

```
0: kd> !acxdataformatlist 0x000049f6bf8be668
Dumping info for ACXDATAFORMATLIST 0x000049f6bf8be668

    # Scan count: 0
    # Data formats: 6
    Data formats:
        Sample Rate: 48000, #Channels: 2, #Bits: 16, ValidBits: 16, Mask: 0x3
(default)
        Sample Rate: 48000, #Channels: 2, #Bits: 32, ValidBits: 24, Mask: 0x3
        Sample Rate: 44100, #Channels: 2, #Bits: 16, ValidBits: 16, Mask: 0x3
        Sample Rate: 44100, #Channels: 2, #Bits: 32, ValidBits: 24, Mask: 0x3
        Sample Rate: 32000, #Channels: 2, #Bits: 16, ValidBits: 16, Mask: 0x3
        Sample Rate: 32000, #Channels: 2, #Bits: 32, ValidBits: 24, Mask: 0x3

    !wdfhandle 000049f6bf8be668
    dt Acx01000!Acx::AfxDataFormatList ffffb60940444530
```

# Пошаговое руководство по драйверу ACX

В этом разделе представлено пошаговое руководство по отладке драйвера ACX.

## Путь к символам

Используйте команды Symfix и .sympath (Set Symbol Path) для изменения пути символа. Если вы используете локальный код с драйвером, добавьте путь к этому коду. Используйте команду reload (Reload Module) для перезагрузки символов из текущего пути.

dbgcmd

```
.symfix
.sympath+ C:\Windows-driver-samples-develop\audio\Acx\Samples\AudioCodec\Driver
.reload /f
```

## Контекст отладчика для драйверов

Если вы выполняете отладку активного драйвера ACX, установите точку останова. Это помещает отладчик в контекст объектов ACX, чтобы иметь возможность собирать и отображать сведения.

Эти примеры точек останова предназначены для запуска примера драйвера AudioCodec.

dbgcmd

```
bm AudioCodec!DriverEntry
bm AudioCodec!AcxDriverInitialize
```

Эти примеры точек останова предназначены для запуска при выполнении определенных действий, таких как создание пин-кода или канала.

dbgcmd

```
bm AudioCodec!AcxPinCreate
bm AudioCodec!AcxCircuitCreate
bm AudioCodec!Codec_EvtBusDeviceAdd
```

После загрузки драйвера и запуска соответствующей точки останова и допустимого контекста выполнения используйте команды !acxkd для отображения сведений о любом объекте ACX. Используйте команду !acxobject для общих сведений и определенной команды, например !acxcircuit или !acxpin для более детализированной информации.

## Загрузка библиотеки acxkd dll

Используйте команду load (DLL расширения загрузки) для загрузки расширения acxkd.dll.

dbgcmd

```
.load acxkd.dll
```

## Отображение сведений о драйвере ACX

Чтобы собрать сведения о целевом драйвере, используйте команду lm (список загруженных модулей), чтобы просмотреть все загруженные драйверы. Затем используйте параметры Dvm для отображения сведений о интересующем драйвере ACX, как показано здесь.

dbgcmd

```
0: kd> lm Dvm AcxHdAudio
Browse full module list
start             end                 module name
fffff803`8a3c0000 fffff803`8a448000   AcxHdAudio   (private pdb symbols)
C:\ProgramData\Dbg\sym\AcxHdAudio.pdb\6AEA2622909B20C1AD149C57ACBB4A6F1\AcxHdAudio
.pdb
    Loaded symbol image file: AcxHdAudio.sys
    Mapped memory image file:
C:\ProgramData\Dbg\sym\AcxHdAudio.sys\0829423388000\AcxHdAudio.sys
    Image path: \SystemRoot\System32\drivers\AcxHdAudio.sys
```

```
    Image name: AcxHdAudio.sys
    Browse all global symbols  functions  data  Symbol Reload
    Image was built with /Brepro flag.
    Timestamp:        08294233 (This is a reproducible build file hash, not a
timestamp)
    CheckSum:         00087DD6
    ImageSize:        00088000
    Translations:     0000.04b0 0000.04e4 0409.04b0 0409.04e4
    Information from resource tables:
```

Используйте команду x (Проверка символов) и маску подстановочного знака для отображения определенных структур ACX, таких как ACXPIN.

dbgcmd

```
0: kd> x /D AcxHdAudio!acxpin*
 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

fffff803`8a3e3216 AcxHdAudio!AcxPinGetRawDataFormatList =  (inline caller)
AcxHdAudio!HDACodec_EvtFormatChange+66
fffff803`8a3e31e0 AcxHdAudio!AcxPinGetCircuit =  (inline caller)
AcxHdAudio!HDACodec_EvtFormatChange+30
fffff803`8a3e361f AcxHdAudio!AcxPinNotifyDataFormatChange =  (inline caller)
AcxHdAudio!HDACodec_EvtFormatChange+46f
fffff803`8a3e7396 AcxHdAudio!AcxPinCreate =  (inline caller)
AcxHdAudio!HDACodecR_CreateRenderCircuit+962
fffff803`8a3e74a2 AcxHdAudio!AcxPinGetRawDataFormatList =  (inline caller)
AcxHdAudio!HDACodecR_CreateRenderCircuit+a6e
fffff803`8a3e75f8 AcxHdAudio!AcxPinCreate =  (inline caller)
AcxHdAudio!HDACodecR_CreateRenderCircuit+bc4
fffff803`8a3e789a AcxHdAudio!AcxPinAddJacks =  (inline caller)
AcxHdAudio!HDACodecR_CreateRenderCircuit+e66
fffff803`8a3e5ae5 AcxHdAudio!AcxPinGetCircuit =  (inline caller)
AcxHdAudio!HDACodecR_EvtAcxPinRetrieveJackSinkInfo+25
fffff803`8a3e5913 AcxHdAudio!AcxPinGetCircuit =  (inline caller)
AcxHdAudio!HDACodecR_EvtAcxPinRetrieveName+73
fffff803`8a3ea277 AcxHdAudio!AcxPinCreate =  (inline caller)
AcxHdAudio!HDACodecC_CreateCaptureCircuit+b27
fffff803`8a3ea39c AcxHdAudio!AcxPinGetRawDataFormatList =  (inline caller)
AcxHdAudio!HDACodecC_CreateCaptureCircuit+c4c
fffff803`8a3ea4bc AcxHdAudio!AcxPinCreate =  (inline caller)
AcxHdAudio!HDACodecC_CreateCaptureCircuit+d6c
fffff803`8a3ea7b2 AcxHdAudio!AcxPinAddJacks =  (inline caller)
AcxHdAudio!HDACodecC_CreateCaptureCircuit+1062
```

В зависимости от текущего контекста выполнения в отладчике можно использовать dx (выражение объектной модели отладчика) для детализации конкретных структур ACX.

dbgcmd

```
0: kd> dx -r2 AudioCodec!AcxDeviceAddCircuit
AudioCodec!AcxDeviceAddCircuit                 :
AudioCodec!AcxDeviceAddCircuit+0x0 [Type: long __cdecl(WDFDEVICE__ *,ACXCIRCUIT__
*)]
0: kd> u fffff8007ead1120
AudioCodec!AcxDeviceAddCircuit [C:\Program Files (x86)\Windows
Kits\10\Include\10.0.26016.0\km\acx\km\1.1\AcxDevice.h @ 206]:
fffff800`7ead1120 4889542410      mov     qword ptr [rsp+10h],rdx
fffff800`7ead1125 48894c2408      mov     qword ptr [rsp+8],rcx
fffff800`7ead112a 4883ec38        sub     rsp,38h
fffff800`7ead112e b808000000      mov     eax,8
fffff800`7ead1133 486bc046        imul    rax,rax,46h
fffff800`7ead1137 488d0dc2ab0000  lea     rcx,[AudioCodec!AcxFunctions
(fffff800`7eadbd00)]
fffff800`7ead113e 488b0401        mov     rax,qword ptr [rcx+rax]
fffff800`7ead1142 4889442420      mov     qword ptr [rsp+20h],rax
```

## !wdfkd.wdfdriverinfo

Используйте команду !wdfdriverinfo с именем драйвера для сбора сведений WDF, таких как связанные объекты ACX и wdfhandle ACXDEVICE.

dbgcmd

```
0: kd> !wdfdriverinfo AcxHdAudio.sys 1 -v
---------------------------------
Default driver image name: AcxHdAudio
WDF library image name: Wdf01000
 FxDriverGlobals  0xffffb609403e3de0
 WdfBindInfo      0xfffff8038a3dd1b0
   Version        v1.25
 Library module   0xffffb60929cc6050
   ServiceName    \Registry\Machine\System\CurrentControlSet\Services\Wdf01000
   ImageName      Wdf01000
---------------------------------
WDFDRIVER: 0x000049f6bfe9d5d8
    context:  dt 0xffffb60940162bc0 AcxHdAudio!CODEC_DRIVER_CONTEXT (size is 0x1
bytes)
    <no associated attribute callbacks>

    context:  dt 0xffffb6093ccead20 Acx01000!AfxDriver (size is 0x20 bytes)
    EvtCleanupCallback fffff8038a455780
Acx01000!Acx::WdfCpp::ObjectContext<WDFDRIVER__
*,Acx::AfxDriver>::EvtObjectContextCleanupThunk
    EvtDestroyCallback fffff8038a455740
Acx01000!Acx::WdfCpp::ObjectContext<WDFDRIVER__
*,Acx::AfxDriver>::EvtObjectContextDestroyThunk
Object Hierarchy: !wdfhandle 0x000049f6bfe9d5d8 0xff
Driver logs: !wdflogdump AcxHdAudio.sys -d
Framework logs: !wdflogdump AcxHdAudio.sys -f
```

```
    !wdfdevice 0x000049f6bffba488 ff (FDO)
        Pnp/Power State: WdfDevStatePnpStarted, WdfDevStatePowerD0,
WdfDevStatePwrPolStartedWakeCapable
        context:  dt 0xffffb60940045e60 AcxHdAudio!CODEC_DEVICE_CONTEXT (size is
0x110 bytes)
        EvtCleanupCallback fffff8038a3e3c90
AcxHdAudio!HDACodec_EvtDeviceContextCleanup

        context:  dt 0xffffb60933f030f0 Acx01000!AfxDevice (size is 0x150 bytes)
        EvtCleanupCallback fffff8038a451910
Acx01000!Acx::WdfCpp::ObjectContext<WDFDEVICE__
*,Acx::AfxDevice>::EvtObjectContextCleanupThunk
        EvtDestroyCallback fffff8038a451740
Acx01000!Acx::WdfCpp::ObjectContext<WDFDEVICE__
*,Acx::AfxDevice>::EvtObjectContextDestroyThunk

        context:  dt 0xffffb60940a26b90 AcxHdAudio!CODEC_RENDER_DEVICE_CONTEXT
(size is 0x38 bytes)
        <no associated attribute callbacks>

        context:  dt 0xffffb609409f2e00 AcxHdAudio!CODEC_CAPTURE_DEVICE_CONTEXT
(size is 0x8 bytes)
        <no associated attribute callbacks>
        !wdfdevicequeues 0x000049f6bffba488

        !wdfdevice 0x000049f6bef1a848 ff (PDO)
            Pnp/Power State: WdfDevStatePnpStarted,
WdfDevStatePowerD0BusWakeOwner, WdfDevStatePwrPolStartedWakeCapable
            context:  dt 0xffffb609410e5aa0 AcxHdAudio!CODEC_RENDER_DEVICE_CONTEXT
(size is 0x38 bytes)
            EvtCleanupCallback fffff8038a3e6400
AcxHdAudio!HDACodecR_EvtDeviceContextCleanup

            context:  dt 0xffffb60933f090d0 Acx01000!AfxDevice (size is 0x150
bytes)
            EvtCleanupCallback fffff8038a451910
Acx01000!Acx::WdfCpp::ObjectContext<WDFDEVICE__
*,Acx::AfxDevice>::EvtObjectContextCleanupThunk
            EvtDestroyCallback fffff8038a451740
Acx01000!Acx::WdfCpp::ObjectContext<WDFDEVICE__
*,Acx::AfxDevice>::EvtObjectContextDestroyThunk
            !wdfdevicequeues 0x000049f6bef1a848

---------------------------------

WDF Verifier settings for AcxHdAudio.sys is OFF
---------------------------------
```

В выходных данных, показанных выше, ссылка доступна связанному wdfdevice. Щелкните ссылку, чтобы отобразить сведения о связанных объектах устройств WDF.

```
dbgcmd
```

```
0: kd> !wdfdevice 0x000049f6bef1a848 ff
Treating handle as a KMDF handle!

Dumping WDFDEVICE 0x000049f6bef1a848
================================

WDM PDEVICE_OBJECTs:  self ffffb60940ddbdd0

Pnp state:  119 ( WdfDevStatePnpStarted )
Power state:  309 ( WdfDevStatePowerD0BusWakeOwner )
Power Pol state:  531 ( WdfDevStatePwrPolStartedWakeCapable )
```

## !acxkd.acxdevice

Использование того же wdfhandle с `!acxdevice` информацией, ориентированной на ACX.

dbgcmd

```
3: kd> !acxdevice 0x00007dfadb0a5358
Dumping info for ACXDEVICE 0x00007dfadb0a5358

    In connected standby: FALSE

    State: AfxDeviceStateInitialized
    State history:
        0 : AfxDeviceStateInvalid
        1 : AfxDeviceStateInvalid
        2 : AfxDeviceStateInvalid
        3 : AfxDeviceStateInvalid
        4 : AfxDeviceStateInvalid
        5 : AfxDeviceStateCreated
        6 : AfxDeviceStateInitializing
        7 : AfxDeviceStateInitialized

    Create dispatch list:
        Create name: eHDMIOutTopo
        Dispatch routine: fffff80393179918
        Dispatch context: ffff82052513b960

    Circuits:
        ---------------------------------

        [Circuit 0]

        Name: eHDMIOutTopo
        Type: AcxCircuitTypeRender
        ComponentId: {BFCA9AD9-4EED-46C2-9323-B5D4400761A5}

        State: AfxCircuitStatePoweredUp

        Interface is enabled
```

```
        SymolicLinkName: \??
\HDAUDIO#SUBFUNC_01&VEN_8086&DEV_281F&NID_0001&SUBSYS_00000000&REV_1000#6&4948348&
0&0002&00000025#{2c6bb644-e1ae-47f8-9a2b-1d1fa750f2fa}\eHDMIOutTopo

        !acxproperties 00007dfad9ccccb8
        !acxmethods 00007dfad9ccccb8
        !acxevents 00007dfad9ccccb8

        # Pins: 2
            !acxpin 00007dfadf996dd8
            !acxpin 00007dfad4697238

        # Elements: 1
            !acxelement 00007dfadf997a18

        # Streams: 0

        !acxcircuit 00007dfad9ccccb8

    !wdfqueue 00007dfade9beaf8
    !wdfdevice 00007dfadb0a5358

    !wdfhandle 00007dfadb0a5358
    dt Acx01000!Acx::AfxDevice ffff8205256ab420
```

Чтобы отобразить сведения о других объектах ACX, таких как ACXCIRCUIT, используйте ссылку в выходных данных выше, которая вызывается `!acxcircuit` с соответствующим wdfhandle.

dbgcmd

```
3: kd> !acxcircuit 00007dfad9ccccb8
Dumping info for ACXCIRCUIT 0x00007dfad9ccccb8

    Name: eHDMIOutTopo
    Type: AcxCircuitTypeRender
    ComponentId: {BFCA9AD9-4EED-46C2-9323-B5D4400761A5}

    State: AfxCircuitStatePoweredUp
    State history:
        0 : AfxCircuitStateInvalid
        1 : AfxCircuitStateInvalid
        2 : AfxCircuitStateInvalid
        3 : AfxCircuitStateCreated
        4 : AfxCircuitStateInitializing
        5 : AfxCircuitStateInitialized
        6 : AfxCircuitStatePoweredDown
        7 : AfxCircuitStatePoweredUp

    Interface is enabled
    SymolicLinkName: \??
\HDAUDIO#SUBFUNC_01&VEN_8086&DEV_281F&NID_0001&SUBSYS_00000000&REV_1000#6&4948348&
```

```
0&0002&00000025#{2c6bb644-e1ae-47f8-9a2b-1d1fa750f2fa}\eHDMIOutTopo

    # Power references: 0
    # Open handles: 18

    !acxproperties 00007dfad9ccccb8
    !acxmethods 00007dfad9ccccb8
    !acxevents 00007dfad9ccccb8

    # Pins: 2
        !acxpin 00007dfadf996dd8
        !acxpin 00007dfad4697238

    # Elements: 1
        !acxelement 00007dfadf997a18
```

## !acxkd.acxproperties

В выходных ссылках команды с wdfhandle предоставляются для других объектов, таких как свойства ACX, которые можно отобразить.

dbgcmd

```
0: kd> !acxproperties 000049f6bf436b88
Dumping properties info for ACXOBJECT 0x000049f6bf436b88

    # sets: 4

        Set: {8C134960-51AD-11CF-878A-94F801C10000}
            Id: 0, Flags: 0x1
            Id: 1, Flags: 0x1

        Set: {720D4AC0-7533-11D0-A5D6-28DB04C10000}
            Id: 0, Flags: 0x1
            Id: 1, Flags: 0x1
            Id: 2, Flags: 0x1
            Id: 3, Flags: 0x1

        Set: {C034FDB0-FF75-47C8-AA3C-EE46716B50C6}
            Id: 1, Flags: 0x1
            Id: 2, Flags: 0x1
            Id: 3, Flags: 0x1

        Set: {4D12807E-55DB-48B8-A466-F15A510F5817}
            Id: 1, Flags: 0x1
```

## !wdfhandle

Также доступно в выходных `!wdfdriverinfo` данных ссылка на wdfhandle для иерархии объектов WDF, связанной с ACX.

dbgcmd

```
Object Hierarchy: !wdfhandle 0x000049f6bfe9d5d8 0xff
```

При нажатии этой ссылки отображается иерархия объектов WDF для драйвера ACX. Эти выходные данные можно использовать для поиска дескрипторов WDF для других объектов ACX и WDF.

dbgcmd

```
0: kd> !wdfhandle 0x000049f6bfe9d5d8 0xff
Treating handle as a KMDF handle!

Dumping WDFHANDLE 0x000049f6bfe9d5d8
============================
Handle type is WDFDRIVER
Refcount: 1
Contexts:
    context:  dt 0xffffb60940162bc0 AcxHdAudio!CODEC_DRIVER_CONTEXT (size is 0x1
bytes)
    <no associated attribute callbacks>

    context:  dt 0xffffb6093ccead20 Acx01000!AfxDriver (size is 0x20 bytes)
    EvtCleanupCallback fffff8038a455780
Acx01000!Acx::WdfCpp::ObjectContext<WDFDRIVER__
*,Acx::AfxDriver>::EvtObjectContextCleanupThunk
    EvtDestroyCallback fffff8038a455740
Acx01000!Acx::WdfCpp::ObjectContext<WDFDRIVER__
*,Acx::AfxDriver>::EvtObjectContextDestroyThunk

Child WDFHANDLEs of 0x000049f6bfe9d5d8:
    !wdfhandle 0x000049f6bfe9d5d8  dt FxDriver 0xffffb60940162a20 Context
ffffb60940162bc0, Context ffffb6093ccead20 Cleanup fffff8038a455780 Destroy
fffff8038a455740
        !wdfdevice 0x000049f6bffba488  dt FxDevice 0xffffb60940045b70 Context
ffffb60940045e60 Cleanup fffff8038a3e3c90, Context ffffb60933f030f0 Cleanup
fffff8038a451910 Destroy fffff8038a451740, Context ffffb60940a26b90, Context
ffffb609409f2e00
            WDF INTERNAL   dt FxDefaultIrpHandler 0xffffb6093fa54130
            WDF INTERNAL   dt FxPkgGeneral 0xffffb609401f5610
            WDF INTERNAL   dt FxWmiIrpHandler 0xffffb609401f6510
            WDF INTERNAL   dt FxPkgIo 0xffffb6093f6d2400
                !wdfqueue 0x000049f6c00114b8  dt FxIoQueue 0xffffb6093ffeeb40
            WDF INTERNAL   dt FxPkgFdo 0xffffb6093f2ae020
            !wdfhandle 0x000049f6bfe099f8  dt FxCmResList 0xffffb609401f6600
            !wdfhandle 0x000049f6bfe0a358  dt FxCmResList 0xffffb609401f5ca0
            !wdfchildlist 0x000049f6c09adb98  dt FxChildList 0xffffb6093f652460
            !wdfiotarget 0x000049f6c092d1d8  dt FxIoTarget 0xffffb6093f6d2e20
```

```
            !wdfqueue 0x000049f6c04904c8  dt FxIoQueue 0xffffb6093fb6fb30 Context
ffffb6093fa8dcd0 Cleanup fffff8038a45a350 Destroy fffff8038a45a310
            !wdfhandle 0x000049f6c09e7ed8  dt FxWorkItem 0xffffb6093f618120
Context ffffb6093f618220
            WDF INTERNAL   dt FxWmiProvider 0xffffb609403d9b50
                WDF INTERNAL   dt FxWmiInstanceExternal 0xffffb60940a11320
            WDF INTERNAL   dt FxWmiProvider 0xffffb609403d9300
                WDF INTERNAL   dt FxWmiInstanceExternal 0xffffb60940a11720
        !wdfdevice 0x000049f6bef1a848  dt FxDevice 0xffffb609410e57b0 Context
ffffb609410e5aa0 Cleanup fffff8038a3e6400, Context ffffb60933f090d0 Cleanup
fffff8038a451910 Destroy fffff8038a451740
            WDF INTERNAL   dt FxDefaultIrpHandler 0xffffb60940c16450
            WDF INTERNAL   dt FxPkgGeneral 0xffffb60940bc9a10
            WDF INTERNAL   dt FxWmiIrpHandler 0xffffb60940bc9380
            WDF INTERNAL   dt FxPkgIo 0xffffb609400ab5c0
                !wdfqueue 0x000049f6beefd588  dt FxIoQueue 0xffffb60941102a70
            WDF INTERNAL   dt FxPkgPdo 0xffffb609410e6020
            !wdfhandle 0x000049f6bf436e58  dt FxCmResList 0xffffb60940bc91a0
            !wdfhandle 0x000049f6bf436d68  dt FxCmResList 0xffffb60940bc9290
            !wdfqueue 0x000049f6bef185c8  dt FxIoQueue 0xffffb609410e7a30 Context
ffffb60940e72ed0 Cleanup fffff8038a45a350 Destroy fffff8038a45a310
            !wdfhandle 0x000049f6bf436b88 [ACXCIRCUIT]  dt FxUserObject
0xffffb60940bc9470 Context ffffb60940bc9520, Context
...
```

## Команды журнала WDF

!wdflogdump может быть полезно для устранения неполадок драйвера ACX, отображая сведения о журнале WDF.

Отображение журнала для ACX с помощью параметра -d driver.

dbgcmd

```
0: kd> !wdflogdump acx01000 -d
Log dump command                          Log ID                  Size
================                          ======                  ====
 !wdflogdump  Acx01000 -a 0xFFFF820527861000  AFX_Client1             4096
 !wdflogdump  Acx01000 -a 0xFFFF8205215C0000  AFX_Log                 4096
Trace searchpath is:

Trace format prefix is: %7!u!: %!FUNC! -
Trying to extract TMF information from -
C:\ProgramData\Dbg\sym\Acx01000.pdb\B13D39B43205B60C07935803D7CB96981\Acx01000.pdb
--- start of log ---
AFX_Client1 1: Acx::AfxCircuit::Register - INFO:ACXCIRCUIT 00007DFAD9CCCCB8
Registered
AFX_Client1 2: Acx::AfxCircuit::PowerUpNotification - INFO:ACXCIRCUIT
00007DFAD9CCCCB8, EvtAcxCircuitPowerUp callback, STATUS_SUCCESS
AFX_Client1 5: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
```

```
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 7: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 8: Acx::AfxMute::EvtMuteEventEnableCallback - INFO:ACXMUTE
00007DFADF997A18, enabled ACXEVENT 00007DFADF996F98
AFX_Client1 10: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 12: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 13: Acx::AfxPin::EvtJackEventEnableCallback - INFO:ACXPIN
00007DFAD4697238, enabled ACXEVENT 00007DFADF9973F8
AFX_Client1 15: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
AFX_Client1 17: Acx::AfxPin::GetModesCount - WARN:ACXPIN 00007DFAD4697238, failed
to get default format for processing mode 9e90ea20-b493-4fd1-a1a8-7e1361a956cf,
0xc0000225(STATUS_NOT_FOUND)
```

Используйте !wdflogdump , чтобы отобразить журнал платформы для определенного драйвера ACX с помощью параметра -f.

dbgcmd

```
0: kd> !wdflogdump AcxHdAudio -f
Trace searchpath is:

Trace format prefix is: %7!u!: %4!s! %!FUNC! -
Trying to extract TMF information from -
C:\ProgramData\Dbg\sym\Wdf01000.pdb\CBDEA3A4F64C17C1752E652A91DD14761\Wdf01000.pdb
Gather log: Please wait, this may take a moment (reading 4024 bytes).
% read so far ... 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
There are 65 log entries
--- start of log ---
16131: 09/03/2023-23:43:07.3233594 imp_WdfRegistryOpenKey - new WDFKEY object open
failed, 0xc0000034(STATUS_OBJECT_NAME_NOT_FOUND)
16132: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleDecrementIo from FxIdleBusy
16133: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleStartTimer from FxIdleDecrementIo
16134: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleTimerRunning from FxIdleStartTimer
16135: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleCancelTimer from FxIdleTimerRunning
16136: 09/03/2023-23:43:07.3233594 FxPowerIdleMachine::ProcessEventLocked -
WDFDEVICE 0x00007DFADB0A5358 !devobj 0xFFFF820521608AF0 entering power idle state
FxIdleCheckIoCount from FxIdleCancelTimer
```

```
16137:
...
```

## См. также

Дополнительные сведения см. в разделе "Отладка потоковой передачи ядра". Пошаговое руководство по отладке с помощью звукового драйвера WDM см. в разделе "Драйверы отладки " Пошаговые лаборатории (режим ядра Sysvad)". Дополнительные сведения о расширениях класса ACX см. в обзоре расширений аудиоклассов ACX.

# Расширения минипорта SCSI (Scsikd.dll и Minipkd.dll)

18.07.2025

Команды расширения, которые полезны для отладки драйверов мини-порта SCSI, можно найти в Scsikd.dll и Minipkd.dll.

Команды расширения Scsikd.dll можно использовать с любой версией Windows. Однако можно использовать только команды расширения Minipkd.dll с Windows XP и более поздними версиями Windows. Команды в Minipkd.dll применимы только к мини-портам на основе SCSIport.

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

# !scsikd.help

Статья • 05.04.2024

Расширение **!scsikd.help** отображает текст справки для команд расширения Scsikd.dll.

```
dbgcmd
```

```
!scsikd.help
```

## DLL-библиотеки

Scsikd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !scsikd.classext

Статья • 05.04.2024

Расширение **!scsikd.classext** отображает указанное устройство класса самонастраивающийся (PnP).

```
dbgcmd
```

```
!scsikd.classext [Device [Level]]
```

## Параметры

*Устройство*
Указывает расширение объекта устройства или устройства класса PnP. Если *устройство* опущено, отображается список всех расширений PnP класса.

*Уровень*
Указывает объем отображаемых сведений. Этот параметр может принимать значение 0, 1 или 2 в качестве значений, при этом 2 дают наиболее подробные сведения. По умолчанию установлено значение 0.

## DLL-библиотеки

Развернуть таблицу

| | |
|---|---|
| **Windows 2000** | Scsikd.dll |
| **Windows XP и более поздние версии** | Scsikd.dll |

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

Ниже приведен пример **отображения !scsikd.classext** :

```
dbgcmd
```

```
0: kd> !scsikd.classext
  ' !scsikd.classext 8633e3f0 '   (             ) "IBM     " / "DDYS-T09170M
" / "S93E" / "        XBY45906"
  ' !scsikd.classext 86347b48 '   (paging device) "IBM     " / "DDYS-T09170M
" / "S80D" / "        VDA60491"
  ' !scsikd.classext 86347360 '   (             ) "UNISYS  " /
"003451ST34573WC " / "5786" / "HN02207500001813O0L6"
  ' !scsikd.classext 861d1898 '   (             ) "" / "MATSHITA CD-ROM CR-
177" / "7T03" / ""

 usage: !classext <class fdo> <level [0-2]>
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !scsikd.scsiext

Статья • 05.04.2024

Расширение **!scsikd.scsiext** отображает подробные сведения о указанном расширении порта SCSI.

dbgcmd

```
!scsikd.scsiext Device
```

## Параметры

*Устройство*
Указывает объект устройства или расширение устройства расширения порта SCSI.

## DLL-библиотеки

Scsikd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

Ниже приведен пример **дисплея !scsikd.scsiext** , в котором расширение порта SCSI было указано объектом функционального устройства (FDO). Это можно получить из **поля DO** или **DevExt** в **отображении !minipkd.adapters** :

dbgcmd

```
kd> !scsikd.scsiext 816f9a40
Scsiport functional device extension at address 816f9af8
Common Extension:
  Initialized
  DO 0x816f9a40  LowerObject 0x816e8030  SRB Flags 00000000
  Current Power (D0,S0)  Desired Power D-1 Idle 00000000
  Current PnP state 0x0    Previous state 0xff
  DispatchTable f9aee200   UsePathCounts (P0, H0, C0)
Adapter Extension:
  Port 2     IsPnp VirtualSlot HasInterrupt
```

```
  LowerPdo 0x816e8030   HwDevExt 0x8170a004   Active Requests 0xffffffff
  MaxBus 0x01   MaxTarget 0x10   MaxLun 0x08
  Port Flags (0x00001000): PD_DISCONNECT_RUNNING
  NonCacheExt 0x81702000  IoBase 0x00002000   Int 0x1a
  RealBus# 0x0  RealSlot# 0x2
  Timeout 0xffffffff   DpcFlags 0x00000000   Sequence 0x00000003
  Srb Ext Header 0x817061a0   No. Requests 0x00000012
  QueueTag BitMap 0x00000000   Hint 0x00000000
  MaxQueueTag 0xfe (@0x816f9c58)
  LuExt Size 0x00000038   SrbExt Size 0x00000188
    SG List Size - Small 17   Large 0
    Emergency  - SrbData 0x816f9830  Blocked List @0x816f9e94
    CommonBuff - Size: 0x00006000    PA: 0x0000000001702000  VA: 0x81702000
    Ke Objects - Int1: 0x8175ba50    Int2: 0x00000000        Dma: 0x816f9340
    Lookaside  - SrbData @ 0x816f9e40 SgList @0x00000000  Remove:
@0x00000000
    Resources  - Raw: 0x817ba190     Translated: 0x81709678
    Port Config 8177fde8
    DeviceMap Handles: Port 0000009c    Busses e12d7b38
  Interrupt Data @0x816f9ce4:
    Flags (0x00000000):
    Ready LUN 0x00000000   Wmi Events 0x00000000
    Completed Request List (@0x816f9ce8): 0 entries
    LUN 816ea0e8 @ (  0,  1,  0) c ev  pnp(00/ff) pow(0 ,0) DevObj 816ea030
```

Ниже приведен пример **отображения !scsikd.scsiext**, где расширение порта SCSI было указано объектом физического устройства (PDO); это можно получить из **поля DevObj** или **LUN** в [отображении адаптеров !minipkd.adapters](#):

dbgcmd

```
kd> !scsikd.scsiext 816ea030
Scsiport physical device extension at address 816ea0e8
Common Extension:
  Initialized
  DO 0x816ea030  LowerObject 0x816f9a40  SRB Flags 00000000
  Current Power (D0,S0)  Desired Power D-1 Idle 0x8176c780
  Current PnP state 0x0    Previous state 0xff
  DispatchTable f9aee180   UsePathCounts (P0, H0, C0)
Logical Unit Extension:
  Address (2, 0, 1, 0) Claimed  Enumerated Visible
  LuFlags (0x00000000):
  Retry 0x00          Key 0x00000000
  Lock 0x00000000  Pause 0x00000000   CurrentLock: 0x00000000
  HwLuExt 0x8177ce10  Adapter 0x816f9af8  Timeout 0xffffffff
  NextLun 0x00000000  ReadyLun 0x00000000
  Pending 0x00000000  Busy 0x00000000     Untagged 0x00000000
  Q Depth 000 (1628450047)   InquiryData 0x816ea206
  DeviceMap Keys: Target 0x0000d0   Lun 00000000
  Bypass SRB_DATA blocks 4 @ 816ea270   List 816ea810
  RS Irp 8177dd80  Srb @ 816eaa0c   MDL @ 816eaa4c
  Request List @0x816ea1f0 is empty
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !scsikd.srbdata

Статья • 05.04.2024

Расширение **!scsikd.srbdata** отображает указанный блок отслеживания SRB_DATA.

```
dbgcmd
```

```
!scsikd.srbdata Address
```

## Параметры

*Адрес*
Задает адрес блока отслеживания SRB_DATA.

## DLL-библиотеки

Scsikd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !minipkd.help

Статья • 05.04.2024

Расширение **!minipkd.help** отображает текст справки для команд расширения Minipkd.dll.

dbgcmd

```
!minipkd.help
```

## DLL-библиотеки

Minipkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

Если появится сообщение об ошибке, аналогичное приведенному ниже, оно указывает на неправильный путь символа и не указывает на правильную версию символов Scsiport.sys.

dbgcmd

```
minipkd error (0) <path> ... \minipkd\minipkd.c @ line 435
```

**Используйте команду Sympath (Задать путь символа),** чтобы отобразить текущий путь и изменить путь. **Используйте команду reload (Reload Module)** для перезагрузки символов из текущего пути.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !minipkd.adapter

Статья • 05.04.2024

Расширение **!minipkd.adapter** отображает сведения о указанном адаптере.

```
dbgcmd
```

```
!minipkd.adapter Address
```

## Параметры

*Адрес*
Указывает адрес адаптера.

## DLL-библиотеки

Minipkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

Адрес адаптера можно найти в **поле DevExt дисплея !minipkd.adapters**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !minipkd.adapters

Статья • 05.04.2024

Расширение **!minipkd.adapters отображает все адаптеры** , работающие с драйвером портов SCSI, которые были определены в системе, и отдельные устройства, связанные с каждым адаптером.

dbgcmd

```
!minipkd.adapters
```

## DLL-библиотеки

Minipkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

Отображается имя драйвера, адрес объекта устройства и адрес расширения устройства для каждого адаптера. Отображение для каждого адаптера также содержит список каждого устройства на адаптере. Отображение для каждого устройства включает адрес расширения устройства, адрес SCSI, адрес объекта устройства и некоторые флаги для устройства. Также включены сведения о состоянии самонастраивающийся и состоянии питания.

Флаги в дисплее описаны в следующей таблице:

Развернуть таблицу

| Флаг | Значение |
| --- | --- |
| c | Утверждал. Указывает, что на устройстве есть драйвер. |
| m | Значение отсутствует. Указывает, что устройство присутствовало на шине в предыдущей проверке, но не |

| Флаг | Значение |
| --- | --- |
| | присутствовало во время последней проверки. |
| e | Перечисленных. Указывает, что устройство было сообщено диспетчеру самонастраивающийся. |
| v | Видимый. Указывает, что устройство было перечислено системой. Этот флаг является более значительным, если он отсутствует для устройства. |
| п | Подкачки. Указывает, что устройство находится в пути к разбиению по страницам. |
| d | Дампа. Указывает, что устройство находится в пути аварийного дампа и будет использоваться для аварийного дампа. |
| ч | Спящий режим. Указывает, что устройство находится в режиме гибернации. |

Ниже приведен пример **отображения !minipkd.adapters** :

dbgcmd

```
0: kd> !minipkd.adapters
Adapter \Driver\lp6nds35     DO 86334a70         DevExt 86334b28
Adapter \Driver\adpu160m     DO 8633da70         DevExt 8633db28
 LUN 862e60f8 @(0,0,0) c ev     pnp(00/ff) pow(0,0) DevObj 862e6040
 LUN 863530f8 @(0,1,0) c ev p d pnp(00/ff) pow(0,0) DevObj 86353040
 LUN 862e50f8 @(0,2,0) c ev     pnp(00/ff) pow(0,0) DevObj 862e5040
 LUN 863520f8 @(0,6,0)   ev     pnp(00/ff) pow(0,0) DevObj 86352040
Adapter \Driver\adpu160m     DO 86376040         DevExt 863760f8
```

Сообщение об ошибке, аналогичное приведенному ниже, указывает на неправильный путь символа и не указывает на правильную версию символов Scsiport.sys или что Windows не определила адаптеры, работающие с драйвером портов SCSI.

dbgcmd

```
minipkd error (0) <path> ... \minipkd\minipkd.c @ line 435
```

**Если команда расширения !minipkd.help** возвращает сведения о справке успешно, символы порта SCSI исправляются.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !minipkd.exports

Статья • 08.07.2024

Расширение **!minipkd.exports** отображает адреса экспорта мини-порта для указанного адаптера.

dbgcmd

```
!minipkd.exports Adapter
```

## Параметры

*Адаптер*
Указывает адрес адаптера.

## DLL-библиотеки

Minipkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !minipkd.lun

Статья • 05.04.2024

Расширение **!minipkd.lun** отображает подробные сведения об указанном расширении логического блока (LUN).

dbgcmd

```
!minipkd.lun LUN
!minipkd.lun Device
```

## Параметры

*LUN*
Указывает адрес LUN.

*Устройство*
Указывает объект физического устройства (PDO) для LUN.

## DLL-библиотеки

Minipkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

LUN обычно называется устройством. Таким образом, это расширение отображает сведения об устройстве на адаптере.

LUN можно указать по адресу (который можно найти в поле LUN **дисплея !minipkd.adapters**), **или его физическим объектом устройства (который можно найти в поле DevObj дисплея !minipkd.adapters).**

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !minipkd.portconfig

Статья • 05.04.2024

Расширение **!minipkd.portconfig** отображает сведения о указанной PORT_CONFIGURATION_INFORMATION структуре данных.

dbgcmd

```
!minipkd.portconfig PortConfig
```

## Параметры

*PortConfig*
Задает адрес структуры данных PORT_CONFIGURATION_INFORMATION.

## DLL-библиотеки

Minipkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

Адрес PortConfig *можно найти в* ***поле "Сведения о конфигурации порта" дисплея !minipkd.adapter***.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !minipkd.req

Статья • 08.07.2024

Расширение **!minipkd.req** отображает сведения обо всех текущих активных запросах на указанном адаптере или устройстве.

```
dbgcmd
```

```
!minipkd.req Adapter
!minipkd.req Device
```

## Параметры

*Адаптер*
Указывает адрес адаптера.

*Устройство*
Указывает объект физического устройства (PDO) для устройства логического модуля (LUN).

## DLL-библиотеки

Minipkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

PDO для LUN можно найти в **поле DevObj дисплея !minipkd.adapters**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !minipkd.srb

Статья • 05.04.2024

Расширение **!minipkd.srb** отображает указанную структуру данных блока запросов SCSI (SRB).

```
dbgcmd
```

```
!minipkd.srb SRB
```

## Параметры

*SRB*
Указывает адрес SRB.

## DLL-библиотеки

Minipkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка минипорта SCSI".

## Замечания

Адреса всех активных запросов в настоящее время можно найти в полях SRB выходных данных команды **!minipkd.req**.

Это расширение отображает состояние SRB, драйвер, к которому он обращается, SCSI, выдаваемый SRB и его адрес, и шестнадцатеричное значение флага. Если 0x10000 задано в значении флага, этот запрос в настоящее время находится в мини-порту.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# Расширения Платформы драйверов Windows (Wdfkd.dll)

18.07.2025

Команды расширения, которые полезны для отладки драйверов, созданных с помощью платформы драйверов Kernel-Mode (KMDF) или версии 2 User-Mode Driver Framework (UMDF 2), реализуются в Wdfkd.dll.

Эти расширения можно использовать в Microsoft Windows XP и более поздних операционных системах. Некоторые расширения имеют дополнительные ограничения; Эти ограничения отмечаются на отдельных справочных страницах.

**Note** When you create a new KMDF or UMDF driver, you must select a driver name that has 32 characters or less. Это ограничение длины определяется в wdfglobals.h. Если имя драйвера превышает максимальную длину, драйвер не будет загружаться.

Дополнительные сведения об использовании этих расширений см. вKernel-Mode отладке Driver Framework.

# !wdfkd.help

Статья • 05.04.2024

Расширение **!wdfkd.help** отображает сведения о всех командах расширения Wdfkd.dll.

dbgcmd

```
!wdfkd.help
```

## DLL-библиотеки

Wdfkd.dll

## Дополнительная информация

**Расширение !wdfkd.help** эквивалентно расширению **!wdfkd.wdfhelp**.

Дополнительные сведения об отладке драйверов на основе платформы см. в разделе Отладка платформы драйверов в режиме ядра.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfchildlist

Статья • 05.04.2024

Расширение **!wdfkd.wdfchildlist** отображает состояние дочернего списка и сведения обо всех описаниях идентификации устройства, которые находятся в дочернем списке.

dbgcmd

```
!wdfkd.wdfchildlist Handle
```

## Параметры

*Дескриптор*
Дескриптор типа WDFCHILDLIST в дочерний список.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

В следующем примере показан **экран !wdfkd.wdfchildlist** .

dbgcmd

```
kd> !wdfchildlist 0x7cc090c8

## Dumping WDFCHILDLIST 0x7cc090c8
---------------------------------------
owning !WDFDEVICE 0x7ca7b1c0
ID description size 0x8
```

```
State:
-----
List is unlocked, changes will be applied immediately
No scans or enumerations are active on the list

Descriptions:
------------
 PDO !WDFDEVICE 0x7cad31c8, ID description 0x83ac4ff4
 +Device WDM !devobj 0x81fb00e8, WDF pnp state WdfDevStatePnpStarted (0x119)
 +Device found in last scan

No pending insertions are in the list.

Callbacks:
---------
 EvtChildListCreateDevice:  wdfrawbusenumtest!RawBus_RawPdo_Create
(f22263b0)
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfcollection

Статья • 08.07.2024

Расширение **!wdfkd.wdfcollection** отображает все объекты, хранящиеся в структуре WDFCOLLECTION.

```
dbgcmd

!wdfkd.wdfcollection Handle
```

## Параметры

*Ручка*
Дескриптор типа WDFCOLLECTION в структуру.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfcommonbuffer

Статья • 08.07.2024

Расширение **!wdfkd.wdfcommonbuffer** отображает сведения о общем объекте буфера WDF.

dbgcmd

```
!wdfkd.wdfcommonbuffer Handle
```

## Параметры

*Ручка*
Дескриптор для общего буферного объекта платформы (WDFCOMMONBUFFER).

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfcrashdump

Статья • 05.04.2024

Расширение **!wdfkd.wdfcrashdump** отображает сведения журнала ошибок и другие сведения о аварийном дампе из мини-файла, если данные присутствуют.

KMDF

dbgcmd

```
!wdfkd.wdfcrashdump [InfoType]
```

UMDF

dbgcmd

```
!wdfkd.wdfcrashdump [DriverName.dll][-d | -f | -m]
```

## Параметры

*InfoType*
Указывает тип отображаемой информации. *InfoType является необязательным* и может быть одним из следующих значений:

**log**
Отображает сведения журнала ошибок, если они доступны в файле аварийного дампа. Это значение по умолчанию.

**Погрузчик**
Отображает динамические драйверы минидумпа.

*DriverName*.dll
Указывает имя драйвера UMDF. Необходимо включить суффикс .dll файла. Если этот необязательный параметр опущен, выходные данные включают метаданные, загруженный список модулей и доступные журналы.

**-d**
Отображает только журналы драйверов.

**-f**
Отображает только журналы платформы.

**-m**
Объединяет журналы платформы и драйверов в их записанном порядке.

# DLL-библиотеки

Wdfkd.dll

## Платформы

KMDF

UMDF 2.15

# Замечания

В этом примере показано, как использовать **!wdfkd.wdfcrashdump** для просмотра сведений о драйверах KMDF. При указании **загрузчика** для *InfoType* выходные данные включают драйверы с динамической привязкой в мини-файл.

dbgcmd

```
0: kd> !wdfcrashdump loader
Retrieving crashdump loader information...
## Local buffer 0x002B4D00, bufferSize 720
-----------------------------------------------
  ImageName      Version    FxGlobals

  Wdf01000       v1.9(6902)
  msisadrv       v1.9(6913) 0x84deb260
  vdrvroot       v1.9(6913) 0x860e8260
  storflt        v1.5(6000) 0x861dfe90
  cdrom          v1.9(6913) 0x84dca008
  intelppm       v1.9(6913) 0x864704a8
  HDAudBus       v1.7(6001) 0x86101c98
  1394ohci       v1.7(6001) 0x8610d2e8
  CompositeBus   v1.9(6913) 0x86505b98
  ObjTestClassExt v1.9(6902) 0x865b7f00
  mqfilter       v1.9(6902) 0x865b8008
  mqueue         v1.9(6902) 0x865b6910
  umbus          v1.9(6913) 0x8618aea0
  monitor        v1.9(6913) 0x86aac1d8
  PEAUTH         v1.5(6000) 0x854e5350
-----------------------------------------------
```

В этом примере показано, как использовать **!wdfkd.wdfcrashdump** для просмотра сведений о драйверах UMDF. Если вы выдаете **!wdfkd.wdfcrashdump** без

параметров, выходные данные включают драйвер, который вызвал сбой, и список всех загруженных драйверов в процессе узла, который завершился сбоем. Вы можете щелкнуть драйверы в этом списке с связанными журналами.

dbgcmd

```
0:001> !wdfkd.wdfcrashdump
Opening minidump at location C:\temp\WudfHost_ext__1312.dmp

Faulting driver: wpptest.dll
Failure type: Unhandled Exception (WUDFUnhandledException)
Faulting thread ID: 2840

Listing all drivers loaded in this host process at the time of the failure:

  ServiceName
  wpptest
  CoverageCx0102
  coverage
  WUDFVhidmini
  ToastMon
  WUDFOsrUsbFilter
```

В приведенном выше примере выходные данные включают тип сбоя, который является типом события в отчете WER. Здесь это может быть **WUDFVerifierFailure** или **WUDFUnhandledException**. Дополнительные сведения см. в разделе "Доступ к метаданным UMDF" в отчетах WER. Выходные данные для UMDF включают код ошибки, если тип события — **WUDFVerifierFailure**.

Чтобы отобразить записи журнала ошибок платформы из полного дампа памяти, дампа памяти ядра или целевого объекта режима динамического ядра, можно также попробовать **расширение !wdfkd.wdflogdump**.

**Дополнительная информация**

Сведения о включении средства записи трассировки в полете для драйвера см. в разделе Использование средства записи трассировки inflight (IFR) в драйверах KMDF и UMDF 2. Дополнительные сведения об отладке драйверов WDF см. в разделе Отладка драйверов WDF. Сведения об отладке KMDF см. в разделе "Отладка платформы драйверов в режиме ядра".

# См. также

**!wdfkd.wdflogdump**

**!wdfkd.wdfsettraceprefix**

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfdevext

Статья • 05.04.2024

Расширение **!wdfkd.wdfdevext** отображает сведения, связанные с **членом DeviceExtension** DEVICE_OBJECT структуры microsoft Windows Driver Model (WDM).

dbgcmd

```
!wdfkd.wdfdevext DeviceExtension
```

## Параметры

*DeviceExtension*
Указатель на расширение устройства.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Ниже приведен пример HdAudBus.sys, который является драйвером KMDF. Используйте **!devnode** , чтобы найти узел устройства с hdAudBus в качестве драйвера функции. Возьмите объект физического устройства (PDO) из выходных данных и передайте его в !devstack**.** Возьмите адрес расширения устройства из выходных **данных !devstack** и передайте его в **!wdfdevext**.

dbgcmd

```
0: kd> !devnode 0 1 hdaudbus
Dumping IopRootDeviceNode (= 0xffffe000002cfd30)
```

```
DevNode 0xffffe000009b7a50 for PDO 0xffffe00000226880
  InstancePath is
"PCI\VEN_8086&DEV_293E&SUBSYS_2819103C&REV_02\3&33fd14ca&0&D8"
  ServiceName is "HDAudBus"
  ...
0: kd> !devstack 0xffffe00000226880
  !DevObj           !DrvObj            !DevExt           ObjectName
  ffffe00001351e20  \Driver\HDAudBus   ffffe000009a3c00
> ffffe00000226880  \Driver\pci        ffffe000002269d0  NTPNP_PCI0009
!DevNode ffffe000009b7a50 :
  DeviceInst is
"PCI\VEN_8086&DEV_293E&SUBSYS_2819103C&REV_02\3&33fd14ca&0&D8"
  ServiceName is "HDAudBus"
0: kd> *
0: kd> !wdfdevext ffffe000009a3c00
Device context is 0xffffe000009a3c00
    context:  dt 0xffffe000009a3c00 HDAudBus!HDAudioDeviceExtension (size is
0xa8 bytes)
    EvtCleanupCallback fffff80001f35950
HDAudBus!HdAudBusEvtDeviceCleanupCallback

!wdfdevice 0x00001fffff65c6e8
!wdfobject 0xffffe000009a3910
```

Ниже приведен пример Wudfrd.sys, который является драйвером функции для части ядра в стеке драйверов UMDF 2. Используйте **!devnode** , чтобы найти узел устройства, имеющий Wudfrd в качестве драйвера функции. Возьмите объект физического устройства (PDO) из выходных данных и передайте его в !devstack**.** Возьмите адрес расширения устройства из выходных **данных !devstack** и передайте его в **!wdfdevext**.

dbgcmd

```
0: kd> !devnode 0 1 wudfrd
Dumping IopRootDeviceNode (= 0xffffe000002cfd30)
DevNode 0xffffe00000a1e530 for PDO 0xffffe00000b15b00
  InstancePath is "ROOT\SAMPLE\0001"
  ServiceName is "WUDFRd"
  ...
0: kd> !devstack 0xffffe00000b15b00
  !DevObj           !DrvObj            !DevExt           ObjectName
  ffffe00000c11040  \Driver\WUDFRd     ffffe00000c11190
> ffffe00000b15b00  \Driver\PnpManager 00000000  00000052
!DevNode ffffe00000a1e530 :
  DeviceInst is "ROOT\SAMPLE\0001"
  ServiceName is "WUDFRd"
0: kd> *
0: kd> !wdfdevext ffffe00000c11190
## Device context is 0xffffe00000c11190

##  UMDF Device Instances for this Redirector extension
```

```
  DriverManagerProcess: 0xffffe00003470500

  ImageName               Ver    DevStack           HostProcess
DeviceID
  MyUmdf2Driver.dll       v2.0   0x000000a5a3ab5f70 0xffffe00000c32900
\Device\00000052
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfdevice

Статья • 05.04.2024

Расширение **!wdfkd.wdfdevice** отображает сведения, связанные с дескриптором типа WDFDEVICE.

dbgcmd

```
!wdfkd.wdfdevice Handle [Flags]
```

## Параметры

*Дескриптор*
Дескриптор типа WDFDEVICE.

*Flags*
Необязательно. Тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов:

Bit 0 (0x1)
Отображение будет содержать подробные сведения об устройстве, такие как связанные дескрипторы WDFCHILDLIST, область синхронизации и уровень выполнения.

Бит 1 (0x2)
Отображение будет содержать подробные сведения о состоянии питания.

Бит 2 (0x4)
Отображение будет содержать подробные сведения о состоянии политики питания.

Бит 3 (0x8)
Отображение будет содержать подробные сведения о состоянии самонастраивающийся (PnP).

Бит 4 (0x10)
Отображение будет включать функции обратного вызова объекта устройства.

## DLL-библиотеки

Wdfkd.dll

## Платформы

KMDF 1, UMDF 2

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

# Замечания

В следующем примере используется **расширение !wdfkd.wdfdevice** в дескрипторе WDFDEVICE, представляющего объект физического устройства (PDO), без указания флагов.

dbgcmd

```
kd> !wdfdevice 0x7cad31c8

# Dumping WDFDEVICE 0x7cad31c8
================================

WDM PDEVICE_OBJECTs:  self 81fb00e8

Pnp state:  119 ( WdfDevStatePnpStarted )
Power state:  31f ( WdfDevStatePowerDx )
Power Pol state:  508 ( WdfDevStatePwrPolWaitingUnarmed )

Parent WDFDEVICE 7ca7b1c0
Parent states:
   Pnp state:  119 ( WdfDevStatePnpStarted )
   Power state:  307 ( WdfDevStatePowerD0 )
   Power Pol state:  565 ( WdfDevStatePwrPolStarted )

No pended pnp or power irps
Device is the power policy owner for the stack
```

В следующем примере отображается тот же объект устройства, что и предыдущий пример, но на этот раз со значением флага 0xF. Это значение флага, сочетание битов 0x1, 0x2, 0x4 и 0x8, приводит к тому, что отображение включает подробные сведения об устройстве, сведения о состоянии питания, сведения о состоянии политики питания и сведения о состоянии PnP.

dbgcmd

```
kd> !wdfdevice 0x7cad31c8 f
```

```
# Dumping WDFDEVICE 0x7cad31c8
================================

WDM PDEVICE_OBJECTs:  self 81fb00e8

Pnp state:  119 ( WdfDevStatePnpStarted )
Power state:  31f ( WdfDevStatePowerDx )
Power Pol state:  508 ( WdfDevStatePwrPolWaitingUnarmed )

Parent WDFDEVICE 7ca7b1c0
Parent states:
   Pnp state:  119 ( WdfDevStatePnpStarted )
   Power state:  307 ( WdfDevStatePowerD0 )
   Power Pol state:  565 ( WdfDevStatePwrPolStarted )

No pended pnp or power irps
Device is the power policy owner for the stack

Pnp state history:
[0] WdfDevStatePnpObjectCreated (0x100)
[1] WdfDevStatePnpInit (0x105)
[2] WdfDevStatePnpInitStarting (0x106)
[3] WdfDevStatePnpHardwareAvailable (0x108)
[4] WdfDevStatePnpEnableInterfaces (0x109)
[5] WdfDevStatePnpStarted (0x119)

Power state history:
[0] WdfDevStatePowerD0StartingConnectInterrupt (0x310)
[1] WdfDevStatePowerD0StartingDmaEnable (0x311)
[2] WdfDevStatePowerD0StartingStartSelfManagedIo (0x312)
[3] WdfDevStatePowerDecideD0State (0x313)
[4] WdfDevStatePowerD0BusWakeOwner (0x309)
[5] WdfDevStatePowerGotoDx (0x31a)
[6] WdfDevStatePowerGotoDxIoStopped (0x31c)
[7] WdfDevStatePowerDx (0x31f)

Power policy state history:
[0] WdfDevStatePwrPolStarting (0x501)
[1] WdfDevStatePwrPolStartingSucceeded (0x502)
[2] WdfDevStatePwrPolStartingDecideS0Wake (0x504)
[3] WdfDevStatePwrPolStartedIdleCapable (0x505)
[4] WdfDevStatePwrPolTimerExpiredNoWake (0x506)
[5] WdfDevStatePwrPolTimerExpiredNoWakeCompletePowerDown (0x507)
[6] WdfDevStatePwrPolWaitingUnarmedQueryIdle (0x509)
[7] WdfDevStatePwrPolWaitingUnarmed (0x508)

WDFCHILDLIST Handles:
 !WDFCHILDLIST 0x7ce710c8

SyncronizationScope is WdfSynchronizationScopeNone
ExecutionLevel is WdfExecutionLevelDispatch
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfdeviceinterrupts

Статья • 08.07.2024

Расширение **!wdfkd.wdfdeviceinterrupts** отображает все объекты прерывания для указанного дескриптора устройства.

dbgcmd

```
!wdfkd.wdfdeviceinterrupts Handle
```

## Параметры

*Ручка*
Дескриптор типа WDFDEVICE.

## DLL-библиотеки

Wdfkd.dll

## Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfdevicequeues

Статья • 08.07.2024

Расширение **!wdfkd.wdfdevicequeues** отображает сведения обо всех объектах очереди платформы, принадлежащих указанному устройству.

dbgcmd

```
!wdfkd.wdfdevicequeues Handle
```

## Параметры

*Ручка*
Дескриптор типа WDFDEVICE.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе Отладка платформы драйвера в режиме ядра и **!wdfkd.wdfqueue**.

## Замечания

В следующем примере показан экран из **расширения !wdfkd.wdfdevicequeues** .

dbgcmd

```
kd> !wdfdevicequeues 0x7cad31c8

# Dumping queues of WDFDEVICE 0x7cad31c8
====================================
## Number of queues: 3
--------------------------------
```

```
Queue: 1 (!wdfqueue 0x7d67d1e8)
    Manual, Not power-managed, PowerOn, Can accept, Can dispatch,
ExecutionLevelDispatch, SynchronizationScopeNone
    Number of driver owned requests: 0
    Number of waiting requests: 0


## This is WDF internal queue for create requests.
---------------------------------
Queue: 2 (!wdfqueue 0x7ce7d1e8)
    Parallel, Power-managed, PowerOff, Can accept, Can dispatch,
ExecutionLevelDispatch, SynchronizationScopeNone
    Number of driver owned requests: 0
    Number of waiting requests: 0


##     EvtIoDefault: (0xf221fad0) wdfrawbusenumtest!EvtIoQueueDefault
---------------------------------
Queue: 3 (!wdfqueue 0x7cd671e8)
    Parallel, Power-managed, PowerOff, Can accept, Can dispatch,
ExecutionLevelDispatch, SynchronizationScopeNone
    Number of driver owned requests: 0
    Number of waiting requests: 0


    EvtIoDeviceControl: (0xf2226ac0)
wdfrawbusenumtest!RawBus_RawPdo_EvtDeviceControl
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfdmaenabler

Статья • 08.07.2024

Расширение **!wdfkd.wdfdmaenabler** отображает сведения о объекте прямого доступа к памяти WDF (DMA) и его транзакциях и общих буферных объектах.

dbgcmd

```
!wdfkd.wdfdmaenabler Handle
```

## Параметры

*Ручка*
Дескриптор объекта включения DMA платформы (WDFDMAENABLER).

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfdmaenablers

Статья • 08.07.2024

Расширение **!wdfkd.wdfdmaenablers** отображает все объекты прямого доступа к памяти WDF (DMA), связанные с указанным объектом устройства. Он также отображает связанные транзакции и общие объекты буфера.

dbgcmd

```
!wdfkd.wdfdmaenablers Handle
```

## Параметры

*Ручка*
Дескриптор объекта устройства платформы (WDFDEVICE).

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfdmatransaction

Статья • 08.07.2024

Расширение **!wdfkd.wdfdmatransaction** отображает сведения о объекте транзакции WDF direct memory access (DMA).

dbgcmd

```
!wdfkd.wdfdmatransaction Handle
```

## Параметры

*Ручка*
Дескриптор объекта транзакции DMA платформы (WDFDMATRANSACTION).

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfdriverinfo

Статья • 05.04.2024

Расширение **!wdfkd.wdfdriverinfo** отображает сведения о указанном драйвере, включая его дерево устройств, версию библиотеки драйвера в режиме ядра (KMDF), с которым был скомпилирован драйвер, и список объектов устройств платформы, созданных драйвером.

dbgcmd

```
!wdfkd.wdfdriverinfo [DriverName [Flags]]
```

## Параметры

*DriverName*
Необязательно. Имя драйвера. *DriverName* не должен включать расширение имени файла .sys.

*Flags*
Необязательно. Флаги, указывающие тип отображаемых сведений. *Флаги* могут быть любым сочетанием следующих битов:

Bit 0 (0x1)
Отображение будет включать параметры проверяющего элемента для драйвера, а также будет содержать количество объектов WDF. Этот флаг можно объединить с битом 6 (0x40) для отображения внутренних объектов.

Бит 4 (0x10)
Отображение будет включать иерархию дескрипторов KMDF для драйвера.

Бит 5 (0x20)
Отображение будет содержать сведения о контексте и функции обратного вызова для каждого дескриптора. Этот флаг действителен только в том случае, если установлен бит 4 (0x10).

Бит 6 (0x40)
Отображение будет содержать дополнительные сведения для каждого дескриптора. Этот флаг действителен только в том случае, если установлен бит 4 (0x10). Этот флаг можно объединить с битом 0 (0x1) для отображения внутренних объектов.

Бит 7 (0x80)
Сведения об дескрипторе будут отображаться в более компактном формате.

Бит 8 (0x100)
Отображение будет выравнивать сведения о внутреннем типе. Этот флаг действителен только в том случае, если установлен бит 4 (0x10).

Bit 9 (0x200)
Отображение будет включать дескриптор, который драйвер потенциально утечки. KMDF версии 1.1 и более поздних версий поддерживают этот флаг. Этот флаг действителен только в том случае, если установлен бит 4 (0x10).

Бит 10 (0x400)
Отображение будет включать дерево устройства в подробной форме.

# DLL-библиотеки

Wdfkd.dll

## Платформы

KMDF 1, UMDF 2

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

# Замечания

Если параметр DriverName *не указан*, используется драйвер по умолчанию. Драйвер по умолчанию можно отобразить с помощью расширения !wdfkd.wdfgetdriver; драйвер по умолчанию можно задать с помощью **расширения !wdfkd.wdfsetdriver.**

В следующем примере показан экран из **расширения !wdfkd.wdfdriverinfo** .

dbgcmd

```
## kd> !wdfdriverinfo wdfrawbusenumtest
---------------------------------
Default driver image name:   wdfrawbusenumtest
WDF library image name:      Wdf01000
 FxDriverGlobals  0x83b7af18
```

```
 WdfBindInfo      0xf22250ec
##    Version        v1.5 build(1234)
---------------------------------
WDFDRIVER: 0x7cbc90d0

    !WDFDEVICE 0x7ca7b1c0
            context:  dt 0x83584ff8 ROOT_CONTEXT (size is 0x1 bytes)
             <no associated attribute callbacks>

    !WDFDEVICE 0x7cad31c8
            context:  dt 0x8352cff0 RAW_PDO_CONTEXT (size is 0xc bytes)
             <no associated attribute callbacks>
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfextendwatchdog

Статья • 08.07.2024

Расширение **!wdfkd.wdfextendwatchdog** расширяет период ожидания (от 10 минут до 24 часов) таймера платформы во время переходов питания.

```
dbgcmd
```

```
!wdfkd.wdfextendwatchdog Handle [Extend]
```

## Параметры

*Ручка*
Дескриптор типа WDFDEVICE.

*Вытягивать*
Необязательно. Значение, указывающее, следует ли включить или отключить расширение периода ожидания. Если *расширение* равно 0, расширение отключено, а время ожидания — 10 минут. Если *расширение* равно 1, расширение включено, а время ожидания — 24 часа. Значение по умолчанию равно 1.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Платформа запускает внутренний таймер наблюдателя каждый раз, когда он вызывает политику питания или функцию обратного вызова события питания для драйвера, который не доступен для работы с питанием (т. е. DO_POWER_PAGABLE

бит ясно). Если функция обратного вызова вызывает разбиение по страницам ввода-вывода и поэтому блокируется, операционная система зависает, так как для обслуживания запроса не доступно ни разбиение на разбиение по страницам.

Если время ожидания истекает, платформа выдает ошибку 0x10D (WDF_VIOLATION). Дополнительные сведения см. в разделе **"Проверка ошибок" 0x10D**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdffindobjects

Статья • 08.07.2024

Расширение **!wdfkd.wdffindobjects** ищет память для объектов WDF.

```
dbgcmd
```

```
!wdfkd.wdffindobjects [StartAddress [Flags]]
```

## Параметры

*StartAddress*
Необязательно. Указывает адрес, с которого должен начинаться поиск. Если это опущено, поиск начнется с самого последнего **поиска !wdfkd.wdffindobjects** .

*Флаги*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x0. *Флаги* нельзя использовать, если *не указан startAddress* .

Bit 0 (0x1)
Отображает подробные выходные данные.

Бит 1 (0x2)
Отображает сведения о внутреннем типе для каждого дескриптора.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

В следующих примерах показаны выходные данные **расширения** **!wdfkd.wdffindobjects** . Флаг 0x1 задан во втором примере.

dbgcmd

```
1: kd> !wdffindobjects 0xfffffa600211b668
  Address             Value               Object
  ------------------  ------------------  ------------------
  0xfffffa600211b668  0x0000000000000008
  0xfffffa600211b670  0xfffffa8002e7b1f0  !WDFREQUEST 0x0000057ffd184e08
  0xfffffa600211b678  0x0000000000000004
  0xfffffa600211b680  0x0000000000000001
  0xfffffa600211b688  0xfffffa8006aa3640  !WDFUSBPIPE 0x0000057ff955c9b8
  0xfffffa600211b690  0x0000000000000000
  0xfffffa600211b698  0xfffff80001e61f78
  0xfffffa600211b6a0  0x0000000000000010
  0xfffffa600211b6a8  0x0000000000010286
  0xfffffa600211b6b0  0xfffffa600211b6c0
  0xfffffa600211b6b8  0x0000000000000000
  0xfffffa600211b6c0  0xfffffa8006aa3640  !WDFUSBPIPE 0x0000057ff955c9b8
  0xfffffa600211b6c8  0x0000057ffd184e08  !WDFREQUEST 0x0000057ffd184e08
  0xfffffa600211b6d0  0x0000000000000000
  0xfffffa600211b6d8  0x0000057ffc51ea18  !WDFMEMORY 0x0000057ffc51ea18
  0xfffffa600211b6e0  0x0000000000000000

1: kd> !wdffindobjects 0xfffffa600211b668 1
  Address             Value               Type    Object
  ------------------  ------------------  ------  ------------------
  0xfffffa600211b668  0x0000000000000008
  0xfffffa600211b670  0xfffffa8002e7b1f0  Object  !WDFREQUEST
0x0000057ffd184e08
  0xfffffa600211b678  0x0000000000000004
  0xfffffa600211b680  0x0000000000000001
  0xfffffa600211b688  0xfffffa8006aa3640  Object  !WDFUSBPIPE
0x0000057ff955c9b8
  0xfffffa600211b690  0x0000000000000000
  0xfffffa600211b698  0xfffff80001e61f78
  0xfffffa600211b6a0  0x0000000000000010
  0xfffffa600211b6a8  0x0000000000010286
  0xfffffa600211b6b0  0xfffffa600211b6c0
  0xfffffa600211b6b8  0x0000000000000000
  0xfffffa600211b6c0  0xfffffa8006aa3640  Object  !WDFUSBPIPE
0x0000057ff955c9b8
  0xfffffa600211b6c8  0x0000057ffd184e08  Handle  !WDFREQUEST
0x0000057ffd184e08
  0xfffffa600211b6d0  0x0000000000000000
  0xfffffa600211b6d8  0x0000057ffc51ea18  Handle  !WDFMEMORY
0x0000057ffc51ea18
  0xfffffa600211b6e0  0x0000000000000000
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfforwardprogresss

Статья • 08.07.2024

Расширение **!wdfkd.wdfforwardprogresss** отображает сведения о ходе выполнения указанного объекта очереди платформы.

```
dbgcmd
```

```
!wdfkd.wdfforwardprogress Handle
```

## Параметры

*Ручка*
Дескриптор объекта очереди платформы.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1

## Дополнительная информация

Дополнительные сведения об отладке драйверов в режиме ядра (KMDF) см. в разделе "Отладка платформы драйверов в режиме ядра".

## Замечания

Это расширение будет выполнено успешно, только если указанный объект очереди платформы настроен для поддержки хода выполнения пересылки. Если это расширение используется с другими объектами, отобразится сообщение об ошибке.

В следующем примере показан экран из **расширения !wdfkd.wdfforwardprogress.**

```
dbgcmd
```

```
kd> !wdfkd.wdfforwardprogress 0x79af3250

# Dumping forward progress fields for WDFQUEUE 0x79af3250
================================================
    ForwardProgressReservedPolicy: UseExamine (0x2)

    Total reserved requests: 44

    Number of available reserved requests in list: 41
            !WDFREQUEST 0x7bcaf4c0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bc67eb0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bccf678 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bb6ce40 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7be30a58 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x79af37d0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bc7f428 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bbd40f0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bd333a8 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bd241d8 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bd594e0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bd80d10 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x78ea2d50 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x792020f0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bc37258 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bbc1fb0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bbc4fb0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7be0cb80 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bc84890 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x78acbd18 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bcf1ad8 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bead540 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7922c0f0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7a34a0f0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x625195d0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bc33640 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bba9f28 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bba44c8 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bb77cd8 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7a2b89a8 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7a41ab88 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bc7cc88 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bd37180 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bca40f0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x64b4af20 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bd01a40 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7a25cfb0 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bba9330 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bd14a40 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7bcc0210 (Reserved) !IRP 0x00000000
            !WDFREQUEST 0x7a54eb00 (Reserved) !IRP 0x00000000

    Number of reserved requests in use: 3
            !WDFREQUEST 0x7bf0ab80 (Reserved) !IRP 0x8438f008
            !WDFREQUEST 0x7bc53ca8 (Reserved) !IRP 0x875f59f0
```

```
            !WDFREQUEST 0x7bced8b8 (Reserved) !IRP 0x85c25348

    Number of undispatched IRP's in list: 0

    EvtIoReservedResourcesAllocate: (0x9a3f1b70)
mqueue!EvtIoAllocateResourcesForReservedRequest
    EvtIoExamineIrp: (0x9a3f19d0) mqueue!EvtIoWdmIrpForForwardProgress
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfgetdriver

Статья • 05.04.2024

Расширение **!wdfkd.wdfgetdriver** отображает имя текущего драйвера по умолчанию.

dbgcmd

```
!wdfkd.wdfgetdriver
```

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfhandle

Статья • 05.04.2024

Расширение **!wdfkd.wdfhandle** отображает сведения о указанном дескрипторе объекта платформы, например тип дескриптора, указателя контекста объекта объекта и указателя базового объекта платформы.

dbgcmd

```
!wdfkd.wdfhandle Handle [Flags]
```

## Параметры

*Дескриптор*
Дескриптор объекта платформы.

*Flags*
Необязательно. Флаги, указывающие тип отображаемых сведений. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x0.

Бит 4 (0x10)
Отображение будет содержать поддерев дочерних объектов для указанного дескриптора.

Бит 5 (0x20)
Отображение будет содержать сведения о контексте и функции обратного вызова для указанного дескриптора. Этот флаг действителен только в том случае, если установлен бит 4 (0x10).

Бит 6 (0x40)
Отображение будет содержать дополнительные сведения для указанного дескриптора. Этот флаг действителен только в том случае, если установлен бит 4 (0x10).

Бит 7 (0x80)
Сведения об дескрипторе будут отображаться в более компактном формате.

Бит 8 (0x100)
Отображение будет выравнивать сведения о внутреннем типе. Этот флаг действителен только в том случае, если установлен бит 4 (0x10).

# DLL-библиотеки

Wdfkd.dll

## Платформы

KMDF 1, UMDF 2

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

# Замечания

В следующем примере показаны выходные данные **расширения !wdfhandle** с битом 4, заданным в *параметре Flags (поэтому выходные данные отображают сведения о дочерних объектах* ).

dbgcmd

```
kd> !wdfhandle 0x7ca7b1c0 10

handle 0x7ca7b1c0, type is WDFDEVICE

Contexts:
    context:  dt 0x83584ff8 ROOT_CONTEXT (size is 0x1 bytes)
     <no associated attribute callbacks>

Child WDFHANDLEs of 0x7ca7b1c0:
    WDFDEVICE 0x7ca7b1c0
        WDFCMRESLIST 0x7ccfb058
        WDFCMRESLIST 0x7cadb058
        WDFCHILDLIST 0x7c72f0c8
        WDFCHILDLIST 0x7cc090c8
        WDFIOTARGET 0x7c9630b8

!wdfobject 0x83584e38
```

В предыдущем примере дескриптор входных данных ссылается на объект WDFDEVICE. Этот конкретный объект устройства содержит пять дочерних объектов — два объекта WDFCMRESLIST, два объекта WDFCHILDLIST и один объект WDFIOTARGET.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfhelp

Статья • 05.04.2024

Расширение **!wdfkd.wdfhelp** отображает сведения о всех командах расширения Wdfkd.dll.

dbgcmd

```
!wdfkd.wdfhelp
```

## DLL-библиотеки

Wdfkd.dll

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Расширение !wdfkd.wdfhelp **эквивалентно расширению !wdfkd.help**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfinterrupt

Статья • 05.04.2024

Расширение **!wdfkd.wdfinterrupt** отображает сведения о объекте WDFINTERRUPT.

```
dbgcmd
```

```
!wdfkd.wdfinterrupt Handle [Flags]
```

## Параметры

*Дескриптор*
Дескриптор объекта WDFINTERRUPT.

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x0.

Bit 0 (0x1)
Отображает подпрограммы службы прерываний (ISR) для таблицы отправки прерываний (IDT), связанной с этим объектом WDFINTERRUPT. Установка этого флага эквивалентна следующему **расширению !wdfinterrupt** с расширением **!idt** .

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

В следующем примере показаны выходные данные **расширения !wdfinterrupt** с битом 0 в *параметре Flags* (поэтому выходные данные отображают сведения об

idT).

dbgcmd

```
kd> !wdfkd.wdfinterrupt 0x7a988698  1

# Dumping WDFINTERRUPT 0x7a988698
========================
  Interrupt Type: Line-based, Connected, Enabled
  Vector: 0xa1 (!idt 0xa1)
  Irql: 0x9
  Mode: LevelSensitive
  Polarity: WdfInterruptPolarityUnknown
  ShareDisposition: CmResourceShareShared
  FloatingSave: FALSE
  Interrupt Priority Policy: WdfIrqPriorityUndefined
  Processor Affinity Policy: WdfIrqPolicyOneCloseProcessor
  Processor Group: 0
  Processor Affinity: 0x3

  dt nt!KINTERRUPT 0x8594eb28

  EvtInterruptIsr: 1394ohci!Interrupt::WdfEvtInterruptIsr (0x8d580552)
  EvtInterruptDpc: 1394ohci!Interrupt::WdfEvtInterruptDpc (0x8d580682)

Dumping IDT:

a1:          85167a58 ndis!ndisMiniportIsr (KINTERRUPT 85167a00)
                                    Wdf01000!FxInterrupt::_InterruptThunk
(KINTERRUPT 85987500)

To get ISR from KINTERRUPT:
   dt <KINTERRUPT> nt!KINTERRUPT ServiceContext
   dt <ServiceContext> wdf01000!FxInterrupt m_EvtInterruptIsr
```

В предыдущем примере отображение завершается двумя предлагаемыми **командами dt (display Type),** которые можно использовать для отображения дополнительных данных.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfiotarget

Статья • 08.07.2024

Расширение **!wdfkd.wdfiotarget** отображает сведения о указанном целевом объекте ввода-вывода.

dbgcmd

```
!wdfkd.wdfiotarget Handle [Flags]
```

## Параметры

*Ручка*
Дескриптор целевого объекта ввода-вывода.

*Флаги*
Необязательно. Флаги, указывающие тип отображаемых сведений. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x0.

Bit 0 (0x1)
Отображение будет содержать сведения о каждом из ожидающих объектов запроса ввода-вывода.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

В следующем примере показан экран из **расширения !wdfkd.wdfiotarget** .

dbgcmd

```
kd> !wdfiotarget 0x7c9630b8

# WDFIOTARGET 8369cf40
========================
WDFDEVICE: 0x7ca7b1c0
Target Device:  !devobj 0x81ede5d8
Target PDO: !devobj 0x822b8a90

Type: Instack target
State:  WdfIoTargetStarted

Requests waiting: 0

Requests sent: 0

Requests sent with ignore-target-state: 0
```

Выходные данные в предыдущем примере включают адрес родительского объекта устройства целевой платформы ввода-вывода, а также адреса структур WDM DEVICE_OBJECT, представляющих объект устройства целевого драйвера и объект физического устройства целевого устройства (PDO).

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfldr

Статья • 05.04.2024

Расширение **!wdfkd.wdfldr** отображает сведения о драйверах, которые в настоящее время динамически привязаны к Платформам драйверов Windows. Это включает платформу драйверов в режиме ядра (KMDF) и среда выполнения платформы драйвера режима пользователя (UMDF).

dbgcmd

```
!wdfkd.wdfldr [DriverName]
```

## Параметры

*DriverName*
Имя драйвера, включая расширение имени файла. Если указать имя драйвера, эта команда отображает подробные сведения об одном драйвере. Если имя диска не указано, эта команда отображает сведения обо всех драйверах, привязанных к Windows Driver Framework.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Ниже приведен пример выходных **данных !wdfldr**.

dbgcmd

```
## 0: kd> !wdfkd.wdfldr

##  KMDF Drivers

##  LoadedModuleList      0xfffff800003b61f8

LIBRARY_MODULE  0xffffe0000039f7c0
  Version       v1.13
  Service       \Registry\Machine\System\CurrentControlSet\Services\Wdf01000
  ImageName     Wdf01000.sys
  ImageAddress  0xfffff800002e7000
  ImageSize     0xc5000
  Associated Clients: 16

  ImageName                     Ver   WdfGlobals         FxGlobals
ImageAddress       ImageSize
  peauth.sys                    v1.7  0xffffe00003a95880 0xffffe00003a956e0
0xfffff80002678000 0x000ab000
  monitor.sys                   v1.11 0xffffe000001abc70 0xffffe000001abad0
0xfffff800022e7000 0x0000e000
  UsbHub3.sys                   v1.11 0xffffe000028a47b0 0xffffe000028a4610
0xfffff8000220b000 0x00077000
##   ...

## Total: 1 library loaded

##  UMDF Drivers

  DriverManagerProcess: 0xffffe00003470500

  ImageName                     Ver
  MyUmdfDriver.dll              v1.11
  SomeUmdf2Driver.dll           v2.0
  MyUmdf2Driver.dll             v2.0
```

Ниже приведен еще один пример, который предоставляет имя драйвера.

dbgcmd

```
0: kd> !wdfldr MyUmdf2Driver.dll

Version    v2.0
Service    \Registry\Machine\System\CurrentControlSet\Services\MyUmdf2Driver

## !wdflogdump  MyUmdf2Driver.dll

##  UMDF Device Instances using MyUmdf2Driver.dll

Process             DevStack           DeviceId
0xffffe00000c32900  a5a3ab5f70         \Device\00000052 !wdfdriverinfo
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdflogdump

Статья • 05.04.2024

Расширение **!wdfkd.wdflogdump** отображает записи журналов WDF in-flight Recorder, если они доступны, для драйвера KMDF или драйвера UMDF 2. Эту команду можно использовать с полным дампом памяти, дампом памяти ядра или целевым объектом режима динамического ядра.

KMDF

```
dbgcmd
!wdfkd.wdflogdump [DriverName][WdfDriverGlobals][-d | -f | -a LogAddress]
```

UMDF

```
dbgcmd
!wdfkd.wdflogdump  [DriverName.dll][HostProcessId][-d | -f | -m]
```

# Параметры

*DriverName*

- KMDF: имя драйвера KMDF. Имя не должно включать расширение имени файла .sys.
- UMDF: имя драйвера UMDF 2. Имя должно содержать расширение .dll имени файла.

*Параметр2*

- KMDF: *WdfDriverGlobals* — адрес *структуры WdfDriverGlobals* . Этот адрес можно определить, выполнив **команду !wdfkd.wdfldr** и найдите поле с меткой WdfGlobals. Кроме того, можно указать @@(Driver! WdfDriverGlobals) в качестве значения адреса, где *Driver* — имя драйвера. Если указан любой *адрес WdfDriverGlobals* , *driverName* игнорируется (хотя он должен быть предоставлен).

- UMDF: *HostProcessId* — идентификатор процесса экземпляра wudfhost.exe. Если указать идентификатор процесса, эта команда отображает записи

журнала для этого процесса. Если идентификатор процесса не указан, эта команда отображает список команд в этой форме:

**!wdflogdump***DriverName* **** *ProcessID*

Если можно определить один процесс, он будет выбран автоматически.

*Параметры* KMDF:

**-d** Отображает только журналы драйверов.

**-f** Отображает только журналы платформы.

—*LogAddress*Отображает определенный журнал драйверов. Если этот параметр используется, необходимо предоставить LogAddress.

UMDF:

**-d** Отображает только журналы драйверов.

**-f** Отображает только журналы платформы.

**-m** Merges framework и driver logs в их записанном порядке.

# DLL-библиотеки

Wdfkd.dll

## Платформы

KMDF 1, UMDF 2

# Замечания

Если параметр DriverName *не указан*, используется имя драйвера по умолчанию. Используйте расширение !wdfkd.wdfgetdriver, чтобы отобразить имя драйвера по умолчанию и использовать **расширение !wdfkd.wdfsetdriver для задания имени драйвера по умолчанию.**

Чтобы отобразить записи журнала ошибок платформы из небольшого дампа памяти, используйте **расширение !wdfkd.wdfcrashdump** .

Сведения о настройке сведений, необходимых отладчику для форматирования сообщений трассировки WPP, см. в разделе !wdfkd.wdftmffile **и**

**!wdfkd.wdfsettraceprefix**.

**Дополнительная информация**

Сведения о включении средства записи трассировки в полете для драйвера см. в разделе Использование средства записи трассировки inflight (IFR) в драйверах KMDF и UMDF 2. Дополнительные сведения об отладке драйверов WDF см. в разделе Отладка драйверов WDF. Сведения об отладке KMDF см. в разделе "Отладка платформы драйверов в режиме ядра".

## См. также

**!wdfkd.wdfcrashdump**

**!wdfkd.wdfsettraceprefix**

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdflogsave

Статья • 05.04.2024

Расширение **!wdfkd.wdflogsave** сохраняет записи журнала ошибок в режиме ядра (KMDF) для указанного драйвера в файл журнала трассировки событий (etl), который можно просмотреть с помощью TraceView.

dbgcmd

```
!wdfkd.wdflogsave [DriverName [FileName]]
```

## Параметры

*DriverName*
Необязательно. Имя драйвера. *DriverName* не должен включать расширение имени файла .sys.

*FileName*
Необязательно. Имя файла, в который должны быть сохранены записи журнала ошибок KMDF. Имя файла не должно включать расширение ETL-файла. Если не указано *имя_файла*, записи журнала ошибок KMDF сохраняются в файле DriverName.etl.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Если параметр DriverName *не указан*, используется имя драйвера по умолчанию. Используйте расширение !wdfkd.wdfgetdriver, чтобы отобразить имя драйвера по умолчанию и использовать **расширение !wdfkd.wdfsetdriver для задания имени драйвера по умолчанию.**

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfmemory

Статья • 08.07.2024

Расширение **!wdfkd.wdfmemory** отображает адрес и размер буфера, связанного с объектом памяти платформы.

dbgcmd

```
!wdfkd.wdfmemory Handle
```

## Параметры

*Ручка*
Дескриптор объекта памяти платформы.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfobject

Статья • 05.04.2024

Расширение **!wdfkd.wdfobject** отображает сведения об указанном объекте платформы.

```
dbgcmd

!wdfkd.wdfobject FrameworkObject
```

## Параметры

*FrameworkObject*
Указатель на объект платформы.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Если средство проверки драйвера в режиме ядра (KMDF) включено для драйвера, а тип общедоступного дескриптора помечен для отслеживания, отображение из **расширения !wdfkd.wdfobject** включает средство отслеживания тегов (т. е. объект отслеживания), как показано в следующем примере.

```
dbgcmd

kd> !wdfobject 0x83584e38

The type for object 0x83584e38 is FxDevice
State: FxObjectStateCreated (0x1)
```

```
!wdfhandle 0x7ca7b1c0

dt FxDevice 0x83584e38

    context:  dt 0x83584ff8 ROOT_CONTEXT (size is 0x1 bytes)
     <no associated attribute callbacks>

Object debug extension 83584e20
   !wdftagtracker 0x83722d80
   Verifier lock 0x831cefa8

   State history:
    [0] FxObjectStateCreated (0x1)
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfopenhandles

Статья • 08.07.2024

Расширение **!wdfkd.wdfopenhandles** отображает сведения обо всех дескрипторах, открытых на указанном устройстве WDF.

dbgcmd

```
!wdfkd.wdfopenhandles Handle [Flags]
```

## Параметры

*Ручка*
Дескриптор объекта устройства платформы (WDFDEVICE).

*Флаги*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x0.

Bit 0 (0x1)
Отображает сведения о контексте объекта файла.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

# !wdfkd.wdfpoolusage

Статья • 05.04.2024

Расширение **!wdfkd.wdfpoolusage** отображает сведения об использовании пула для указанного драйвера, если средство проверки драйвера в режиме ядра (KMDF) включено для драйвера.

dbgcmd

```
!wdfkd.wdfpoolusage [DriverName [SearchAddress] [Flags]]]
```

## Параметры

*DriverName*
Необязательно. Имя драйвера. *DriverName* не должен включать расширение имени файла .sys.

*SearchAddress*
Необязательно. Строка, представляющая адрес памяти. Отображается запись пула , *содержащая SearchAddress* . Если *значение SearchAddress* равно 0 или опущено, отображаются все записи пула драйверов.

*Flags*
Необязательно. Тип отображаемой информации. Этот параметр действителен, только если *SearchAddress* ненулево. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x0.

Bit 0 (0x1)
Отображает подробные выходные данные. Для каждой строки отображаются несколько строк. Если этот флаг не задан, сведения о выделении отображаются в одной строке.

Бит 1 (0x2)
Отображает сведения о внутреннем типе для каждого дескриптора.

Бит 2 (0x4)
Отображает вызывающий объект каждой записи пула.

## DLL-библиотеки

Wdfkd.dll

## Платформы

KMDF 1, UMDF 2

# Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

# Замечания

Если параметр DriverName *не указан*, используется драйвер по умолчанию. Драйвер по умолчанию можно отобразить с помощью расширения !wdfkd.wdfgetdriver; драйвер по умолчанию можно задать с помощью **расширения !wdfkd.wdfsetdriver.**

В следующем примере показаны выходные данные расширения !wdfpoolusage**, если выделение пула** не отмечено, а *значение Flags* имеет значение 0.

dbgcmd

```
## kd> !wdfpoolusage wdfrawbusenumtest 0 0
---------------------------------
## FxDriverGlobals 83b7af18 pool stats
---------------------------------
Driver Tag: 'RawB'
15126 NonPaged Bytes, 548 Paged Bytes
94 NonPaged Allocations, 10 Paged Allocations
15610 PeakNonPaged Bytes, 752 PeakPaged Bytes
100 PeakNonPaged Allocations, 14 PeakPaged Allocations

pool 82dbae00, Size  512 Tag 'RawB', NonPaged, Caller:
Wdf01000!FxVerifierLock::AllocateThreadTable+5d
```

В следующем примере показаны выходные данные из **!wdfpoolusage** , отображаемые при значении *флагов* 1. (Обратите внимание, что многоточие (...) во второй строке указывает на пропуск некоторых выходных данных, аналогичных приведенному в предыдущем примере.)

dbgcmd

```
kd> !wdfpoolusage wdfrawbusenumtest 0 1
. . .
100 PeakNonPaged Allocations, 14 PeakPaged Allocations

Client alloc starts at 82dbae00
Size  512 Tag 'RawB'
```

```
NonPaged (0x0)
Caller:  Wdf01000!FxVerifierLock::AllocateThreadTable+5d
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfqueue

Статья • 08.07.2024

Расширение **!wdfkd.wdfqueue** отображает сведения о указанном объекте очереди платформы и объектах запроса платформы, которые находятся в очереди.

dbgcmd

```
!wdfkd.wdfqueue Handle
```

## Параметры

*Ручка*
Дескриптор объекта очереди платформы.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

В следующем примере показан экран из **расширения !wdfkd.wdfqueue** .

dbgcmd

```
kd> !wdfqueue 0x7ce7d1e8

# Dumping WDFQUEUE 0x7ce7d1e8
=========================
Parallel, Power-managed, PowerOff, Can accept, Can dispatch,
ExecutionLevelDispatch, SynchronizationScopeNone
    Number of driver owned requests: 0
    Number of waiting requests: 0
```

```
    EvtIoDefault: (0xf221fad0) wdfrawbusenumtest!EvtIoQueueDefault
```

Очередь в предыдущем примере настроена для параллельной отправки, управляется питанием, но в настоящее время находится в состоянии off и может принимать и отправлять запросы.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfrequest

Статья • 08.07.2024

Расширение **!wdfkd.wdfrequest** отображает сведения о указанном объекте запроса платформы и пакете запросов WDM I/O, связанном с объектом запроса.

dbgcmd

```
!wdfkd.wdfrequest Handle
```

## Параметры

*Ручка*
Дескриптор объекта запроса платформы.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfsearchpath

Статья • 05.04.2024

Расширение **!wdfkd.wdfsearchpath** задает путь поиска к файлам форматирования для записей журнала ошибок Driver Framework в режиме ядра (KMDF).

```
dbgcmd
```

```
!wdfkd.wdfsearchpath Path
```

## Параметры

*Путь*
Путь к каталогу, который содержит файлы форматирования KMDF.

## DLL-библиотеки

Wdfkd.dll

## Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Файлы форматирования KMDF включены в комплект драйверов Windows (WDK). Путь к файлам форматирования зависит от каталога установки WDK и установленной версии WDK. Файлы форматирования KMDF имеют расширение tmf (форматирование сообщений трассировки). Чтобы определить путь поиска, перейдите к установке WDK или выполните поиск по именам файлов формы Wdf*VersionNumber.tmf*. В следующем примере показано, как использовать **расширение !wdfkd.wdfsearchpath** .

```
dbgcmd
```

```
kd> !wdfsearchpath C:\WinDDK\7600\tools\tracing\amd64
```

Переменная среды TRACE_FORMAT_SEARCH_PATH также управляет путем поиска, но **расширение !wdfkd.wdfsearchpath** имеет приоритет над путем поиска, который TRACE_FORMAT_SEARCH_PATH указывает.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfsettraceprefix

Статья • 05.04.2024

Расширение **!wdfkd.wdfsettraceprefix** задает строку формата префикса трассировки.

```
dbgcmd
```

```
!wdfkd.wdfsettraceprefix String
```

## Параметры

*String*
Строка префикса трассировки.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Строка префикса трассировки добавляется к каждому сообщению трассировки в журнале ошибок Driver Framework (KMDF). Переменная среды TRACE_FORMAT_PREFIX также управляет строкой префикса трассировки.

Формат строки префикса трассировки определяется средствами трассировки Microsoft Windows. Дополнительные сведения о формате этой строки и его настройке см. в разделе "Префикс сообщения трассировки" раздела "Средства разработки драйверов" в комплекте драйверов Windows (WDK).

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfsetdriver

Статья • 05.04.2024

Расширение **!wdfkd.wdfsetdriver** задает имя драйвера driver Framework (KMDF) по умолчанию, к которому применяются команды расширения отладчика.

dbgcmd

```
!wdfkd.wdfsetdriver DriverName
```

## Параметры

*DriverName*
Имя драйвера. *DriverName* не должен включать расширение имени файла .sys.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Расширение **!wdfkd.wdfsetdriver** задает имя драйвера по умолчанию. Это имя можно использовать с другими **расширениями wdfkd** , которые в противном случае требуют указания имени драйвера.

Чтобы получить имя текущего драйвера KMDF по умолчанию, используйте **расширение !wdfkd.wdfgetdriver** .

---

## Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfspinlock

Статья • 08.07.2024

Расширение **!wdfkd.wdfspinlock** отображает сведения об объекте spin-lock платформы. Эти сведения включают историю приобретения спин-блокировки и время хранения блокировки.

dbgcmd

```
!wdfkd.wdfspinlock Handle
```

## Параметры

*Ручка*
Дескриптор типа WDFSPINLOCK.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdftagtracker

Статья • 05.04.2024

Расширение **!wdfkd.wdftagtracker** отображает все доступные сведения тега (включая значение тега, строку, файл и время) для указанного средства отслеживания тегов.

dbgcmd

```
!wdfkd.wdftagtracker TagObjectPointer [Flags]
```

## Параметры

*TagObjectPointer*
Указатель на средство отслеживания тегов.

*Flags*
Необязательно. Тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x0.

Bit 0 (0x1)
Отображает журнал операций получения и операций выпуска объекта.

Бит 1 (0x2)
Отображает номер строки объекта в шестнадцатеричном виде вместо десятичного.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Чтобы получить указатель на средство отслеживания тегов, используйте **расширение !wdfkd.wdfobject** во внутреннем указателе объекта платформы.

Чтобы использовать отслеживание тегов, необходимо включить средство проверки драйвера в режиме ядра (KMDF) и обрабатывать отслеживание в реестре. Оба этих параметра хранятся в подразделе "Параметры драйвера **\Wdf** " ключа **HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services** .

Чтобы включить средство проверки KMDF, задайте ненулевое значение для **VerifierOn**.

Чтобы включить отслеживание дескрипторов, задайте для параметра **TrackHandles** имя одного или нескольких типов объектов или укажите звездочку (*) для отслеживания всех типов объектов. Например, в следующем примере указывается отслеживание ссылок на все объекты WDFDEVICE и WDFQUEUE.

```text
TrackHandles: MULTI_SZ: WDFDEVICE WDFQUEUE
```

При включении отслеживания дескрипторов для типа объекта платформа отслеживает ссылки, принятые на любой объект этого типа. Этот параметр полезен при поиске утечек памяти драйвера, из-за которые не удается найти ссылки. **TrackHandles** работает только в том случае, если средство проверки KMDF включено.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdftmffile

Статья • 05.04.2024

Расширение **!wdfkd.wdftmffile** задает файл формата сообщения трассировки (.tmf) для использования при форматировании отладчиком записей журналов ошибок Driver Framework (KMDF) для **расширений !wdfkd.wdflogdump или !wdfkd.wdfcrashdump**.

dbgcmd

```
!wdfkd.wdftmffile TMFpath
```

## Параметры

*TMFpath*
Путь, содержащий TMF-файл.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Если драйвер использует версию KMDF выше 1.11, необходимо использовать расширение !wdfkd.wdftmffile, прежде чем использовать **расширения !wdfkd.wdflogdump или !wdfkd.wdfcrashdump**.

Начиная с KMDF версии 1.11, файл символов библиотеки платформы (например, wdf01000.pdb) содержит записи формата сообщений трассировки (TMF). Начиная с версии отладчика ядра Windows 8 расширения платформы драйверов в режиме

ядра (Wdfkd.dll) считывают записи из PDB-файла. В результате, если драйвер использует KMDF версии 1.11 или более поздней версии, и вы используете отладчик ядра из Windows 8 или более поздней версии, вам не нужно использовать **!wdfkd.wdftmffile**. Необходимо включить каталог, содержащий файл символов, в путь к символам отладчика. Целевой компьютер отладки может работать с любой операционной системой, поддерживающей KMDF.

В следующем примере показано, как использовать **расширение !wdfkd.wdftmffile** из корневого каталога WDK для KMDF версии 1.5.

```
dbgcmd

kd> !wdftmffile tools\tracing\<platform>\wdf1005.tmf
```

Обратите внимание, что путь может отличаться для используемой версии пакета драйверов Windows (WDK). Кроме того, обратите внимание, что имя TMF-файла представляет версию KMDF, которую вы используете. Например, Wdf1005.tmf — это tmf-файл для KMDF версии 1.5.

Сведения о просмотре журнала KMDF во время сеанса отладки см. в разделе "Использование средства ведения журнала событий Платформы".

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdftraceprtdebug

Статья • 08.07.2024

Расширение **!wdfkd.wdftraceprtdebug** включает и отключает режим диагностики Traceprt.dll, который создает подробные сведения об отладке.

dbgcmd

```
!wdfkd.wdftraceprtdebug {on | off}
```

## Параметры

**on**

Включает режим диагностики Traceprt.dll.

**от**

Отключает режим диагностики Traceprt.dll.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Вы должны использовать расширение !wdfkd.wdftraceprtdebug только в направлении технической поддержки.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfumdevstack

Статья • 05.04.2024

Расширение **!wdfkd.wdfumdevstack** отображает подробные сведения о стеке устройств UMDF в неявном процессе.

dbgcmd

```
!wdfkd.wdfumdevstack DevstackAddress [Flags]
```

## Параметры

*DevstackAddress*
Указывает адрес стека устройств для отображения сведений. Вы можете использовать **!wdfkd.wdfumdevstacks** для получения адресов стека устройств UMDF в неявном процессе.

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x01.

Bit 0 (0x01)
Отображает подробные сведения о стеке устройств.

Бит 7 (0x80)
Отображает сведения о внутренней платформе.

## DLL-библиотеки

Wdfkd.dll

## Платформы

UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

# Замечания

Эту команду можно использовать в сеансе отладки в режиме ядра или в сеансе отладки в пользовательском режиме, подключенном к процессу узла UMDF (wudfhost.exe).

Эта команда отображает ту же информацию, что и команда пользовательского режима !wudfext.umdevstack.

Ниже приведен пример использования **!wdfumdevstack**. Сначала используйте **!wdfumdevstacks для отображения стеков** устройств UMDF в неявном процессе.

dbgcmd

```
0: kd> !wdfkd.wdfumdevstacks
Number of device stacks: 1
  Device Stack: 0x000000a5a3ab5f70     Pdo Name: \Device\00000052
    Active: Yes
    Number of UM devices: 1
    Device 0
      Driver Config Registry Path: MyUmdf2Driver
      UMDriver Image Path:
C:\WINDOWS\System32\drivers\UMDF\MyUmdf2Driver.dll
      FxDriver: 0xa5a3acaaa0
      FxDevice: 0xa5a3ac4fc0
      Open UM files (use !wdfumfile <addr> for details): <None>
      Device XFerMode: Deferred RW: Buffered CTL: Buffered
      DevStack XFerMode: Deferred RW: Buffered CTL: Buffered
```

Предыдущие выходные данные показывают, что в неявном процессе существует один стек устройств UMDF. Вы также можете увидеть, что стек устройств имеет один объект устройства (число устройств единой системы обмена сообщениями: 1).

Предыдущие выходные данные отображают адрес стека устройств (0x000000a5a3ab5f70). Чтобы получить подробные сведения о стеке устройств, передайте свой адрес **в !wdfumdevstack**. В этом примере для параметра Flags *необходимо* 0x80 включить сведения о платформе.

dbgcmd

```
0: kd> !wdfkd.wdfumdevstack 0x000000a5a3ab5f70 0x80
  Device Stack: 0x000000a5a3ab5f70     Pdo Name: \Device\00000052
    Active: Yes
    Number of UM devices: 1
    Device 0
      Driver Config Registry Path: MyUmdf2Driver
      UMDriver Image Path:
C:\WINDOWS\System32\drivers\UMDF\MyUmdf2Driver.dll
```

```
        FxDriver: 0xa5a3acaaa0
        FxDevice: 0xa5a3ac4fc0
        Open UM files (use !wdfumfile <addr> for details): <None>
        Device XFerMode: Deferred RW: Buffered CTL: Buffered
        Internal Values:
          wudfhost!WudfDriverAndFxInfo 0x000000a5a3ac21b8
          IUMDFramework: 0x0000000000000000
          IFxMessageDispatch: 0x000000a5a3aba630
          FxDevice 0x000000a5a3ac4fc0
          Modules:
            Driver: wudfhost!CWudfModuleInfo 0x000000a5a3ac18f0
            Fx:     wudfhost!CWudfModuleInfo 0x000000a5a3aca7a0
            wudfx02000!FxDriver: 0x000000a5a3acaaa0
        DevStack XFerMode: Deferred RW: Buffered CTL: Buffered
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfumdevstacks

Статья • 05.04.2024

Расширение **!wdfkd.wdfumdevstacks** отображает сведения обо всех стеках устройств UMDF в неявном процессе.

dbgcmd

```
!wdfkd.wdfumdevstacks [Flags]
```

## Параметры

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x01.

Bit 0 (0x01)
Отображает подробные сведения о каждом стеке устройств.

Бит 7 (0x80)
Отображает сведения о внутренней платформе.

## DLL-библиотеки

Wdfkd.dll

## Платформы

UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Эту команду можно использовать в сеансе отладки в режиме ядра или в сеансе отладки в пользовательском режиме, подключенном к процессу узла UMDF (wudfhost.exe).

Эта команда отображает те же сведения, что и команда в пользовательском режиме !wudfext.umdevstacks.

Перед использованием этой команды используйте **!process** для получения списка всех процессов узла UMDF.

dbgcmd

```
0: kd> !process 0 0 wudfhost.exe
PROCESS ffffe00000c32900
    SessionId: 0  Cid: 079c    Peb: 7ff782537000  ParentCid: 037c
    DirBase: 607af000  ObjectTable: ffffc00009807940  HandleCount: <Data Not
Accessible>
    Image: WUDFHost.exe
```

Предыдущие выходные данные показывают, что существует один процесс узла UMDF; то есть существует один экземпляр wudfhost.exe.

Затем используйте **процесс .process** , чтобы задать неявный процесс wudfhost.exe.

dbgcmd

```
0: kd> .process /P ffffe00000c32900
Implicit process is now ffffe000`00c32900
.cache forcedecodeptes done
```

Теперь используйте **!wdfkd.wdfumdevstacks** для отображения стека устройств UMDF в неявном процессе (wudfhost.exe).

dbgcmd

```
0: kd> !wdfkd.wdfumdevstacks
Number of device stacks: 1
  Device Stack: 0x000000a5a3ab5f70     Pdo Name: \Device\00000052
    Active: Yes
    Number of UM devices: 1
    Device 0
      Driver Config Registry Path: MyUmdf2Driver
      UMDriver Image Path:
C:\WINDOWS\System32\drivers\UMDF\MyUmdf2Driver.dll
      FxDriver: 0xa5a3acaaa0
      FxDevice: 0xa5a3ac4fc0
      Open UM files (use !wdfumfile <addr> for details): <None>
      Device XFerMode: Deferred RW: Buffered CTL: Buffered
      DevStack XFerMode: Deferred RW: Buffered CTL: Buffered
```

Предыдущие выходные данные показывают, что в неявном процессе существует один стек устройств UMDF. Вы также можете увидеть, что стек устройств имеет

один объект устройства (число устройств единой системы обмена сообщениями: 1).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfumdownirp

Статья • 05.04.2024

Расширение **!wdfkd.wdfumdownirp** отображает пакет запроса в режиме ядра (IRP), связанный с указанным IRP в пользовательском режиме. Эта команда используется в двух шагах. См. заметки.

dbgcmd

```
!wdfkd.wdfumdownirp UmIrp [FileObject]
```

## Параметры

*UmIrp*
Указывает адрес IRP в пользовательском режиме. Вы можете использовать **!wdfkd.wdfumirps** для получения адресов UM IRPs в неявном процессе.

*FileObject*
Задает адрес **структуры _FILE_OBJECT** . Сведения о том, как получить этот адрес, см. в разделе "Примечания".

## DLL-библиотеки

Wdfkd.dll

## Платформы

UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Эту команду можно использовать в сеансе отладки в режиме ядра или в сеансе отладки в пользовательском режиме, подключенном к процессу узла UMDF (wudfhost.exe).

Чтобы использовать эту команду, выполните следующие действия.

1. Введите эту команду, передав только адрес IRP в пользовательском режиме. Команда отображает дескриптор.
2. Передайте отображаемый дескриптор команде **!handle** . В выходных **данных !handle** найдите адрес _FILE_OBJECT структуры.
3. Введите эту команду еще раз, передавая как адрес IRP в пользовательском режиме, так и адрес **структуры _FILE_OBJECT** .

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfumfile

Статья • 05.04.2024

Расширение **!wdfkd.wdfumfile** отображает сведения о файле внутри стека UMDF.

```
dbgcmd
```

```
!wdfkd.wdfumfile Address
```

## Параметры

*Адрес*
Указывает адрес файла UMDF внутри стека для отображения сведений о нем.

## DLL-библиотеки

Wdfkd.dll

## Платформы

UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Эту команду можно использовать в сеансе отладки в режиме ядра или в сеансе отладки в пользовательском режиме, подключенном к процессу узла UMDF (wudfhost.exe).

Эта команда отображает те же сведения, что и команда пользовательского режима !wudfext.umfile.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfumirp

Статья • 05.04.2024

Расширение **!wdfkd.wdfumirp** отображает сведения о пакете запросов в режиме пользователя (UM IRP).

```
dbgcmd
```

```
!wdfkd.wdfumirp Address
```

## Параметры

*Адрес*
Указывает адрес IRP единой системы обмена сообщениями для отображения сведений. Вы можете использовать **!wdfkd.wdfumirps** для получения адресов UM IRPs в неявном процессе.

## DLL-библиотеки

Wdfkd.dll

## Платформы

UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Эту команду можно использовать в сеансе отладки в режиме ядра или в сеансе отладки в пользовательском режиме, подключенном к процессу узла UMDF (wudfhost.exe).

Эта команда отображает ту же информацию, что и команда в пользовательском режиме !wudfext.umirp.

Вы можете использовать **!process** для получения списка всех процессов узла UMDF, и можно использовать **процесс .,** чтобы задать неявный процесс одному из узлов UMDF. Подробный пример см. в разделе !wdfkd.wdfumdevstacks.

Ниже показано, как использовать **!wdfkd.wdfumirps** и **!wdfkd.wdfumirp** для отображения сведений об отдельном IRP UM.

dbgcmd

```
0: kd> !wdfkd.wdfumirps
Number of pending IRPS: 0x4
####  CWudfIrp     Current Type           UniqueId KernelIrp         Device
Stack
----  ---------------  ------------------------------------------------
----
...
0003  1ab9eae370   Power (WAIT_WAKE)          0     ffffe00000c53010
1ab9eaa6d0

0: kd> !wdfkd.wdfumirp 1ab9eae370
UM IRP: 0x0000001ab9eae370  UniqueId: 0x0  Kernel Irp: 0xffffe00000c53010
  Type: Power (WAIT_WAKE)
  ClientProcessId: 0x0
  Device Stack: 0x0000001ab9eaa6d0
  IoStatus
    hrStatus: 0x0
    Information: 0x0
  Total number of stack locations: 2
  CurrentStackLocation: StackLocation[ 0 ]
  > StackLocation[ 0 ]
      FxDevice:   (None)
      Completion:
        Callback:   0x0000000000000000
        Context:    0x0000001ab9ebc750
    StackLocation[ 1 ]
    ...
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfumirps

Статья • 05.04.2024

Расширение **!wdfkd.wdfumirps** отображает список ожидающих пакетов запросов ввода-вывода в режиме пользователя (UM IRPs) в неявном процессе.

dbgcmd

```
!wdfkd.wdfumirps NumberOfIrps Flags
```

## Параметры

*NumberOfIrps*
Необязательно. Указывает количество ожидающих операций irps единой системы обмена сообщениями для отображения сведений о нем. Если *NumberOfIrps* является звездочкой (*) или опущен, будут отображаться все UM IRPs.

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x01.

Bit 0 (0x01)
Отображает сведения о ожидающих irps.

## DLL-библиотеки

Wdfkd.dll

## Платформы

UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Эту команду можно использовать в сеансе отладки в режиме ядра или в сеансе отладки в пользовательском режиме, подключенном к процессу узла UMDF (wudfhost.exe).

Эта команда отображает те же сведения, что и команда в пользовательском режиме !wudfext.umirps.

Вы можете использовать **!process** для получения списка всех процессов узла UMDF, и можно использовать **процесс .,** чтобы задать неявный процесс одному из узлов UMDF. Подробный пример см. в разделе !wdfkd.wdfumdevstacks.

Ниже приведен пример выходных **данных !wdfkd.wdfumirps**.

dbgcmd

```
0: kd> !wdfkd.wdfumirps
Number of pending IRPS: 0x4
####  CWudfIrp     Current Type           UniqueId KernelIrp         Device
Stack
----  ---------------  ---------------------------------------------------
----
0000  1ab9e90c40   WdfRequestUndefined        0     0
1ab9eaa6d0
0001  1ab9ebfa90   WdfRequestInternalIoctl    0     0
1ab9eaa6d0
0002  1ab9ebfd10   WdfRequestInternalIoctl    0     0
1ab9eaa6d0
0003  1ab9eae370   Power (WAIT_WAKE)          0     ffffe00000c53010
1ab9eaa6d0
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfumtriage

Статья • 05.04.2024

Расширение **!wdfkd.wdfumtriage** отображает сведения обо всех устройствах UMDF в системе, включая объекты устройств, соответствующий процесс узла, загруженные драйверы и расширения классов, стек устройств PnP, узлы устройств PnP, отправленные IRPs и состояние проблемы при необходимости.

```
dbgcmd
```

```
!wdfkd.wdfumtriage
```

## DLL-библиотеки

Wdfkd.dll

## Платформы

UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Эту команду можно использовать в сеансе отладки в режиме ядра.

Ниже приведен пример выходных **данных !wdfkd.wdfumtriage**.

```
0: kd> !wdfkd.wdfumtriage
-----------------------------
WudfRd Driver Object Info
-----------------------------
dt WudfRd!RdDriver 0xffffffff9a3e9ee8
Driver manager process 0x9a291680(Switch Context) (Dump Process)
HostFailKdDebugBreak 1

    -----------------------------
    UMDF FDO Info
    -----------------------------
    dt WudfRd!RdFdoDevice 0xffffffff9a3ce9c8
    dt WudfRd!RdProcess 0xffffffffa5863828
    Device instance path Root\umdf2\FusionV2
    !devnode 0xffffffff8aeca600
    !devstack 0xffffffff8aeca800
    # of UMDF drivers 2
    1)SensorsCx.dll
    2)FusionV2.dll

    -----------------------------
    UMDF FDO Info
    -----------------------------
    dt WudfRd!RdFdoDevice 0xffffffffa5a26758
    dt WudfRd!RdProcess 0xffffffffa5863828
    Device instance path ACPI\QCOM2495\2&daba3ff&0
    !devnode 0xffffffff8ee50e30
    !devstack 0xffffffff8ee43a00
    # of UMDF drivers 2
    1)SensorsCx.dll
    2)qcSensors8626.dll

    -----------------------------
    UMDF FDO Info
    -----------------------------
    dt WudfRd!RdFdoDevice 0xffffffffa6be4980
    dt WudfRd!RdProcess 0xffffffffa5863828
    Device instance path SWD\umdf2\SDOV2
    !devnode 0xffffffffa6be8980
    !devstack 0xffffffff9a297aa0
    # of UMDF drivers 2
    1)SensorsCx.dll
    2)SdoV2.dll

        -----------------------------
        Driver host process Info
        -----------------------------
        Host Process 0x9a3fdb80(Switch Context) (Dump Process)
        dt WudfRd!RdProcess 0xffffffffa5863828
        Process timeout 60 seconds

        I/O Port
        WudfPf!WdfLpcCommPort a583bc30
        # of pending message with timeout 0
        Pending Messages:
            WUDFPf!WudfLpcMessage 9a3ec750        !irp 0xffffffffa9a27008        (03/00)         - id 00000000000000ae

        I/O Cancel Port
        WudfPf!WdfLpcCommPort 9a3f1008
        # of pending message with timeout 0

        System Event Port
        WudfPf!WdfLpcCommPort a585af78
        # of pending message with timeout 0

        Non-state changing events port
        WudfPf!WdfLpcCommPort a585a498
        # of pending message with timeout 0
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfusbdevice

Статья • 05.04.2024

Расширение !wdfkd.wdfusbdevice отображает сведения об указанном объекте USB-устройства в режиме ядра (KMDF). Эта информация включает все USB-интерфейсы и каналы, настроенные для каждого интерфейса.

dbgcmd

```
!wdfkd.wdfusbdevice Handle [Flags]
```

## Параметры

*Дескриптор*
Дескриптор объекта USB-устройства WDFUSBDEVICE, типизированного с типом WDFUSBEVICE.

*Flags*
Необязательно. Шестнадцатеричное значение, которое изменяет тип возвращаемой информации. Значение по умолчанию — 0x0. Флаги могут быть любым сочетанием следующих битов:

Bit 0 (0x1)
Отображение будет содержать свойства целевого объекта ввода-вывода.

Бит 1 (0x2)
Отображение будет содержать свойства целевого объекта ввода-вывода для каждого объекта USB-канала.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfusbinterface

Статья • 08.07.2024

Расширение **!wdfkd.wdfusbinterface** отображает сведения об указанном объекте USB-интерфейса Driver Framework (KMDF) ядра, включая возможные и текущие параметры.

dbgcmd

```
!wdfkd.wdfusbinterface Handle [Flags]
```

## Параметры

*Ручка*
Дескриптор объекта ИНТЕРФЕЙСА USB с типом WDFUSBINTERFACE.

*Флаги*
Необязательно. Шестнадцатеричное значение, которое изменяет тип возвращаемой информации. Значение по умолчанию — 0x0. Флаги могут быть любым сочетанием следующих битов:

Bit 0 (0x1)
Отображение будет содержать свойства целевого объекта ввода-вывода для каждого объекта USB-канала KMDF.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfusbpipe

Статья • 08.07.2024

Расширение **!wdfkd.wdfusbpipe** отображает сведения о целевом объекте ввода-вывода объекта USB-канала в режиме ядра (KMDF).

dbgcmd

```
!wdfkd.wdfusbpipe Handle [Flags]
```

## Параметры

*Ручка*
Дескриптор объекта USB-канала WDFUSBPIPE.

*Флаги*
Необязательно. Шестнадцатеричное значение, которое изменяет тип возвращаемой информации. Значение по умолчанию — 0x0. Флаги могут быть любым сочетанием следующих битов:

Bit 0 (0x1)
Отображение будет содержать свойства целевого объекта ввода-вывода.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1, UMDF 2

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wdfkd.wdfwmi

Статья • 08.07.2024

Расширение **!wdfkd.wdfwmi** отображает сведения о инструментарии управления Microsoft Windows (WMI) для указанного объекта устройства платформы. Эти сведения включают все объекты поставщика WMI и связанные с ними объекты экземпляра WMI.

dbgcmd

```
!wdfkd.wdfwmi Handle
```

## Параметры

*Ручка*
Дескриптор объекта устройства платформы.

## DLL-библиотеки

Wdfkd.dll

### Платформы

KMDF 1

## Дополнительная информация

Дополнительные сведения см. в разделе "Отладка driver Framework в режиме ядра".

## Замечания

Выходные данные **расширения !wdfkd.wdfwmi** содержат сведения о регистрации WMI, поставщике и экземплярах.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# расширения платформы драйверов User-Mode (Wudfext.dll)

18.07.2025

Команды расширения, полезные для отладки драйверов Driver Framework User-Mode, реализуются в Wudfext.dll.

Некоторые расширения имеют дополнительные ограничения на требуемую версию Windows или версию UMDF; Эти ограничения отмечаются на отдельных справочных страницах.

**Note** When you create a new KMDF or UMDF driver, you must select a driver name that has 32 characters or less. Это ограничение длины определяется в wdfglobals.h. Если имя драйвера превышает максимальную длину, драйвер не будет загружаться.

Способы использования этих расширений см. в разделе User-Mode Отладка Driver Framework.

# !wudfext.help

Статья • 05.04.2024

Расширение **!wudfext.help** отображает все команды расширения Wudfext.dll.

dbgcmd

```
!wudfext.help
```

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.umdevstack

Статья • 05.04.2024

Расширение **!wudfext.umdevstack** отображает подробные сведения о стеке устройств в процессе узла.

```
dbgcmd
```

```
!wudfext.umdevstack DevstackAddress [Flags]
```

## Параметры

*DevstackAddress*
Указывает адрес стека устройств для отображения сведений.

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x01.

Bit 0 (0x01)
Отображает подробные сведения о стеке устройств.

Бит 8 (0x80)
Отображает сведения о внутренней платформе.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Ниже приведен пример **отображения !wudfext.umdevstack** :

```
dbgcmd
```

```
kd> !umdevstack 0x0034e4e0
Device Stack: 0x0034e4e0  Pdo Name: \Device\00000057
 Number of UM drivers: 0x1
  Driver 00:
    Driver Config Registry Path: WUDFEchoDriver
    UMDriver Image Path: C:\Windows\system32\DRIVERS\UMDF\WUDFEchoDriver.dll
    Fx Driver: IWDFDriver 0xf2db8
    Fx Device: IWDFDevice 0xf2f80
        IDriverEntry: WUDFEchoDriver!CMyDriver 0x000f2c70
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.umdevstacks

Статья • 05.04.2024

Расширение **!wudfext.umdevstacks** отображает сведения обо всех стеках устройств в текущем процессе узла.

dbgcmd

```
!wudfext.umdevstacks [Flags]
```

# Параметры

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x01.

Bit 0 (0x01)
Отображает подробные сведения о каждом стеке устройств.

Бит 8 (0x80)
Отображает сведения о внутренней платформе.

# DLL-библиотеки

Wudfext.dll

## Замечания

Расширение **!wudfext.umdevstacks отображает объекты интерфейса платформы** , связанные с каждым стеком устройств. Дополнительные сведения об использовании выходных данных из **!wudfext.umdevstacks** см. в разделе !wudfext.umdevstack.

Выходные **данные !wudfext.umdevstacks** включают два поля под названием "Отслеживание объектов" и "Отслеживание ссылок". Они указывают, включен ли параметр отслеживания объектов (TrackObjects) **и параметр отслеживания счетчика ссылок (TrackRefCounts**) в WDF Verifier соответственно. Если включен параметр отслеживания объектов, отображается адрес отслеживания объектов; этот адрес можно передать **в !wudfext.wudfdumpobjects** для отображения сведений об отслеживании.

Ниже приведен пример **отображения !wudfext.umdevstacks** :

dbgcmd

```
0: kd> !umdevstacks
Number of device stacks: 1
  Device Stack: 0x038c6f08    Pdo Name: \Device\USBPDO-11
    Number of UM devices: 1
    Device 0
      Driver Config Registry Path: WUDFOsrUsbFx2
      UMDriver Image Path:
D:\Windows\system32\DRIVERS\UMDF\WUDFOsrUsbFx2.dll
      Fx Driver: IWDFDriver 0x3076ff0
      Fx Device: IWDFDevice 0x3082e70
        IDriverEntry: WUDFOsrUsbFx2!CMyDriver 0x0306eff8
      Open UM files (use !umfile <addr> for details):
        0x04a8ef84
      Device XFerMode: CopyImmediately RW: Buffered CTL: Buffered
      Object Tracker Address: 0x03074fd8
        Object   Tracking ON
        Refcount Tracking OFF
    DevStack XFerMode: CopyImmediately RW: Buffered CTL: Buffered
```

# Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.umfile

Статья • 05.04.2024

Расширение **!wudfext.umfile** отображает сведения о файле внутри стека UMDF.

```
dbgcmd
```

```
!wudfext.umfile Address
```

## Параметры

*Адрес*
Указывает адрес файла UMDF внутри стека для отображения сведений о нем.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.umirp

Статья • 05.04.2024

Расширение **!wudfext.umirp** отображает сведения о пакете запросов ввода-вывода в режиме узла (UM IRP).

```
dbgcmd

!wudfext.umirp Address
```

## Параметры

*Адрес*
Указывает адрес IRP единой системы обмена сообщениями для отображения сведений.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Вы можете использовать **команду расширения !wudfext.umirps** , чтобы отобразить список всех невыполненных UM IRP в процессе узла.

Каждый IRP единой системы обмена сообщениями содержит одно или несколько расположений стека. Каждое расположение стека соответствует параметрам, которые один драйвер в стеке устройств будет получать при вызове запроса.

**!wudfext.umirp** дамп всех расположений стека и помечает текущее расположение с правой угловой скобкой (>). Текущее расположение соответствует драйверу, которому в настоящее время принадлежит запрос. Текущее расположение изменяется, когда драйвер перенаправит запрос на следующий нижний драйвер в стеке или когда драйвер завершает запрос, принадлежащий драйверу.

Ниже приведен пример **отображения !wudfext.umirp** :

dbgcmd

```
kd> !umirp 3dd480
UM IRP: 0x003dd480  UniqueId: 0xde  Kernel Irp: 0x0x85377850
  Type: WudfMsg_READ
  ClientProcessId: 0x338
  Device Stack: 0x0034e4e0
  IoStatus
    hrStatus: 0x0
    Information: 0x0
  Driver/Framework created IRP: No
  Data Buffer: 0x00000000 / 0
  IsFrom32BitProcess: Yes
  CancelFlagSet: No
  Cancel callback: 0x01102224
  Total number of stack locations: 2
  CurrentStackLocation: 2 (StackLocation[ 1 ])
    StackLocation[ 0 ]
      UNINITIALIZED
  > StackLocation[ 1 ]
      IWDFRequest:  ????
      IWDFDevice:   0x000f2f80
      IWDFFile:     0x003a7648
      Completion:
        Callback:   0x00000000
        Context:    0x00000000
      Parameters: (RequestType: WdfRequestRead)
        Buffer length:        0x400
        Key:                  0x00000000
        Offset:               0x0
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.umirps

Статья • 05.04.2024

Расширение **!wudfext.umirps** отображает список ожидающих пакетов запросов в режиме пользователя (UM IRPs) в процессе узла.

```
dbgcmd
```

```
!wudfext.umirps NumberOfIrps Flags
```

## Параметры

*NumberOfIrps*
Необязательно. Указывает количество ожидающих операций irps единой системы обмена сообщениями для отображения сведений о нем. Если *NumberOfIrps* является звездочкой (*) или опущен, будут отображаться все UM все UM IRPs.

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x01.

Bit 0 (0x01)
Отображает сведения о ожидающих irps.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Список ожидающих irps UM, отображаемых драйверу или ожидающих отправки драйверу.

По умолчанию **!wudfext.umirps** отображает все UM IRPs. Однако для ограничения этого дисплея можно использовать *параметр NumberOfIrps* .

Ниже приведен пример **отображения !wudfext.umirps** :

dbgcmd

```
kd> !umirps 0xa
Number of pending IRPS: 0xc8
####  CWudfIrp          Type        UniqueId          KernelIrp
----  ----------------  ----------  ----------------  ---------
0000            3dd280        READ                dc  856f02f0
0001            3dd380       WRITE                dd  85b869e0
0002            3dd480        READ                de  85377850
0003            3dd580        READ                df  93bba4e8
0004            3dd680       WRITE                e0  84cb9d70
0005            3dd780        READ                e1  85bec150
0006            3dd880       WRITE                e2  86651db0
0007            3dd980        READ                e3  85c22818
0008            3dda80        READ                e4  9961d150
0009            3ddb80       WRITE                e5  85c15148
```

Чтобы определить соответствующий IRP в режиме ядра, используйте **расширение !wudfext.wudfdownkmirp** . Кроме того, значения в **столбцах UniqueId** и **KernelIrp** можно использовать для сопоставления UMDF IRP (или UM IRP) с соответствующим IRP ядра. Значения в столбце **CWudfIrp можно передать в расширение !wudfext.umirp**, чтобы определить объекты платформы **IWDFRequest**, к которым может получить доступ каждый слой в стеке устройств.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfdevice

Статья • 05.04.2024

Расширение **!wudfext.wudfdevice** отображает самонастраивающийся (PnP) и системы состояния управления питанием для устройства.

```
dbgcmd

!wudfext.wudfdevice pWDFDevice
```

## Параметры

*pWDFDevice*
Указывает адрес **интерфейса IWDFDevice** для отображения состояния PnP или управления питанием.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Расширение !wudfext.wudfdevice можно использовать **для определения текущего состояния PnP или управления питанием устройства, указанного параметром *pWDFDevice***.

Ниже приведен пример **отображения !wudfext.wudfdevice** :

```
dbgcmd

kd> !wudfdevice 0xf2f80
Pnp Driver Callbacks:
  IPnpCallback: 0x0
  IPnpCallbackHardware: 0x0
  IPnpSelfManagedIo: 0x0
Pnp State Machine:
```

```
  CurrentState:  WdfDevStatePnpStarted
  Pending UMIrp: 0x00000000
    Could not read event queue depth, assuming 8
  Event queue:
    Processed/in-process events:
      PnpEventAddDevice
      PnpEventStartDevice
      PnpEventPwrPolStarted
    Pending events:
    State History:
      WdfDevStatePnpInit
      WdfDevStatePnpInitStarting
      WdfDevStatePnpHardwareAvailable
      WdfDevStatePnpEnableInterfaces
      WdfDevStatePnpStarted
Power State Machine:
  CurrentState:      WdfDevStatePowerD0
  Pending UMIrp:     0x00000000
  IoCallbackFailure: false
    Could not read event queue depth, assuming 8
  Event queue:
    Processed/in-process events:
      PowerImplicitD0
    Pending events:
    State History:
      WdfDevStatePowerStartingCheckDeviceType
      WdfDevStatePowerD0Starting
      WdfDevStatePowerD0StartingConnectInterrupt
      WdfDevStatePowerD0StartingDmaEnable
      WdfDevStatePowerD0StartingStartSelfManagedIo
      WdfDevStatePowerDecideD0State
      WdfDevStatePowerD0
Power Policy State Machine:
  CurrentState             : WdfDevStatePwrPolStartingSucceeded
  PowerPolicyOwner         : false
  PendingSystemPower UMIrp : 0x00000000
  PowerFailed              : false
    Could not read event queue depth, assuming 8
  Event queue:
    Processed/in-process events:
      PwrPolStart
      PwrPolPowerUp
    Pending events:
    State History:
      WdfDevStatePwrPolStarting
      WdfDevStatePwrPolStarted
      WdfDevStatePwrPolStartingSucceeded
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfdevicequeues

Статья • 05.04.2024

Расширение **!wudfext.wudfdevicequeues** отображает сведения обо всех очередях ввода-вывода для устройства.

```
dbgcmd
```

```
!wudfext.wudfdevicequeues pWDFDevice
```

## Параметры

*pWDFDevice*
Указывает адрес **интерфейса IWDFDevice** , для которого отображаются сведения обо всех связанных очередях ввода-вывода. Команда **расширения !wudfext.wudfdriverinfo** определяет адрес **IWDFDevice**.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Ниже приведен пример **отображения !wudfext.wudfdevicequeues** :

```
dbgcmd
```

```
## kd> !wudfdevicequeues 0xf2f80
-----------------------------------------------
Queue: 1 (!wudfqueue 0x000f3500)
    WdfIoQueueDispatchSequential, POWER MANAGED, WdfIoQueuePowerOn, CAN
ACCEPT, CAN DISPATCH
    Number of driver owned requests: 1
      IWDFIoRequest 0x000fa7c0     CWdfIoRequest 0x000fa748
    Number of waiting requests: 199
      IWDFIoRequest 0x000fa908     CWdfIoRequest 0x000fa890
```

```
      IWDFIoRequest 0x000faa50     CWdfIoRequest 0x000fa9d8
...
      IWDFIoRequest 0x000fa678     CWdfIoRequest 0x000fa600
    Driver's callback interfaces.
      IQueueCallbackRead 0x000f343c
      IQueueCallbackDeviceIoControl 0x000f3438
      IQueueCallbackWrite 0x000f3440
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfdownkmirp

Статья • 05.04.2024

Расширение **!wudfext.downkmmirp** отображает пакет запроса в режиме ядра (IRP), соответствующий указанному пакету запросов в режиме пользователя (UM IRP).

dbgcmd

```
!wudfext.wudfdownkmirp Address
```

## Параметры

*Адрес*
Указывает адрес UM IRP, соответствующий IRP в режиме ядра, который должен отображаться.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Вы можете использовать **команду расширения !wudfext.umirps** , чтобы отобразить список всех невыполненных UM IRP в процессе узла.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfdriverinfo

Статья • 05.04.2024

Расширение **!wudfext.wudfdriverinfo** отображает сведения о драйвере UMDF в текущем процессе узла.

dbgcmd

```
!wudfext.wudfdriverinfo Name
```

## Параметры

*Имя*
Указывает имя драйвера UMDF для отображения сведений.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Расширение **!wudfext.wudfdriverinfo** выполняет итерацию по каждому уровню в каждом стеке устройств и отображает сведения о драйвере и устройстве для каждой записи, которая соответствует драйверу, имя которого указано в параметре *Name* .

Вы можете использовать **!wudfext.wudfdriverinfo** для быстрого поиска объекта устройства для драйвера.

Ниже приведен пример **отображения !wudfext.wudfdriverinfo** :

dbgcmd

```
kd> !wudfdriverinfo wudfechodriver
IWDFDriver: 0xf2db8
```

```
!WDFDEVICE 0xf2f80
  !devstack 0x34e4e0 @ level 0
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfdumpobjects

Статья • 05.04.2024

Расширение **!wudfext.wudfdumpobjects** отображает выдающиеся объекты UMDF.

```
dbgcmd
```

```
!wudfext.wudfdumpobjects ObjTrackerAddress
```

## Параметры

*ObjTrackerAddress*
Указывает адрес для отслеживания утечки объектов. Этот адрес отображается в сообщении остановки драйвера в отладчике при возникновении утечки.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Если параметр отслеживания объектов UMDF (**TrackObjects**) включен в WDF Verifier, можно использовать **!wudfext.wudfdumpobjects** , чтобы увидеть все утечки объектов, оставшихся после выгрузки драйвера.

**Если параметр TrackObjects** включен, адрес отслеживания объектов автоматически отображается при обнаружении утечки. Используйте этот адрес как *ObjTrackerAddress* при выполнении **!wudfext.wudfdumpobjects**.

Это расширение можно использовать в любое время, даже если UMDF не сломался с отладчиком.

Если UMDF имеет версию 1.9 или более поздней, можно использовать **!wudfext.umdevstack или !wudfext.umdevstacks** для определения адреса

отслеживания объектов. Затем этот адрес можно передать **в** **!wudfext.wudfdumpobjects**. Рассмотрим пример:

dbgcmd

```
0: kd> !umdevstacks
Number of device stacks: 1
  Device Stack: 0x038c6f08    Pdo Name: \Device\USBPDO-11
    Number of UM devices: 1
    Device 0
      Driver Config Registry Path: WUDFOsrUsbFx2
      UMDriver Image Path:
D:\Windows\system32\DRIVERS\UMDF\WUDFOsrUsbFx2.dll
      Fx Driver: IWDFDriver 0x3076ff0
      Fx Device: IWDFDevice 0x3082e70
        IDriverEntry: WUDFOsrUsbFx2!CMyDriver 0x0306eff8
      Open UM files (use !umfile <addr> for details):
        0x04a8ef84
      Device XFerMode: CopyImmediately RW: Buffered CTL: Buffered
      Object Tracker Address: 0x03074fd8
        Object   Tracking ON
        Refcount Tracking OFF
    DevStack XFerMode: CopyImmediately RW: Buffered CTL: Buffered

0: kd> !wudfdumpobjects 0x03074fd8
WdfTypeDriver    Object: 0x03076fb0, Interface: 0x03076ff0
WdfTypeDevice   Object: 0x03082e30, Interface: 0x03082e70
WdfTypeIoTarget Object: 0x03088f50, Interface: 0x03088f90
WdfTypeIoQueue                Object: 0x0308ce58, Interface: 0x0308ce98
WdfTypeIoQueue                Object: 0x03090e58, Interface: 0x03090e98
WdfTypeIoQueue                Object: 0x03092e58, Interface: 0x03092e98
WdfTypeIoTarget Object: 0x03098f40, Interface: 0x03098f80
WdfTypeFile         Object: 0x0309cfa0, Interface: 0x0309cfe0
WdfTypeUsbInterface         Object: 0x030a0f98, Interface: 0x030a0fd8
WdfTypeRequest Object: 0x030a2ef8, Interface: 0x030a2f38
WdfTypeIoTarget Object: 0x030a6f30, Interface: 0x030a6f70
WdfTypeIoTarget Object: 0x030aaf30, Interface: 0x030aaf70
WdfTypeIoTarget Object: 0x030aef30, Interface: 0x030aef70
WdfTypeRequest Object: 0x030c6ef8, Interface: 0x030c6f38
WdfTypeRequest Object: 0x030ceef8, Interface: 0x030cef38
WdfTypeMemoryObject    Object: 0x030d6fb0, Interface: 0x030d6ff0
WdfTypeMemoryObject    Object: 0x030dcfb0, Interface: 0x030dcff0
WdfTypeFile         Object: 0x030e4fa8, Interface: 0x030e4fe8
WdfTypeFile         Object: 0x030e6fa8, Interface: 0x030e6fe8
WdfTypeFile         Object: 0x030e8fa8, Interface: 0x030e8fe8
WdfTypeRequest Object: 0x030eaef8, Interface: 0x030eaf38
WdfTypeMemoryObject    Object: 0x030ecfb0, Interface: 0x030ecff0
WdfTypeMemoryObject    Object: 0x030eefb0, Interface: 0x030eeff0
```

# Обратная связь

Были ли сведения на этой странице полезными? Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudffile

Статья • 05.04.2024

Расширение **!wudfext.wudffile** отображает сведения о файле платформы.

```
dbgcmd
```

```
!wudfext.wudffile pWDFFile [TypeName]
```

## Параметры

*pWDFFile*
Указывает адрес интерфейса IWDFFile **для отображения сведений** о нем.

*Typename*
Необязательно. Указывает тип интерфейса (например, **IWDFDevice**). Если задано значение *TypeName* , расширение использует значение в качестве типа интерфейса. Если звездочка (*) предоставляется как *TypeName* или *если typeName* опущен, расширение пытается автоматически определить тип предоставленного интерфейса.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudffilehandletarget

Статья • 05.04.2024

Расширение **!wudfext.wudffilehandletarget** отображает сведения о целевом объекте ввода-вывода на основе файлов.

dbgcmd

```
!wudfext.wudffilehandletarget pWDFFileHandleTarget TypeName
```

## Параметры

*pWDFFileHandleTarget*
Указывает адрес интерфейса IWDFIoTarget **для отображения сведений** о нем. Команда **расширения !wudfext.wudfobject** определяет адрес **IWDFIoTarget**.

*Typename*
Необязательно. Указывает тип интерфейса (например, **IWDFDevice**). Если задано значение *TypeName* , расширение использует значение в качестве типа интерфейса. Если звездочка (*) предоставляется как *TypeName* или *если typeName* опущен, расширение пытается автоматически определить тип предоставленного интерфейса.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfiotarget

Статья • 05.04.2024

Расширение **!wudfext.wudfiotarget** отображает сведения о целевом объекте ввода-вывода, включая состояние целевого объекта и список отправленных запросов.

```
dbgcmd

!wudfext.wudfiotarget pWDFTarget TypeName
```

## Параметры

*pWDFTarget*
Указывает адрес интерфейса IWDFIoTarget **для отображения сведений** о нем. Команда **расширения !wudfext.wudfobject** определяет адрес **IWDFIoTarget**.

*Typename*
Необязательно. Указывает тип интерфейса (например, **IWDFDevice**). Если задано значение *TypeName* , расширение использует значение в качестве типа интерфейса. Если звездочка (*) предоставляется как *TypeName* или *если typeName* опущен, расширение пытается автоматически определить тип предоставленного интерфейса.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfobject

Статья • 05.04.2024

Расширение **!wudfext.wudfobject** отображает сведения о объекте WDF, а также о его родительских и дочерних отношениях.

dbgcmd

```
!wudfext.wudfobject pWDFObject Flags TypeName
```

# Параметры

*pWDFObject*
Указывает адрес интерфейса WDF для отображения сведений.

*Flags*
Необязательно. Указывает тип отображаемой информации. *Флаги* могут быть любым сочетанием следующих битов. Значение по умолчанию — 0x01.

Bit 0 (0x01)
Шаги рекурсивно через иерархию объектов для получения родительских и дочерних связей, которые отображаются.

Бит 1 (0x02)
Отображает только сводные сведения об объекте.

Бит 8 (0x80)
Шаги рекурсивно через иерархию объектов и отображают сведения о внутренней платформе.

*Typename*
Необязательно. Указывает тип интерфейса (например, **IWDFDevice**). Если задано значение *TypeName* , расширение использует значение в качестве типа интерфейса. Если звездочка (*) предоставляется как *TypeName* или *если typeName* опущен, расширение пытается автоматически определить тип предоставленного интерфейса.

# DLL-библиотеки

Wudfext.dll

# Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

# Замечания

Вы можете использовать **!wudfext.wudfobject** для перечисления, например дочерних **объектов объекта IWDFDevice** , которые обычно включают очереди устройства.

Вы также можете использовать **!wudfext.wudfobject** для поиска объектов WDF, связанных с определенным устройством, для проверка состояния объекта WDF (например, находится ли объект WDF в процессе удаления), или определить текущее число ссылок объекта WDF.

Расширение **!wudfext.wudfobject** также отображает функции обратного вызова и объекты контекста, связанные с каждым объектом платформы, и пытается определить тип объекта платформы. Эта последняя функция может не работать с определенными компиляторами.

Ниже приведены некоторые примеры. В первом примере !wudfext.umdevstacks **предоставляет 0x03050E70 в качестве адреса объекта устройства, а затем этот адрес передается в !wudfext.wudfobject.** Флаг 0x1 включается для отображения всех дочерних элементов этого объекта.

dbgcmd

```
0: kd> !umdevstacks
Number of device stacks: 1
  Device Stack: 0x038f6f08    Pdo Name: \Device\USBPDO-11
    Number of UM devices: 1
    Device 0
      Driver Config Registry Path: WUDFOsrUsbFx2
      UMDriver Image Path:
D:\Windows\system32\DRIVERS\UMDF\WUDFOsrUsbFx2.dll
      Fx Driver: IWDFDriver 0x3044ff0
      Fx Device: IWDFDevice 0x3050e70
        IDriverEntry: WUDFOsrUsbFx2!CMyDriver 0x0303eff8
      Open UM files (use !umfile <addr> for details):
        0x049baf84
      Device XFerMode: CopyImmediately RW: Buffered CTL: Buffered
      Object Tracker Address: 0x00000000
        Object   Tracking OFF
        Refcount Tracking OFF
    DevStack XFerMode: CopyImmediately RW: Buffered CTL: Buffered
```

```
0: kd> !wudfobject 0x3050e70 1
IWDFDevice 0x3050e70 Fx: 0x3050e30 [Ref 2]
  15 Children
    00: IWDFIoTarget 0x3056f90 Fx: 0x3056f50 [Ref 3]
        No Children
    01: <Internal>
    02: <Internal>
    03: <Internal>
    04: IWDFIoQueue 0x305ae98 Fx: 0x305ae58 [Ref 8]
        No Children
    05: IWDFIoQueue 0x305ee98 Fx: 0x305ee58 [Ref 2]
        No Children
    06: IWDFIoQueue 0x3060e98 Fx: 0x3060e58 [Ref 2]
        No Children
    07: IWDFIoTarget 0x3066f80 Fx: 0x3066f40 [Ref 2]
        1 Children
          00: IWDFUsbInterface 0x306efd8 Fx: 0x306ef98 [Ref 1]
              3 Children
                00: IWDFIoTarget 0x3074f70 Fx: 0x3074f30 [Ref 2]
                    2 Children
                      00: IWDFMemory 0x30a4ff0 Fx: 0x30a4fb0 [Ref 2]
                          No Children
                      01: IWDFMemory 0x30aaff0 Fx: 0x30aafb0 [Ref 2]
                          No Children
                01: IWDFIoTarget 0x3078f70 Fx: 0x3078f30 [Ref 1]
                    No Children
```

Ниже приведен пример **!wudfext.wudfobject** , отображающий объект файла:

dbgcmd

```
kd> !wudfobject 0xf5060
IWDFFile 0xf5060 Fx: 0xf4fe8 [Ref 1]
  State: Created   Parent: 0xf2f80
  No Children
```

Ниже приведен пример **!wudfext.wudfobject** , отображающий объект драйвера:

dbgcmd

```
kd> !wudfobject 0xf2db8 0x01
IWDFDriver 0xf2db8 Fx: 0xf2d40 [Ref 2]
  Callback: (WUDFEchoDriver!CMyDriver, 0xf2c68)
  State: Created   Parent: 0
  1 Children:
    00: IWDFDevice 0xf2f80 Fx: 0xf2f08 [Ref 2]
        State: Created   Parent: 0xf2db8
        5 Children:
          00: IWDFIoTarget 0xf33c0 Fx: 0xf3348 [Ref 3]
              State: Created   Parent: 0xf2f80
              No Children
          01: IWDFIoQueue 0xf3500 Fx: 0xf3488 [Ref 3]
```

```
        State: Created   Parent: 0xf2f80
        No Children
    02: IWDFFile 0xf5060 Fx: 0xf4fe8 [Ref 1]
        State: Created   Parent: 0xf2f80
        No Children
    03: IWDFFile 0xf5100 Fx: 0xf5088 [Ref 101]
        State: Created   Parent: 0xf2f80
        No Children
    04: IWDFFile 0xf51a0 Fx: 0xf5128 [Ref 101]
        State: Created   Parent: 0xf2f80
        No Children
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfqueue

Статья • 05.04.2024

Расширение **!wudfext.wudfqueue** отображает сведения о очереди ввода-вывода.

dbgcmd

```
!wudfext.wudfqueue pWDFQueue
```

## Параметры

*pWDFQueue*
Указывает адрес интерфейса IWDFIoQueue **для отображения сведений** о нем. **Команда расширения !wudfext.wudfdevicequeues** определяет адрес **IWDFIoQueue**.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Ниже приведен пример **отображения !wudfext.wudfqueue** :

dbgcmd

```
kd> !wudfqueue 0x000f3500
    WdfIoQueueDispatchSequential, POWER MANAGED, WdfIoQueuePowerOn, CAN
ACCEPT, CAN DISPATCH
    Number of driver owned requests: 1
      IWDFIoRequest 0x000fa7c0     CWdfIoRequest 0x000fa748
    Number of waiting requests: 199
      IWDFIoRequest 0x000fa908     CWdfIoRequest 0x000fa890
      IWDFIoRequest 0x000faa50     CWdfIoRequest 0x000fa9d8
      IWDFIoRequest 0x000fab98     CWdfIoRequest 0x000fab20
      ...
      IWDFIoRequest 0x000fa530     CWdfIoRequest 0x000fa4b8
      IWDFIoRequest 0x000fa678     CWdfIoRequest 0x000fa600
```

```
    Driver's callback interfaces.
      IQueueCallbackRead 0x000f343c
      IQueueCallbackDeviceIoControl 0x000f3438
      IQueueCallbackWrite 0x000f3440
```

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfrefhist

Статья • 05.04.2024

Расширение **!wudfext.wudfrefhist** отображает журнал стека ссылок для объекта UMDF.

```
dbgcmd

!wudfext.wudfrefhist ObjectAddress
```

## Параметры

*ObjectAddress*
Указывает адрес объекта UMDF, журнал стека ссылок которого должен отображаться. Обратите внимание, что это адрес самого объекта, а не интерфейса.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

**Команда !wudfext.wudfrefhist** не поддерживается UMDF 1.11.

*Параметр ObjectAddress* должен быть адресом объекта UMDF, а не адресом интерфейса (который используется многими другими командами расширения UMDF). Чтобы определить адрес объекта UMDF, используйте **команду !wudfext.wudfdumpobjects** , которая отображает как адрес объекта UMDF, так и адрес интерфейса. Кроме того, если вы знаете адрес интерфейса, его можно использовать в качестве аргумента **команды !wudfext.wudfobject** , которая отображает адрес объекта (отображается после символа "Fx:").

Если параметр отслеживания счетчика ссылок (**TrackRefCounts**) включен в средстве проверки WDF, можно использовать **!wudfext.wudfrefhist** для отображения

каждого стека вызовов, который увеличивает или уменьшает число ссылок объекта. Вы можете определить, вызывает ли стек вызовов памяти утечку памяти, проверив его **вызовы AddRef** и **Release** для добавленных и не выпущенных ссылок.

Эта команда может использоваться в любое время, даже если UMDF не вломалась в отладчик.

Если эта команда не отображает нужные сведения, убедитесь, что соответствующие данные отображаются и повторите попытку.

Ниже приведен пример использования **команды !wudfext.wudfrefhist** :

dbgcmd

```
0:007> !wudfext.umdevstacks
...
      UMDriver Image Path:
C:\Windows\System32\drivers\UMDF\WUDFOsrUsbFilter.dll
      Object Tracker Address: 0x0000003792ee9fc0
        Object   Tracking ON
        Refcount Tracking ON

0:007> !wudfext.wudfdumpobjects 0x0000003792ee9fc0
...
WdfTypeIoQueue   Object: 0x0000003792f05ce0, Interface: 0x0000003792f05d58

## 0:007> !wudfext.wudfrefhist 0x0000003792f05ce0
-------------------------------------------------------------------------
## 2  (++)
-------------------------------------------------------------------------
WUDFx!UfxObject::TrackRefCounts+0xb2
WUDFx!CWdfIoQueue::AddRef+0x17adc
WUDFx!CWdfIoQueue::CreateAndInitialize+0xeb
WUDFx!CWdfDevice::CreateIoQueue+0x1e7
WUDFOsrUsbFilter!CMyQueue::Initialize+0x48
WUDFOsrUsbFilter!CMyDevice::Configure+0x7d
WUDFOsrUsbFilter!CMyDriver::OnDeviceAdd+0xca
WUDFx!CWdfDriver::OnAddDevice+0x486
WUDFx!CWUDF::AddDevice+0x43
WUDFHost!CWudfDeviceStack::LoadDrivers+0x320
WUDFHost!CLpcNotification::Message+0x1340
WUDFPlatform!WdfLpcPort::ProcessMessage+0x140
WUDFPlatform!WdfLpcCommPort::ProcessMessage+0x92
WUDFPlatform!WdfLpc::RetrieveMessage+0x20c
```

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfrequest

Статья • 05.04.2024

Расширение **!wudfext.wudfrequest** отображает сведения о запросе ввода-вывода.

dbgcmd

```
!wudfext.wudfrequest pWDFRequest
```

## Параметры

*pWDFRequest*
Указывает адрес интерфейса WDFIoRequest **для отображения сведений** о нем. Команда **расширения !wudfext.wudfqueue** определяет адрес **WDFIoRequest**.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Замечания

Ниже приведен пример **отображения !wudfext.wudfrequest** :

dbgcmd

```
kd> !wudfrequest 0x000fa530
CWdfIoRequest  0x000fa4b8
Type: WdfRequestRead
  IWDFIoQueue: 0x000f3500
Completed: No
Canceled: No
UM IRP: 0x00429108  UniqueId: 0xf4  Kernel Irp: 0x0x936ef160
  Type: WudfMsg_READ
  ClientProcessId: 0x1248
  Device Stack: 0x003be4e0
  IoStatus
    hrStatus: 0x0
```

```
    Information: 0x0
  Driver/Framework created IRP: No
  Data Buffer: 0x00000000 / 0
  IsFrom32BitProcess: Yes
  CancelFlagSet: No
  Cancel callback: 0x000fa534
  Total number of stack locations: 2
  CurrentStackLocation: 2 (StackLocation[ 1 ])
    StackLocation[ 0 ]
      UNINITIALIZED
  > StackLocation[ 1 ]
      IWDFRequest:  ????
      IWDFDevice:   0x000f2f80
      IWDFFile:     0x00418cf0
      Completion:
        Callback:   0x00000000
        Context:    0x00000000
      Parameters: (RequestType: WdfRequestRead)
        Buffer length:        0x400
        Key:                  0x00000000
        Offset:               0x0
```

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfusbinterface

Статья • 05.04.2024

Расширение **!wudfext.wudfusbinterface** отображает сведения об объекте USB-интерфейса.

```
dbgcmd
```

```
!wudfext.wudfusbinterface pWDFUSBInterface TypeName
```

## Параметры

*pWDFUSBInterface*
Указывает адрес интерфейса IWDFUsbInterface **для отображения сведений**. Команда **расширения !wudfext.wudfobject** определяет адрес **IWDFUsbInterface**.

*Typename*
Необязательно. Указывает тип интерфейса (например, **IWDFDevice**). Если задано значение *TypeName* , расширение использует значение в качестве типа интерфейса. Если звездочка (*) предоставляется как *TypeName* или *если typeName* опущен, расширение пытается автоматически определить тип предоставленного интерфейса.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfusbpipe

Статья • 08.07.2024

Расширение **!wudfext.wudfusbpipe** отображает сведения об объекте USB-канала.

```
dbgcmd
```

```
!wudfext.wudfusbpipe pWDFUSBPipe TypeName
```

## Параметры

*pWDFUSBPipe*
Указывает адрес **интерфейса IWDFUsbTargetPipe** для отображения сведений. Команда **расширения !wudfext.wudfobject** определяет адрес **IWDFUsbTargetPipe**.

*TypeName*
Необязательно. Указывает тип интерфейса (например, **IWDFDevice**). Если задано значение *TypeName* , расширение использует значение в качестве типа интерфейса. Если звездочка (*) предоставляется как *TypeName* или *если typeName* опущен, расширение пытается автоматически определить тип предоставленного интерфейса.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wudfext.wudfusbtarget

Статья • 08.07.2024

Расширение **!wudfext.wudfusbtarget** отображает сведения о целевом объекте ввода-вывода USB.

dbgcmd

```
!wudfext.wudfusbtarget pWDFUSBTarget TypeName
```

## Параметры

*pWDFUSBTarget*
Указывает адрес интерфейса IWDFUsbTargetDevice **для отображения сведений**. **Команда расширения !wudfext.wudfobject** определяет адрес **IWDFUsbTargetDevice**.

*TypeName*
Необязательно. Указывает тип интерфейса (например, **IWDFDevice**). Если задано значение *TypeName* , расширение использует значение в качестве типа интерфейса. Если звездочка (*) предоставляется как *TypeName* или *если typeName* опущен, расширение пытается автоматически определить тип предоставленного интерфейса.

## DLL-библиотеки

Wudfext.dll

## Дополнительная информация

Дополнительные сведения см. в разделе среда выполнения платформы драйвера режима пользователя Отладка.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# Расширения трассировки WMI (Wmitrace.dll)

Статья • 26.10.2023

Команды расширения для сеансов трассировки программного обеспечения можно найти в Wmitrace.dll, библиотеке функций, предназначенной для использования инструментария управления Windows (WMI) для трассировки событий.

# !wmitrace.disable

Статья • 05.04.2024

Расширение **!wmitrace.disable отключает** поставщика для указанного сеанса трассировки событий для Windows (ETW).

dbgcmd

```
!wmitrace.disable { LoggerID | LoggerName } GUID
```

## Параметры

*LoggerID*
Указывает сеанс трассировки. *LoggerID* — это порядковый номер, который система назначает каждому сеансу трассировки на компьютере.

*Имя средства ведения журнала*
Указывает сеанс трассировки. *LoggerName* — это текстовое имя, указанное при запуске сеанса трассировки.

*GUID*
Указывает GUID поставщика, который необходимо отключить.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 7 и более поздних версиях Windows.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о средствах трассировки см. в комплекте драйверов Windows (WDK).

## Замечания

После использования этого расширения необходимо возобновить выполнение программы (например, с помощью **команды g (Go),** чтобы она ввела силу. Через некоторое время целевой компьютер автоматически переходит в отладчик.

Чтобы включить поставщик, используйте **!wmitrace.enable**.

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.dumpmini

Статья • 05.04.2024

Расширение **!wmitrace.dumpmini** отображает фрагмент трассировки системы, который хранится в файле дампа.

dbgcmd

```
!wmitrace.dumpmini
```

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows Vista и более поздних версиях Windows.

Это расширение полезно только при отладке мини-файла или полного файла дампа.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о средствах трассировки см. в комплекте драйверов Windows (WDK).

## Замечания

Фрагмент *трассировки* системы — это копия содержимого последнего буфера журнала контекста системы. В обычных условиях это сеанс трассировки, идентификатор средства ведения журнала которого равен 2.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.dumpminievent

Статья • 05.04.2024

Расширение **!wmitrace.dumpminievent** отображает фрагмент трассировки журнала событий системы, который хранится в файле дампа.

dbgcmd

```
!wmitrace.dumpminievent
```

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows Vista с пакетом обновления 1 (SP1) и более поздних версиях Windows.

Это расширение полезно только при отладке мини-файла или полного файла дампа.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о средствах трассировки см. в комплекте драйверов Windows (WDK).

## Замечания

Фрагмент *трассировки* журнала событий системы — это копия содержимого последнего буфера журнала событий системы. Расширение !wmitrace.dumpminievent **отображает его содержимое** в формате журнала событий.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.dynamicprint

Статья • 05.04.2024

Расширение **!wmitrace.dynamicprint** определяет, отображает ли отладчик сообщения трассировки, созданные сеансом, запущенным в KD_FILTER_MODE.

dbgcmd

```
!wmitrace.dynamicprint {0 | 1}
```

## Параметры

**0**
Включает отображение сообщения трассировки.

**1**
Включает отображение сообщения трассировки.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 2000 и более поздних версиях Windows. Если вы хотите использовать это расширение с Windows 2000, сначала скопируйте файл Wmitrace.dll из подкаталога winxp каталога средств отладки для Windows в подкаталог w2kfre.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о запуске сеанса трассировки см. в разделе "Tracelog" в комплекте драйверов Windows (WDK).

## Замечания

Перед использованием этого расширения запустите сеанс трассировки и укажите, что сообщения трассировки должны отправляться отладчику. Например, если вы используете **!wmitrace.start** для запуска сеанса, используйте **параметр -kd** . Если для запуска сеанса трассировки используется tracelog, используйте его **параметр -**

**kd** . Tracelog (tracelog.exe) — это контроллер трассировки, включенный в комплект драйверов Windows.

Сообщения трассировки хранятся в буферах на целевом компьютере. Эти буферы очищаются и отправляются отладчику на хост-компьютере через регулярные интервалы. Интервал таймера очистки можно указать с помощью **параметра -kd команды !wmitrace.start или параметра -kd** средства Tracelog. Начиная с Windows 8, можно указать значение таймера очистки в миллисекундах, добавив **ms** к значению таймера очистки.

По умолчанию ETW поддерживает буферы трассировки на каждом процессоре на целевом компьютере. При очистке буферов трассировки и отправке отладчику на хост-компьютере нет механизма объединения буферов в хронологическую последовательность событий. Таким образом, события могут отображаться вне порядка. Начиная с Windows 7, эту проблему можно решить, задав **параметр -lowcapacity** при использовании средства tracelog для запуска сеанса трассировки.

**Tracelog***MySession-kd* **-lowcapacity**

При запуске сеанса с **набором -lowcapacity** все события переходят к одному буферу на целевом компьютере, а события отображаются в правильном порядке в отладчике на хост-компьютере.

Кроме того, перед использованием этого расширения используйте **!wmitrace.searchpath** или **!wmitrace.tmffile** , чтобы указать файлы формата сообщений трассировки. Система использует файлы формата сообщений трассировки для форматирования двоичных сообщений трассировки, чтобы они могли отображаться как удобочитаемый человеком текст.

## См. также

!wmitrace.start

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.enable

Статья • 08.07.2024

Расширение **!wmitrace.enable** позволяет поставщику для указанного сеанса трассировки событий windows (ETW).

dbgcmd

```
!wmitrace.enable { LoggerID | LoggerName } GUID [-level Num] [-matchallkw
Num] [-matchanykw Num] [-enableproperty Num] [-flag Num]
```

# Параметры

*LoggerID*
Указывает сеанс трассировки. *LoggerID* — это порядковый номер, который система назначает каждому сеансу трассировки на компьютере.

*Имя средства ведения журнала*
Указывает сеанс трассировки. *LoggerName* — это текстовое имя, указанное при запуске сеанса трассировки.

*GUID*
Указывает GUID поставщика, который необходимо включить.

**-уровень** *Число*
Задает уровень. *Число* может быть любым целым числом.

**-matchallkw** *Число*
Задает одно или несколько ключевых слов. Если задано несколько ключевых слов, поставщик будет включен только в том случае, если все ключевые слова совпадают. *Число* может быть любым целым числом.

**-matchanykw** *Число*
Задает одно или несколько ключевых слов. Если задано несколько ключевых слов, поставщик будет включен, если совпадает по крайней мере одно ключевое слово. *Число* может быть любым целым числом. Эффекты этого параметра перекрываются с эффектами параметра -flag.

**-enableproperty** *Число*
Указывает свойство enable. *Число* может быть любым целым числом.

**-флаг** *Число*
Указывает один или несколько флагов. *Число* может быть любым целым числом. Эффекты этого параметра перекрываются с эффектами параметра -matchanykw.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 7 и более поздних версиях Windows.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о средствах трассировки см. в комплекте драйверов Windows (WDK).

## Замечания

После использования этого расширения необходимо возобновить выполнение программы (например, с помощью **команды g (Go),** чтобы она ввела силу. Через некоторое время целевой компьютер автоматически переходит в отладчик.

Чтобы отключить поставщика, используйте **!wmitrace.disable**.

---


## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.eventlogdump

Статья • 05.04.2024

Расширение **!wmitrace.eventlogdump** отображает содержимое указанного средства ведения журнала. Отображение отформатировано как журнал событий.

```
dbgcmd
```

```
!wmitrace.eventlogdump { LoggerID | LoggerName }
```

## Параметры

*LoggerID*
Указывает сеанс трассировки. *LoggerID* — это порядковый номер, который система назначает каждому сеансу трассировки на компьютере.

*Имя средства ведения журнала*
Указывает сеанс трассировки. *LoggerName* — это текстовое имя, указанное при запуске сеанса трассировки.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 2000 и более поздних версиях Windows. Если вы хотите использовать это расширение с Windows 2000, сначала скопируйте файл Wmitrace.dll из подкаталога winxp каталога средств отладки для Windows в подкаталог w2kfre.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о средствах трассировки см. в комплекте драйверов Windows (WDK).

## Замечания

Это расширение аналогично **расширению !wmitrace.logdump**, за исключением того, что выходные данные !wmitrace.eventlogdump форматируются в стиле журнала событий, а выходные данные **!wmitrace.logdump** форматируются в стиле

препроцессора трассировки программного обеспечения Windows (WPP). Следует выбрать расширение, формат которого подходит для отображаемых данных.

Чтобы найти идентификатор средства ведения журнала сеанса трассировки, используйте **расширение !wmitrace.strdump** . Кроме того, можно использовать командный журнал трассировки tracelog -l для перечисления сеансов трассировки и их основных свойств, включая идентификатор средства ведения журнала.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.help

Статья • 05.04.2024

Расширение **!wmitrace.help** отображает команды расширения в Wmitrace.dll.

dbgcmd

```
!wmitrace.help
```

## DLL-библиотеки

Wmitrace.dll

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о средствах трассировки см. в комплекте драйверов Windows (WDK).

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.logdump

Статья • 05.04.2024

Расширение **!wmitrace.logdump** отображает содержимое буферов трассировки для сеанса трассировки. Вы можете ограничить отображение трассировки сообщений от указанных поставщиков.

dbgcmd

```
!wmitrace.logdump [-t Count] [{LoggerID|LoggerName} [GUIDFile]]
```

## Параметры

**-t***Count*
Ограничивает выходные данные последними сообщениями. *Число* указывает количество отображаемых сообщений.

*LoggerID*
Указывает сеанс трассировки. *LoggerID* — это порядковый номер, который система назначает каждому сеансу трассировки на компьютере. Если параметр не указан, используется сеанс трассировки с идентификатором 1.

*Имя средства ведения журнала*
Указывает сеанс трассировки. *LoggerName* — это текстовое имя, указанное при запуске сеанса трассировки.

*GUIDFile*
Отображает только сообщения трассировки от поставщиков, указанных *в файле GUIDFile* . *GUIDFile* представляет путь (необязательно) и имя файла текстового файла, содержащего идентификаторы GUID элемента управления одного или нескольких поставщиков трассировки, например GUID или CTL-файл.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 2000 и более поздних версиях Windows. Если вы хотите использовать это расширение с Windows 2000, сначала скопируйте файл Wmitrace.dll из подкаталога winxp каталога средств отладки для Windows в подкаталог w2kfre.

# Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о tracelog см. в разделе "Tracelog" в комплекте драйверов Windows (WDK).

# Замечания

Во время трассировки программного обеспечения трассировки программного обеспечения (WPP) программы трассировки буферы сеансов трассировки используются для хранения сообщений трассировки до тех пор, пока они не удаляются в файл журнала или для потребителя трассировки для отображения в режиме реального времени. Расширение **!wmitrace.logdump** отображает содержимое буферов, которые находятся в физической памяти. Отображение отображается в окне команды отладчика.

Это расширение особенно полезно для восстановления последних трассировок при сбое и отображения трассировок, хранящихся в файле аварийного дампа.

Перед использованием этого расширения используйте **!wmitrace.searchpath** или **!wmitrace.tmffile** , чтобы указать файлы формата сообщений трассировки. Система использует файлы форматирования сообщений трассировки для форматирования двоичных сообщений трассировки в буферах, чтобы они могли отображаться как удобочитаемый человеком текст.

**Примечание.** Если драйвер использует UMDF версии 1.11 или более поздней версии, вам не нужно использовать **!wmitrace.searchpath** или **!wmitrace.tmffile.**

При использовании Tracelog для запуска сеанса трассировки с циклическим буферизацией (буферизация) используйте это расширение для отображения содержимого буфера.

Чтобы найти идентификатор средства ведения журнала сеанса трассировки, используйте **расширение !wmitrace.strdump** . Кроме того, можно использовать командный журнал трассировки tracelog -l для перечисления сеансов трассировки и их основных свойств, включая идентификатор средства ведения журнала.

Это расширение полезно только во время трассировки программного обеспечения WPP и более ранних (устаревших) методов трассировки событий для Windows. События трассировки, созданные другими манифестными поставщиками, не используют файлы формата сообщений трассировки (TMF), поэтому это расширение не отображает их содержимое.

Это расширение аналогично **расширению !wmitrace.eventlogdump**, за исключением того, что выходные данные !wmitrace.logdump форматируются в стиле WPP, а выходные данные **!wmitrace.eventlogdump** форматируются в стиле журнала событий. Следует выбрать расширение, формат которого подходит для отображаемых данных.

Сведения о просмотре журнала трассировки UMDF см. в статье Об использовании трассировки программного обеспечения WPP в драйверах на основе UMDF.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.logger

Статья • 05.04.2024

Расширение **!wmitrace.logger** отображает данные о сеансе трассировки, включая данные конфигурации сеанса. Это расширение не отображает сообщения трассировки, созданные во время сеанса.

dbgcmd

```
!wmitrace.logger [ LoggerID | LoggerName ]
```

## Параметры

*LoggerID*
Указывает сеанс трассировки. *LoggerID* — это порядковый номер, который система назначает каждому сеансу трассировки на компьютере. Если параметр не указан, используется сеанс трассировки с идентификатором 1.

*Имя средства ведения журнала*
Указывает сеанс трассировки. *LoggerName* — это текстовое имя, указанное при запуске сеанса трассировки.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 2000 и более поздних версиях Windows. Если вы хотите использовать это расширение с Windows 2000, сначала скопируйте файл Wmitrace.dll из подкаталога winxp каталога средств отладки для Windows в подкаталог w2kfre.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows.

## Замечания

Это расширение предназначено для журналов производительности и событий, которые нельзя отформатировать для отображения, доступного для чтения. Чтобы

отобразить сообщения трассировки в буфере сеансов трассировки, а также данные заголовка, используйте **!wmitrace.logdump**.

Чтобы найти идентификатор средства ведения журнала сеанса трассировки, используйте **расширение !wmitrace.strdump** . Кроме того, можно использовать командный журнал трассировки tracelog -l для перечисления сеансов трассировки и их основных свойств, включая идентификатор средства ведения журнала.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.logsave

Статья • 05.04.2024

Расширение **!wmitrace.logsave** записывает текущее содержимое буферов трассировки для сеанса трассировки в файл.

```
dbgcmd
```

```
!wmitrace.logsave {LoggerID|LoggerName} Filename
```

## Параметры

*LoggerID*
Указывает сеанс трассировки. *LoggerID* — это порядковый номер, который система назначает каждому сеансу трассировки на компьютере.

*Имя средства ведения журнала*
Указывает сеанс трассировки. *LoggerName* — это текстовое имя, указанное при запуске сеанса трассировки.

*Имя файла*
Указывает путь (необязательно) и имя файла для выходного файла.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 2000 и более поздних версиях Windows. Если вы хотите использовать это расширение с Windows 2000, сначала скопируйте файл Wmitrace.dll из подкаталога winxp каталога средств отладки для Windows в подкаталог w2kfre.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о tracelog см. в разделе "Tracelog" в комплекте драйверов Windows (WDK).

## Замечания

Это расширение отображает только трассировки, которые находятся в памяти в то время. Он не отображает сообщения трассировки, которые были удалены из буферов и доставлены в файл журнала трассировки событий или потребителю трассировки.

Буферы сеансов трассировки хранят сообщения трассировки, пока они не удаляются в файл журнала или в потребитель трассировки для отображения в режиме реального времени. Это расширение сохраняет содержимое буферов, которые находятся в физической памяти в указанном файле.

Выходные данные записываются в двоичном формате. Как правило, эти файлы используют расширение имени файла ETL (журнал трассировки событий).

При использовании tracelog для запуска сеанса трассировки с циклической буферизацией (буферизация) можно использовать это расширение для сохранения текущего содержимого буфера.

Чтобы найти идентификатор средства ведения журнала сеанса трассировки, используйте **расширение !wmitrace.strdump** . Кроме того, можно использовать командный журнал трассировки tracelog -l для перечисления сеансов трассировки и их основных свойств, включая идентификатор средства ведения журнала.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.searchpath

Статья • 05.04.2024

Расширение **!wmitrace.searchpath** указывает расположение файлов формата сообщений трассировки для сообщений в буферах трассировки.

dbgcmd

```
!wmitrace.searchpath [+] TMFPath
!wmitrace.searchpath
```

## Параметры

+
Приводит *к добавлению TMFPath* к существующему пути поиска. Если маркер плюса (+) не используется, *TMFPath* заменяет существующий путь поиска.

*TMFPath*
Путь к каталогу, в котором отладчик должен искать файлы формата сообщений трассировки. Пути, содержащие пробелы, не поддерживаются. Если включены несколько путей, они должны быть разделены точкой с запятой, и в кавычки должна быть заключена вся строка. Если путь находится в кавычках, символ обратной косой черты должен предшествовать escape-символу ( `"c:\\debuggers;c:\\debuggers2"` ). + При использовании *маркера TMFPath* будет добавлен к существующему пути с запятой, автоматически вставленной между существующим путем и новым путем. Однако, если + используется маркер, кавычки нельзя использовать.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 2000 и более поздних версиях Windows. Если вы хотите использовать это расширение с Windows 2000, сначала скопируйте файл Wmitrace.dll из подкаталога winxp каталога средств отладки для Windows в подкаталог w2kfre.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о файлах формата сообщений трассировки см. в разделе "Файлы формата сообщений трассировки" в комплекте драйверов Windows (WDK).

## Замечания

При использовании без параметров **!wmitrace.searchpath** отображает текущий путь поиска.

Файлы формата сообщений трассировки (*.tmf) содержат инструкции по форматированию сообщений двоичной трассировки, создаваемых поставщиком трассировки.

Параметр *TMFPath* должен содержать только путь к каталогу; он не может содержать имя файла. Имя файла TMF — это GUID сообщения, за которым следует расширение .tmf. Если система форматирует сообщение, он считывает GUID сообщения в сообщении и выполняет рекурсивный поиск по файлу TMF, имя которого соответствует GUID сообщения, начиная с указанного каталога.

Для форматирования двоичных сообщений трассировки в буфере требуется файл TMF. Используйте **!wmitrace.searchpath** или **!wmitrace.tmffile** , чтобы указать файл TMF перед использованием **!wmitrace.dynamicprint** или **!wmitrace.logdump** для отображения содержимого буфера трассировки.

Если вы не используете **файл !wmitrace.searchpath** или **!wmitrace.tmffile**, система использует значение переменной среды TRACE_FORMAT_SEARCH_PATH. Если эта переменная отсутствует, он использует файл default.tmf, который включен в Windows. Если система не может найти какие-либо сведения о форматировании для сообщения трассировки, она записывает сообщение об ошибке "Не найдено сведений о формате" вместо содержимого сообщения трассировки.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.setprefix

Статья • 05.04.2024

Расширение **!wmitrace.setprefix** указывает префикс сообщения трассировки, который предопределен для сообщений трассировки из этого сеанса. Это расширение позволяет изменять префикс во время сеанса отладки.

dbgcmd

```
!wmitrace.setprefix [+] PrefixVariables
!wmitrace.setprefix
```

## Параметры

+
Приводит *к добавлению префикса сообщения трассировки префиксом префикса* + Если маркер не используется, *префикс ПрефиксVariable* заменяет существующий префикс сообщения трассировки.

*ПрефиксVariables*
Набор переменных, указывающий формат и данные в префиксе сообщения трассировки.

Переменные имеют формат %n!x!, где %n представляет поле данных и !x! представляет тип данных. Вы также можете включать символы разделения, такие как двоеточия (:), точка с запятой (;), скобки ( ( ) ), скобки ( } ) и скобки ( [ ] ) для разделения полей.

Каждая переменная %n представляет параметр, описанный в следующей таблице.

Развернуть таблицу

| Идентификатор переменной префикса | Тип переменной | Description |
|---|---|---|
| 1% | строка | Понятное имя GUID сообщения для сообщения трассировки. По умолчанию понятное имя GUID сообщения — это имя каталога, в котором был создан поставщик трассировки. |

| Идентификатор переменной префикса | Тип переменной | Description |
| --- | --- | --- |
| 2% | строка | Исходный файл и номер строки.<br><br>Эта переменная представляет понятное имя сообщения трассировки. По умолчанию понятное имя сообщения трассировки — это имя исходного файла и номер строки кода, создающего сообщение трассировки. |
| %3 | ULONG | Идентификатор потока.<br><br>Определяет поток, создающий сообщение трассировки. |
| %4 | строка | Метка времени создания сообщения трассировки. |
| 5% | строка | Время ядра.<br><br>Отображает истекшее время выполнения инструкции в режиме ядра в клещах ЦП во время создания сообщения трассировки. |
| %6 | строка | Время пользователя.<br><br>Отображает истекшее время выполнения инструкции в пользовательском режиме в тиках ЦП во время создания сообщения трассировки. |
| 7% | LONG | Порядковый номер.<br><br>Отображает локальный или глобальный порядковый номер сообщения трассировки. Локальные порядковые номера, уникальные только для этого сеанса трассировки, |

| Идентификатор переменной префикса | Тип переменной | Description |
| --- | --- | --- |
| | | являются значениями по умолчанию. |
| %8 | ULONG | Идентификатор процесса.<br><br>Определяет процесс, создающий сообщение трассировки. |
| %9 | ULONG | Номер ЦП.<br><br>Определяет ЦП, на котором было создано сообщение трассировки. |
| %! FUNC! | строка | Имя функции.<br><br>Отображает имя функции, создающей сообщение трассировки. |
| %! ФЛАГИ! | строка | Отображает имя флагов трассировки, включающих сообщение трассировки. |
| %! УРОВЕНЬ! | строка | Отображает значение уровня трассировки, которое включает сообщение трассировки. |
| %! COMPNAME! | строка | Название компонента.<br><br>Отображает имя компонента поставщика, создающего сообщение трассировки. Имя компонента отображается только в том случае, если он указан в коде трассировки. |
| %! SUBCOMP! | строка | Подкомпонентное имя.<br><br>Отображает имя подкомпонента поставщика, создающего сообщение трассировки. Имя подкомпонента отображается только в том |

| Идентификатор переменной префикса | Тип переменной | Description |
| --- | --- | --- |
| | | случае, если он указан в коде трассировки. |

Символ в восклицательных знаках — это символ преобразования, указывающий формат и точность переменной. Например, %8!04X! указывает идентификатор процесса, отформатированный как четырехзначное, шестнадцатеричное число.

# DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 2000 и более поздних версиях Windows. Если вы хотите использовать это расширение с Windows 2000, сначала скопируйте файл Wmitrace.dll из подкаталога winxp каталога средств отладки для Windows в подкаталог w2kfre.

## Пример

Следующая команда изменяет префикс сообщения трассировки в выходных данных отладчика в следующем формате:

**!wmitrace.setprefix %2!s!: %! FUNC!: %8!04x!. %3!04x!: %4!s!:**

Эта команда расширения задает префикс сообщения трассировки в следующем формате:

*SourceFile_LineNumber: FunctionName: ProcessID.ThreadID: SystemTime:*

В результате сообщения трассировки предопределяются указанными сведениями в указанном формате. Следующий пример кода взят из трассировки примера драйвера Tracedrv в WDK.

dbgcmd

```
tracedrv_c258: TracedrvDispatchDeviceControl: 0af4.0c64: 07/25/2003-
13:55:39.998:  IOCTL = 1
```

# Дополнительная информация

Общие сведения о трассировке событий см. в документации по пакету SDK для Microsoft Windows. Сведения о файлах формата сообщений трассировки см. в разделе "Префикс сообщения трассировки" в документации по WDK.

# Замечания

При использовании без параметров **!wmitrace.setprefix** отображает текущее значение префикса сообщения трассировки.

Префикс сообщения трассировки состоит из данных о сообщении трассировки, которое предварительно добавляется к каждому сообщению трассировки во время трассировки программного обеспечения трассировки программного обеспечения (WPP). Эти данные возникают в файле журнала трассировки (ETL) и файле сообщения трассировки (TMF). Формат и данные можно настроить в префиксе сообщения трассировки.

Префикс сообщения трассировки по умолчанию выглядит следующим образом:

```
dbgcmd
```

```
[%9!d!]%8!04X!.%3!04X!::%4!s! [%1!s!]
```

и создает следующий префикс:

```
dbgcmd
```

```
[CPUNumber]ProcessID.ThreadID::SystemTime [ProviderDirectory]
```

Формат и данные в префиксе сообщения трассировки за пределами отладчика можно изменить, задав переменную среды %TRACE_FORMAT_PREFIX% . Пример, показывающий, как задать префикс сообщения трассировки за пределами отладчика, см. в разделе "Пример 7. Настройка префикса сообщения трассировки" в документации по комплекту драйверов Windows (WDK). Если префикс сообщения трассировки сообщений отличается от значения по умолчанию, эта переменная среды может быть задана на компьютере.

Префикс, заданный с помощью этой команды расширения, влияет только на выходные данные отладчика. Префикс сообщения трассировки, который отображается в журнале трассировки, определяется значением по умолчанию и значением переменной среды %TRACE_FORMAT_PREFIX% среды.

Это расширение полезно только во время трассировки программного обеспечения WPP и более ранних (устаревших) методов трассировки событий для Windows. События трассировки, созданные другими манифестными поставщиками, не используют файлы формата сообщений трассировки (TMF), поэтому это расширение не влияет на них.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.start

Статья • 05.04.2024

Расширение **!wmitrace.start** запускает средство ведения журнала трассировки событий для Windows (ETW) на целевом компьютере.

dbgcmd

```
!wmitrace.start LoggerName [-cir Size | -seq Size] [-f File] [-b Size] [-max
Num] [-min Num] [-kd] [-ft Time]
```

## Параметры

*Имя средства ведения журнала*
Предоставляет имя, которое будет использоваться для сеанса трассировки. *LoggerName* не может содержать пробелы или кавычки.

**-cir***Size*
Приводит к тому, что файл журнала записывается циклическим образом. *Размер задает максимальный размер* файла в байтах. Когда файл достигает этой длины, новые данные записываются в файл циклическим образом, перезаписыв файл начиная с конца. Это не может быть объединено с параметром **-seq** . Если файл не указан ни -cir**, ни -seq**, файл записывается в буферизованном режиме.

**-seq***Num*
Приводит к последовательному написанию файла журнала. *Размер задает максимальный размер* файла в байтах. Когда файл достигает этой длины, старые данные будут удалены с самого начала файла при каждом добавлении новых данных в конец. Это не может быть объединено с параметром **-cir** . Если файл не указан ни -cir**, ни -seq**, файл записывается в буферизованном режиме.

**-f***Файл*
Указывает имя файла журнала, который необходимо создать на целевом компьютере. *Файл* должен содержать абсолютный путь к каталогу и не может содержать пробелы или кавычки.

**-b***Size*
Задает размер каждого буфера в килобайтах. Допустимый диапазон *размера* составляет от 1 до 2048 включительно.

**-max***Num*
Указывает максимальное количество используемых буферов. *Число* может быть любым положительным целым числом.

**-min***Num*
Указывает минимальное количество используемых буферов. *Число* может быть любым положительным целым числом.

**-Kd**
Включает режим фильтрации KD. Сообщения отправляются в отладчик ядра и отображаются на экране.

**-ft***Time*
Указывает длительность таймера очистки в секундах. Начиная с Windows 8, можно указать длительность таймера очистки в миллисекундах, добавив **ms** к значению *Time* . Например, **-ft 100ms**.

**Обратите внимание** , что при запуске сеанса трассировки в режиме фильтрации KD (**-kd**) буферы трассировки на целевом компьютере отправляются отладчику на хост-компьютере для отображения. Этот параметр указывает, как часто буферы на целевом компьютере сбрасываются и отправляются на главный компьютер.

# DLL-библиотеки

Wmitrace.dll.

Это расширение доступно в Windows 7 и более поздних версиях Windows.

# Дополнительная информация

Дополнительные сведения о параметрах этого расширения см. в разделе "Функция StartTraceA" и EVENT_TRACE_PROPERTIES. Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о средствах трассировки см. в комплекте драйверов Windows (WDK).

# Замечания

После использования этого расширения необходимо возобновить выполнение программы (например, с помощью **команды g (Go),** чтобы она ввела силу. Через некоторое время целевой компьютер автоматически переходит в отладчик.

При запуске сеанса трассировки система назначает ему порядковый номер ( *идентификатор* средства ведения журнала). Затем сеанс можно ссылаться на имя средства ведения журнала или идентификатор средства ведения журнала.

Чтобы остановить средство ведения журнала ETW, используйте **!wmitrace.stop**.

---

## Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.stop

Статья • 05.04.2024

Расширение **!wmitrace.stop** останавливает средство ведения журнала трассировки событий для Windows (ETW) на целевом компьютере.

```
dbgcmd
```

```
!wmitrace.stop { LoggerID | LoggerName }
```

## Параметры

*LoggerID*
Указывает сеанс трассировки. *LoggerID* — это порядковый номер, который система назначает каждому сеансу трассировки на компьютере.

*Имя средства ведения журнала*
Указывает сеанс трассировки. *LoggerName* — это текстовое имя, указанное при запуске сеанса трассировки.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 7 и более поздних версиях Windows.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о средствах трассировки см. в комплекте драйверов Windows (WDK).

## Замечания

После использования этого расширения необходимо возобновить выполнение программы (например, с помощью **команды g (Go),** чтобы она ввела силу. Через некоторое время целевой компьютер автоматически переходит в отладчик.

Чтобы запустить средство ведения журнала ETW, используйте **!wmitrace.start**.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.strdump

Статья • 05.04.2024

Расширение **!wmitrace.strdump** отображает структуры трассировки событий WMI. Отображение можно ограничить структурами для определенного сеанса трассировки.

dbgcmd

```
!wmitrace.strdump [ LoggerID | LoggerName ]
```

## Параметры

*LoggerID*
Ограничивает отображение структур трассировки событий для указанного сеанса трассировки. *LoggerID* указывает сеанс трассировки. Это порядковый номер, который система назначает каждому сеансу трассировки на компьютере. Если параметр не указан, отображаются все сеансы трассировки.

*Имя средства ведения журнала*
Ограничивает отображение структур трассировки событий для указанного сеанса трассировки. *LoggerName* — это текстовое имя, указанное при запуске сеанса трассировки. Если параметр не указан, отображаются все сеансы трассировки.

## DLL-библиотеки

Wmitrace.dll

Это расширение доступно в Windows 2000 и более поздних версиях Windows. Если вы хотите использовать это расширение с Windows 2000, сначала скопируйте файл Wmitrace.dll из подкаталога winxp каталога средств отладки для Windows в подкаталог w2kfre.

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Дополнительные сведения о tracelog см. в разделе "Tracelog" в комплекте драйверов Windows (WDK).

# Замечания

Чтобы найти идентификатор средства ведения журнала сеанса трассировки, используйте **расширение !wmitrace.strdump** . Кроме того, можно использовать командный журнал трассировки tracelog -l для перечисления сеансов трассировки и их основных свойств, включая идентификатор средства ведения журнала.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.tmffile

Статья • 05.04.2024

Расширение **!wmitrace.tmffile** указывает файл формата сообщения трассировки (TMF). Файл, указанный этим расширением, используется для форматирования сообщений трассировки, отображаемых или написанных другими расширениями трассировки WMI.

```
dbgcmd
```

```
!wmitrace.tmffile TMFFile
```

## Параметры

*TMFFile*
Указывает файл формата сообщения трассировки.

## DLL-библиотеки

Wmitrace.dll

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows. Сведения о файлах форматирования сообщений трассировки см. в разделе "Файл формата сообщений трассировки" в комплекте драйверов Windows (WDK).

## Замечания

*Файлы* формата сообщений трассировки (.tmf) — это структурированные текстовые файлы, созданные во время трассировки программного обеспечения трассировки программного обеспечения (WPP). Эти файлы содержат инструкции по форматированию сообщений двоичной трассировки трассировки, чтобы их можно было отображать в виде, доступном для чтения человеком.

Чтобы отобразить сообщение трассировки в буфере трассировки (**!wmitrace.logdump**) или записать их в файл (**!wmitrace.logsave**), необходимо сначала определить файлы TMF для сообщений трассировки.

С помощью **!wmitrace.searchpath** можно указать каталог, в котором хранятся файлы TMF. Затем система ищет каталог для файла TMF, содержащего инструкции по форматированию сообщений. (Он использует GUID сообщения для связывания сообщения с правильным файлом TMF.)

Однако для указания определенного файла TMF можно использовать **!wmitrace.tmffile** . Необходимо использовать **!wmitrace.tmffile** , если имя файла TMF не является GUID сообщения, за которым следует расширение .tmf. В противном случае система не найдет ее.

Если вы не используете **файл !wmitrace.searchpath** или **!wmitrace.tmffile**, система использует значение переменной среды TRACE_FORMAT_SEARCH_PATH. Если эта переменная отсутствует, используется файл default.tmf. Если системе не удается найти сведения о форматировании для сообщения трассировки, он записывает сообщение об ошибке "Нет сведений о формате", а не содержимое сообщения трассировки.

**Примечание.** Если драйвер использует UMDF версии 1.11 или более поздней версии, вам не нужно использовать **!wmitrace.searchpath** или **!wmitrace.tmffile**.

Это расширение полезно только во время трассировки программного обеспечения WPP и более ранних (устаревших) методов трассировки событий для Windows. События трассировки, созданные другими манифестными поставщиками, не используют файлы формата сообщений трассировки (TMF), поэтому это расширение нельзя использовать с ними.

---

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A

# !wmitrace.traceoperation

Статья • 05.04.2024

Расширение **!wmitrace.traceoperation** отображает сообщения о ходе выполнения из компонентов трассировки в Windows.

dbgcmd

```
!wmitrace.traceoperation {0 | 1 | 2}
```

## Параметры

**0**
Отключает отображение сообщений о ходе трассировки.

**1**
Включает отображение сообщений о ходе трассировки. Отображаются все сообщения, созданные расширениями отладчика трассировки WMI.

**2**
Включает отображение сообщений о ходе трассировки. Отображаются все сообщения, созданные расширениями отладчика трассировки WMI или Tracefmt.

## DLL-библиотеки

Wmitrace.dll

## Дополнительная информация

Общие сведения о трассировке событий см. в пакете SDK для Microsoft Windows.

## Замечания

Это расширение приводит к отображению подробных выходных данных компонентов трассировки. Эта функция полезна для устранения неполадок трассировки программного обеспечения.

# Обратная связь

**Были ли сведения на этой странице полезными?** Да Нет

Отзыв о продукте | Получить справку в Microsoft Q&A