### CUDBGResult (*CUDBGAPI_st::lookupDeviceCodeSymbol) (char *symName, bool *symFound, uintptr_t *symAddr)

Determines whether a symbol represents a function in device code and returns its virtual address.

#### Parameters

**symName**
- symbol name

**symFound**
- set to true if the symbol is found

**symAddr**
- the symbol virtual address if found

#### Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Since CUDA 3.0.

## 3.10. Events

One of those events will create a CUDBGEvent:

- the elf image of the current kernel has been loaded and the addresses within its DWARF sections have been relocated (and can now be used to set breakpoints),
- a device breakpoint has been hit,
- a CUDA kernel is ready to be launched,
- a CUDA kernel has terminated.

When a CUDBGEvent is created, the debugger is notified by calling the callback functions registered with setNotifyNewEventCallback() after the API struct initialization. It is up to the debugger to decide what method is best to be notified. The debugger API routines cannot be called from within the callback function or the routine will return an error.

Upon notification the debugger is responsible for handling the CUDBGEvents in the event queue by using CUDBGAPI_st::getNextEvent(), and for acknowledging the debugger API that the event has been handled by calling CUDBGAPI_st::acknowledgeEvent(). In the case of an event raised by the device itself, such as a breakpoint being hit, the event queue will be empty. It is the responsibility of the debugger to inspect the hardware any time a CUDBGEvent is received.

Example:

```
CUDBGEvent event;
      CUDBGResult res;
      for (res = cudbgAPI->getNextEvent(&event);
           res == CUDBG_SUCCESS && event.kind != CUDBG_EVENT_INVALID;
           res = cudbgAPI->getNextEvent(&event)) {
          switch (event.kind)
              {
              case CUDBG_EVENT_ELF_IMAGE_LOADED:
                  //...
                  break;
              case CUDBG_EVENT_KERNEL_READY:
                  //...
                  break;
              case CUDBG_EVENT_KERNEL_FINISHED:
                  //...
                  break;
              default:
                  error(...);
              }
          }
```

See cuda-tdep.c and cuda-linux-nat.c files in the cuda-gdb source code for a more detailed example on how to use CUDBGEvent.

# struct CUDBGEvent

Event information container.

# struct CUDBGEventCallbackData

Event information passed to callback set with setNotifyNewEventCallback function.

# struct CUDBGEventCallbackData40

Event information passed to callback set with setNotifyNewEventCallback function.

# enum CUDBGEventKind

CUDA Kernel Events.

## Values

**CUDBG_EVENT_INVALID = 0x000**
Invalid event.

**CUDBG_EVENT_ELF_IMAGE_LOADED = 0x001**
The ELF image for a CUDA source module is available.

**CUDBG_EVENT_KERNEL_READY = 0x002**
A CUDA kernel is about to be launched.

**CUDBG_EVENT_KERNEL_FINISHED = 0x003**
A CUDA kernel has terminated.

**CUDBG_EVENT_INTERNAL_ERROR = 0x004**
An internal error occur. The debugging framework may be unstable.

**CUDBG_EVENT_CTX_PUSH = 0x005**
A CUDA context was pushed.
**CUDBG_EVENT_CTX_POP = 0x006**
A CUDA CTX was popped.
**CUDBG_EVENT_CTX_CREATE = 0x007**
A CUDA CTX was created.
**CUDBG_EVENT_CTX_DESTROY = 0x008**
A CUDA context was destroyed.
**CUDBG_EVENT_TIMEOUT = 0x009**
An timeout event is sent at regular interval. This event can safely ge ignored.
**CUDBG_EVENT_ATTACH_COMPLETE = 0x00a**
The attach process has completed and debugging of device code may start.
**CUDBG_EVENT_DETACH_COMPLETE = 0x00b**
**CUDBG_EVENT_ELF_IMAGE_UNLOADED = 0x00c**
**CUDBG_EVENT_FUNCTIONS_LOADED = 0x00d**

## typedef (*CUDBGNotifyNewEventCallback) (CUDBGEventCallbackData* data)

function type of the function called to notify debugger of the presence of a new event in the event queue.

## typedef (*CUDBGNotifyNewEventCallback31) (void* data)

function type of the function called to notify debugger of the presence of a new event in the event queue.
Deprecated in CUDA 3.2.

## CUDBGResult (*CUDBGAPI_st::acknowledgeEvent30) (CUDBGEvent30 *event)

Inform the debugger API that the event has been processed.

### Parameters

**event**
- pointer to the event that has been processed

### Returns

CUDBG_SUCCESS

Since CUDA 3.0.

Deprecated in CUDA 3.1.

# CUDBGResult (*CUDBGAPI_st::acknowledgeEvents42) ()

Inform the debugger API that synchronous events have been processed.

## Returns

CUDBG_SUCCESS

Since CUDA 3.1.

Deprecated in CUDA 5.0.

# CUDBGResult (*CUDBGAPI_st::acknowledgeSyncEvents) ()

Inform the debugger API that synchronous events have been processed.

## Returns

CUDBG_SUCCESS

Since CUDA 5.0.

# CUDBGResult (*CUDBGAPI_st::getErrorStringEx) (char *buf, uint32_t bufSz, uint32_t *msgSz)

Fills a user-provided buffer with an error message encoded as a null-terminated ASCII string. The error message is specific to the last failed API call and is invalidated after every API call.

## Parameters

**buf**

- the destination buffer

**bufSz**

- the size of the destination buffer in bytes

**msgSz**

- the size of an error message including the terminating null character.

## Returns

CUDBG_SUCCESS, CUDBG_ERROR_BUFFER_TOO_SMALL CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED

Since CUDA 12.2.

**See also:**

getErrorString

## CUDBGResult (*CUDBGAPI_st::getNextAsyncEvent50) (CUDBGEvent50 *event)

Copies the next available event in the asynchronous event queue into 'event' and removes it from the queue. The asynchronous event queue is held separate from the normal event queue, and does not require acknowledgement from the debug client.

### Parameters

**event**

- pointer to an event container where to copy the event parameters

### Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.0.

Deprecated in CUDA 5.5.

## CUDBGResult (*CUDBGAPI_st::getNextAsyncEvent55) (CUDBGEvent55 *event)

Copies the next available event in the asynchronous event queue into 'event' and removes it from the queue. The asynchronous event queue is held separate from the normal event queue, and does not require acknowledgement from the debug client.

### Parameters

**event**

- pointer to an event container where to copy the event parameters

### Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.5.

## CUDBGResult (*CUDBGAPI_st::getNextEvent) (CUDBGEventQueueType type, CUDBGEvent *event)

Copies the next available event into 'event' and removes it from the queue.

### Parameters

**type**

- application event queue type

**event**

- pointer to an event container where to copy the event parameters

### Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 6.0.

## CUDBGResult (*CUDBGAPI_st::getNextEvent30) (CUDBGEvent30 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

### Parameters

**event**

- pointer to an event container where to copy the event parameters

### Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 3.0.

Deprecated in CUDA 3.1.

## CUDBGResult (*CUDBGAPI_st::getNextEvent32) (CUDBGEvent32 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

### Parameters

**event**

- pointer to an event container where to copy the event parameters

### Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 3.1.

Deprecated in CUDA 4.0

## CUDBGResult (*CUDBGAPI_st::getNextEvent42) (CUDBGEvent42 *event)

Copies the next available event in the event queue into 'event' and removes it from the queue.

### Parameters

**event**

- pointer to an event container where to copy the event parameters

### Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 4.0.

Deprecated in CUDA 5.0

## CUDBGResult (*CUDBGAPI_st::getNextSyncEvent50) (CUDBGEvent50 *event)

### Parameters

**event**

- pointer to an event container where to copy the event parameters

### Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.0.

Deprecated in CUDA 5.5.

## CUDBGResult (*CUDBGAPI_st::getNextSyncEvent55) (CUDBGEvent55 *event)

Copies the next available event in the synchronous event queue into 'event' and removes it from the queue.

### Parameters

**event**

- pointer to an event container where to copy the event parameters

### Returns

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.5.

## CUDBGResult (*CUDBGAPI_st::setNotifyNewEventCallback) (CUDBGNotifyNewEventCallback callback)

Provides the API with the function to call to notify the debugger of a new application or device event.

### Parameters

**callback**

- the callback function

### Returns

CUDBG_SUCCESS

Since CUDA 4.1.

# CUDBGResult (*CUDBGAPI_st::setNotifyNewEventCallback31) (CUDBGNotifyNewEventCallback31 callback, void *data)

Provides the API with the function to call to notify the debugger of a new application or device event.

## Parameters

**callback**

- the callback function

**data**

- a pointer to be passed to the callback when called

## Returns

CUDBG_SUCCESS

Since CUDA 3.0.

Deprecated in CUDA 3.2.

# CUDBGResult (*CUDBGAPI_st::setNotifyNewEventCallback40) (CUDBGNotifyNewEventCallback40 callback)

Provides the API with the function to call to notify the debugger of a new application or device event.

## Parameters

**callback**

- the callback function

## Returns

CUDBG_SUCCESS

Since CUDA 3.2.

Deprecated in CUDA 4.1.

# 3.3. Debugger API

The API reference guide for the CUDA debugger.

# Chapter 4. Data Structures

Here are the data structures with brief descriptions:

**cudbgGetAPI**
The CUDA debugger API routines

**CUDBGEvent**
Event information container

**CUDBGEvent::CUDBGEvent::cases_st**

**CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st**

**CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st**

**CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st**

**CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st**

**CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st**

**CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::internalError_st**

**CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st**

**CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st**

**CUDBGEventCallbackData**
Event information passed to callback set with setNotifyNewEventCallback function

**CUDBGEventCallbackData40**
Event information passed to callback set with setNotifyNewEventCallback function

**CUDBGGridInfo**
Grid info

## 4.2. CUDBGEvent Struct Reference

Event information container.

### CUDBGEvent::cases

Information for each type of event.

### CUDBGEventKind CUDBGEvent::kind

Event type.

## 4.3. CUDBGEvent::cases_st Union Reference

## struct CUDBGEvent::cases_st::contextCreate_st
## CUDBGEvent::cases_st::contextCreate

Information about the context being created.

## struct CUDBGEvent::cases_st::contextDestroy_st
## CUDBGEvent::cases_st::contextDestroy

Information about the context being destroyed.

## struct CUDBGEvent::cases_st::contextPop_st
## CUDBGEvent::cases_st::contextPop

Information about the context being popped.

## struct CUDBGEvent::cases_st::contextPush_st
## CUDBGEvent::cases_st::contextPush

Information about the context being pushed.

## struct CUDBGEvent::cases_st::elfImageLoaded_st
## CUDBGEvent::cases_st::elfImageLoaded

Information about the loaded ELF image.

## struct CUDBGEvent::cases_st::internalError_st
## CUDBGEvent::cases_st::internalError

Information about internal erros.

## struct CUDBGEvent::cases_st::kernelFinished_st
## CUDBGEvent::cases_st::kernelFinished

Information about the kernel that just terminated.

## struct CUDBGEvent::cases_st::kernelReady_st
## CUDBGEvent::cases_st::kernelReady

Information about the kernel ready to be launched.

## 4.4. CUDBGEvent::cases_st::contextCreate_st Struct Reference

### uint64_t CUDBGEvent::cases_st::contextCreate_st::context

the context being created.

### uint32_t CUDBGEvent::cases_st::contextCreate_st::dev

device index of the context.

### uint32_t CUDBGEvent::cases_st::contextCreate_st::tid

host thread id (or LWP id) of the thread hosting the context (Linux only).

## 4.5. CUDBGEvent::cases_st::contextDestroy_st Struct Reference

### uint64_t CUDBGEvent::cases_st::contextDestroy_st::context

the context being destroyed.

### uint32_t CUDBGEvent::cases_st::contextDestroy_st::dev

device index of the context.

### uint32_t CUDBGEvent::cases_st::contextDestroy_st::tid

host thread id (or LWP id) of the thread hosting the context (Linux only).

## 4.6. CUDBGEvent::cases_st::contextPop_st Struct Reference

### uint64_t CUDBGEvent::cases_st::contextPop_st::context

the context being popped.

### uint32_t CUDBGEvent::cases_st::contextPop_st::dev

device index of the context.

### uint32_t CUDBGEvent::cases_st::contextPop_st::tid

host thread id (or LWP id) of the thread hosting the context (Linux only).

## 4.7. CUDBGEvent::cases_st::contextPush_st Struct Reference

### uint64_t CUDBGEvent::cases_st::contextPush_st::context

the context being pushed.

### uint32_t CUDBGEvent::cases_st::contextPush_st::dev

device index of the context.

### uint32_t CUDBGEvent::cases_st::contextPush_st::tid

host thread id (or LWP id) of the thread hosting the context (Linux only).

## 4.8. CUDBGEvent::cases_st::elfImageLoaded_st Struct Reference

### uint64_t CUDBGEvent::cases_st::elfImageLoaded_st::context

context of the kernel.

### uint32_t CUDBGEvent::cases_st::elfImageLoaded_st::dev

device index of the kernel.

### uint64_t CUDBGEvent::cases_st::elfImageLoaded_st::handle

ELF image handle.

### uint64_t CUDBGEvent::cases_st::elfImageLoaded_st::module

module of the kernel.

### uint32_t CUDBGEvent::cases_st::elfImageLoaded_st::properties

ELF image properties.

### uint64_t CUDBGEvent::cases_st::elfImageLoaded_st::size

size of the ELF image (64-bit).

## 4.9. CUDBGEvent::cases_st::internalError_st Struct Reference

### CUDBGResult CUDBGEvent::cases_st::internalError_st::errorType

Type of the internal error.

## 4.10. CUDBGEvent::cases_st::kernelFinished_st Struct Reference

### uint64_t CUDBGEvent::cases_st::kernelFinished_st::context

context of the kernel.

### uint32_t CUDBGEvent::cases_st::kernelFinished_st::dev

device index of the kernel.

### uint64_t CUDBGEvent::cases_st::kernelFinished_st::function

function of the kernel.

### uint64_t CUDBGEvent::cases_st::kernelFinished_st::functionEntry

entry PC of the kernel.

### uint64_t CUDBGEvent::cases_st::kernelFinished_st::gridId

grid index of the kernel.

### uint64_t CUDBGEvent::cases_st::kernelFinished_st::module

module of the kernel.

### uint32_t CUDBGEvent::cases_st::kernelFinished_st::tid

host thread id (or LWP id) of the thread hosting the kernel (Linux only).

## 4.11. CUDBGEvent::cases_st::kernelReady_st Struct Reference

## CuDim3 CUDBGEvent::cases_st::kernelReady_st::blockDim

block dimensions of the kernel.

## uint64_t CUDBGEvent::cases_st::kernelReady_st::context

context of the kernel.

## uint32_t CUDBGEvent::cases_st::kernelReady_st::dev

device index of the kernel.

## uint64_t CUDBGEvent::cases_st::kernelReady_st::function

function of the kernel.

## uint64_t CUDBGEvent::cases_st::kernelReady_st::functionEntry

entry PC of the kernel.

## CuDim3 CUDBGEvent::cases_st::kernelReady_st::gridDim

grid dimensions of the kernel.

## uint64_t CUDBGEvent::cases_st::kernelReady_st::gridId

grid index of the kernel.

## uint64_t CUDBGEvent::cases_st::kernelReady_st::module

module of the kernel.

## uint64_t CUDBGEvent::cases_st::kernelReady_st::parentGridId

64-bit grid index of the parent grid.

### uint32_t CUDBGEvent::cases_st::kernelReady_st::tid

host thread id (or LWP id) of the thread hosting the kernel (Linux only).

### CUDBGKernelType CUDBGEvent::cases_st::kernelReady_st::type

the type of the kernel: system or application.

## 4.12. CUDBGEventCallbackData Struct Reference

Event information passed to callback set with setNotifyNewEventCallback function.

### uint32_t CUDBGEventCallbackData::tid

Host thread id of the context generating the event. Zero if not available.

### uint32_t CUDBGEventCallbackData::timeout

A boolean notifying the debugger that the debug API timed while waiting for a reponse from the debugger to a previous event. It is up to the debugger to decide what to do in response to a timeout.

## 4.13. CUDBGEventCallbackData40 Struct Reference

Event information passed to callback set with setNotifyNewEventCallback function. Deprecated in CUDA 4.1.

### uint32_t CUDBGEventCallbackData40::tid

Host thread id of the context generating the event. Zero if not available.

## 4.14. CUDBGGridInfo Struct Reference

Grid info.

## CuDim3 CUDBGGridInfo::blockDim

The block dimensions.

## uint64_t CUDBGGridInfo::context

The context this grid belongs to.

## uint32_t CUDBGGridInfo::dev

The index of the device this grid is running on.

## uint64_t CUDBGGridInfo::function

The function corresponding to this grid.

## uint64_t CUDBGGridInfo::functionEntry

The entry address of the function corresponding to this grid.

## CuDim3 CUDBGGridInfo::gridDim

The grid dimensions.

## uint64_t CUDBGGridInfo::gridId64

The 64-bit grid ID of this grid.

## uint64_t CUDBGGridInfo::module

The module this grid belongs to.

## CUDBGKernelOrigin CUDBGGridInfo::origin

The origin of this grid, CPU or GPU.

## uint64_t CUDBGGridInfo::parentGridId

The 64-bit grid ID that launched this grid.

## uint32_t CUDBGGridInfo::tid

The host thread ID that launched this grid.

## CUDBGKernelType CUDBGGridInfo::type

The type of the grid.

# 4.1. Debugger API

The API reference guide for the CUDA debugger.

# Chapter 5. Data Fields

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

## A

**acknowledgeEvent30**
cudbgGetAPI
**acknowledgeEvents42**
cudbgGetAPI
**acknowledgeSyncEvents**
cudbgGetAPI

## B

**blockDim**
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
CUDBGGridInfo

## C

**cases**
CUDBGEvent
**clearAttachState**
cudbgGetAPI
**context**
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st
CUDBGGridInfo
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st
**contextCreate**
CUDBGEvent::CUDBGEvent::cases_st

## D



## E



## F



## G

**getBlockDim**
cudbgGetAPI
**getClusterDim**
cudbgGetAPI
**getConstBankAddress**
cudbgGetAPI
**getConstBankAddress123**
cudbgGetAPI
**getDeviceName**
cudbgGetAPI
**getDevicePCIBusInfo**
cudbgGetAPI
**getDeviceType**
cudbgGetAPI
**getElfImage**
cudbgGetAPI
**getElfImage32**
cudbgGetAPI
**getElfImageByHandle**
cudbgGetAPI
**getErrorStringEx**
cudbgGetAPI
**getGridAttribute**
cudbgGetAPI
**getGridAttributes**
cudbgGetAPI
**getGridDim**
cudbgGetAPI
**getGridDim32**
cudbgGetAPI
**getGridInfo**
cudbgGetAPI
**getGridInfo55**
cudbgGetAPI
**getGridStatus**
cudbgGetAPI
**getGridStatus50**
cudbgGetAPI
**getHostAddrFromDeviceAddr**
cudbgGetAPI
**getLoadedFunctionInfo**
cudbgGetAPI

**getLoadedFunctionInfo118**
cudbgGetAPI
**getManagedMemoryRegionInfo**
cudbgGetAPI
**getNextAsyncEvent50**
cudbgGetAPI
**getNextAsyncEvent55**
cudbgGetAPI
**getNextEvent**
cudbgGetAPI
**getNextEvent30**
cudbgGetAPI
**getNextEvent32**
cudbgGetAPI
**getNextEvent42**
cudbgGetAPI
**getNextSyncEvent50**
cudbgGetAPI
**getNextSyncEvent55**
cudbgGetAPI
**getNumDevices**
cudbgGetAPI
**getNumLanes**
cudbgGetAPI
**getNumPredicates**
cudbgGetAPI
**getNumRegisters**
cudbgGetAPI
**getNumSMs**
cudbgGetAPI
**getNumUniformPredicates**
cudbgGetAPI
**getNumUniformRegisters**
cudbgGetAPI
**getNumWarps**
cudbgGetAPI
**getPhysicalRegister30**
cudbgGetAPI
**getPhysicalRegister40**
cudbgGetAPI
**getSmType**
cudbgGetAPI

## H



## I



## K



## L



## M

**module**
CUDBGGridInfo
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st

## O

**origin**
CUDBGGridInfo

## P

**parentGridId**
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
CUDBGGridInfo
**properties**
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st

## R

**readActiveLanes**
cudbgGetAPI
**readBlockIdx**
cudbgGetAPI
**readBlockIdx32**
cudbgGetAPI
**readBrokenWarps**
cudbgGetAPI
**readCallDepth**
cudbgGetAPI
**readCallDepth32**
cudbgGetAPI
**readCCRegister**
cudbgGetAPI
**readClusterIdx**
cudbgGetAPI
**readCodeMemory**
cudbgGetAPI
**readConstMemory**
cudbgGetAPI
**readDeviceExceptionState**
cudbgGetAPI
**readDeviceExceptionState80**
cudbgGetAPI

**readErrorPC**
cudbgGetAPI
**readGenericMemory**
cudbgGetAPI
**readGlobalMemory**
cudbgGetAPI
**readGlobalMemory31**
cudbgGetAPI
**readGlobalMemory55**
cudbgGetAPI
**readGridId**
cudbgGetAPI
**readGridId50**
cudbgGetAPI
**readLaneException**
cudbgGetAPI
**readLaneStatus**
cudbgGetAPI
**readLocalMemory**
cudbgGetAPI
**readParamMemory**
cudbgGetAPI
**readPC**
cudbgGetAPI
**readPinnedMemory**
cudbgGetAPI
**readPredicates**
cudbgGetAPI
**readRegister**
cudbgGetAPI
**readRegisterRange**
cudbgGetAPI
**readReturnAddress**
cudbgGetAPI
**readReturnAddress32**
cudbgGetAPI
**readSharedMemory**
cudbgGetAPI
**readSyscallCallDepth**
cudbgGetAPI
**readTextureMemory**
cudbgGetAPI

**readTextureMemoryBindless**
cudbgGetAPI
**readThreadIdx**
cudbgGetAPI
**readUniformPredicates**
cudbgGetAPI
**readUniformRegisterRange**
cudbgGetAPI
**readValidLanes**
cudbgGetAPI
**readValidWarps**
cudbgGetAPI
**readVirtualPC**
cudbgGetAPI
**readVirtualReturnAddress**
cudbgGetAPI
**readVirtualReturnAddress32**
cudbgGetAPI
**readWarpState**
cudbgGetAPI
**readWarpState60**
cudbgGetAPI
**requestCleanupOnDetach**
cudbgGetAPI
**requestCleanupOnDetach55**
cudbgGetAPI
**resumeDevice**
cudbgGetAPI
**resumeWarpsUntilPC**
cudbgGetAPI

## S

**setBreakpoint**
cudbgGetAPI
**setBreakpoint31**
cudbgGetAPI
**setKernelLaunchNotificationMode**
cudbgGetAPI
**setNotifyNewEventCallback**
cudbgGetAPI
**setNotifyNewEventCallback31**
cudbgGetAPI

## T



## U



## W

**writeGlobalMemory31**
cudbgGetAPI
**writeGlobalMemory55**
cudbgGetAPI
**writeLocalMemory**
cudbgGetAPI
**writeParamMemory**
cudbgGetAPI
**writePinnedMemory**
cudbgGetAPI
**writePredicates**
cudbgGetAPI
**writeRegister**
cudbgGetAPI
**writeSharedMemory**
cudbgGetAPI
**writeUniformPredicates**
cudbgGetAPI
**writeUniformRegister**
cudbgGetAPI

# Chapter 6. Deprecated List

**Global CUDBGAPI_st::requestCleanupOnDetach55 )(void)**

in CUDA 6.0

**Class CUDBGEventCallbackData40**

in CUDA 4.1.

**Global CUDBGAPI_st::singleStepWarp40 )(uint32_t dev, uint32_t sm, uint32_t wp)**

in CUDA 4.1.

**Global CUDBGAPI_st::singleStepWarp41 )(uint32_t dev, uint32_t sm, uint32_t wp, uint64_t *warpMask)**

in CUDA 7.5.

**Global CUDBGAPI_st::setBreakpoint31 )(uint64_t addr)**

in CUDA 3.2.

**Global CUDBGAPI_st::unsetBreakpoint31 )(uint64_t addr)**

in CUDA 3.2.

**Global CUDBGAPI_st::readBlockIdx32 )(uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *blockIdx)**

in CUDA 4.0.

**Global CUDBGAPI_st::readCallDepth32 )(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *depth)**

in CUDA 4.0.

**Global CUDBGAPI_st::readGlobalMemory31 )(uint32_t dev, uint64_t addr, void *buf, uint32_t sz)**

in CUDA 3.2.

**Global CUDBGAPI_st::readGlobalMemory55 )(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)**

in CUDA 6.0.

**Global CUDBGAPI_st::readGridId50 )(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *gridId)**

in CUDA 5.5.

**Global CUDBGAPI_st::readReturnAddress32 )(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)**

in CUDA 4.0.

**Global CUDBGAPI_st::readVirtualReturnAddress32 )(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)**

in CUDA 4.0.

**Global CUDBGAPI_st::writeGlobalMemory31 )(uint32_t dev, uint64_t addr, const void *buf, uint32_t sz)**

in CUDA 3.2.

**Global CUDBGAPI_st::writeGlobalMemory55 )(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz)**

in CUDA 6.0.

**Global CUDBGAPI_st::getElfImage32 )(uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint32_t *size)**

in CUDA 4.0.

**Global CUDBGAPI_st::getGridDim32 )(uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *gridDim)**

in CUDA 4.0.

**Global CUDBGAPI_st::getGridStatus50 )(uint32_t dev, uint32_t gridId, CUDBGGridStatus *status)**

in CUDA 5.5.

**Global CUDBGAPI_st::getPhysicalRegister30 )(uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)**

in CUDA 3.1.

**Global CUDBGAPI_st::getPhysicalRegister40 )(uint32_t dev, uint32_t sm, uint32_t wp, uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)**

in CUDA 4.1.

**Global CUDBGAPI_st::isDeviceCodeAddress55 )(uintptr_t addr, bool *isDeviceAddress)**

in CUDA 6.0

**Global CUDBGNotifyNewEventCallback31**

in CUDA 3.2.

**Global CUDBGAPI_st::acknowledgeEvent30 )(CUDBGEvent30 *event)**

in CUDA 3.1.

**Global CUDBGAPI_st::acknowledgeEvents42 )(void)**

in CUDA 5.0.

**Global CUDBGAPI_st::getNextAsyncEvent50 )(CUDBGEvent50 *event)**

in CUDA 5.5.

**Global CUDBGAPI_st::getNextEvent30 )(CUDBGEvent30 *event)**

in CUDA 3.1.

**Global CUDBGAPI_st::getNextEvent32 )(CUDBGEvent32 *event)**

in CUDA 4.0

**Global CUDBGAPI_st::getNextEvent42 )(CUDBGEvent42 *event)**

in CUDA 5.0

**Global CUDBGAPI_st::getNextSyncEvent50 )(CUDBGEvent50 *event)**

in CUDA 5.5.

**Global CUDBGAPI_st::setNotifyNewEventCallback31 )(CUDBGNotifyNewEventCallback31 callback, void *data)**

in CUDA 3.2.

**Global CUDBGAPI_st::setNotifyNewEventCallback40 )(CUDBGNotifyNewEventCallback40 callback)**

in CUDA 4.1.

### Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

### OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

### Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## Copyright



NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
https://www.nvidia.com