



# cuDLA API

## API Reference Manual

# Table of Contents

Chapter 1. Modules.....	1
1.1. Data types used by cuDLA driver.....	1
cudlaDevAttribute.....	2
cudlaExternalMemoryHandleDesc_t.....	2
cudlaExternalSemaphoreHandleDesc_t.....	2
CudlaFence.....	2
cudlaModuleAttribute.....	2
cudlaModuleTensorDescriptor.....	2
cudlaSignalEvents.....	2
cudlaTask.....	2
cudlaWaitEvents.....	2
cudlaAccessPermissionFlags.....	2
cudlaDevAttributeType.....	2
cudlaFenceType.....	3
cudlaMode.....	3
cudlaModuleAttributeType.....	3
cudlaModuleLoadFlags.....	3
cudlaNvSciSyncAttributes.....	4
cudlaStatus.....	4
cudlaSubmissionFlags.....	5
cudlaDevHandle.....	5
cudlaModule.....	6
1.2. Modules.....	6
cudlaDevAttribute.....	6
cudlaExternalMemoryHandleDesc_t.....	6
cudlaExternalSemaphoreHandleDesc_t.....	6
CudlaFence.....	6
cudlaModuleAttribute.....	6
cudlaModuleTensorDescriptor.....	6
cudlaSignalEvents.....	6
cudlaTask.....	6
cudlaWaitEvents.....	6
cudlaAccessPermissionFlags.....	6
cudlaDevAttributeType.....	7
cudlaFenceType.....	7
cudlaMode.....	7

cudlaModuleAttributeType.....	7
cudlaModuleLoadFlags.....	8
cudlaNvSciSyncAttributes.....	8
cudlaStatus.....	8
cudlaSubmissionFlags.....	9
cudlaDevHandle.....	10
cudlaModule.....	10
<b>Chapter 2. Data Structures.....</b>	<b>11</b>
cudlaDevAttribute.....	11
deviceVersion.....	11
unifiedAddressingSupported.....	11
cudlaExternalMemoryHandleDesc_t.....	11
extBufObject.....	12
size.....	12
cudlaExternalSemaphoreHandleDesc_t.....	12
extSyncObject.....	12
CudlaFence.....	12
fence.....	12
type.....	12
cudlaModuleAttribute.....	12
inputTensorDesc.....	13
numInputTensors.....	13
numOutputTensors.....	13
outputTensorDesc.....	13
cudlaModuleTensorDescriptor.....	13
cudlaSignalEvents.....	13
devPtrs.....	13
eofFences.....	13
numEvents.....	13
cudlaTask.....	14
inputTensor.....	14
moduleHandle.....	14
numInputTensors.....	14
numOutputTensors.....	14
outputTensor.....	14
signalEvents.....	14
waitEvents.....	14
cudlaWaitEvents.....	14

numEvents.....	14
preFences.....	15
Chapter 3. Data Fields.....	16

---

# Chapter 1. Modules

Here is a list of all modules:

- ▶ [Data types used by cuDLA driver](#)
- ▶ [cuDLA API](#)

## 1.1. Data types used by cuDLA driver

`union cudlaDevAttribute`

`struct cudlaExternalMemoryHandleDesc_t`

`struct cudlaExternalSemaphoreHandleDesc_t`

`struct CudlaFence`

`union cudlaModuleAttribute`

`struct cudlaModuleTensorDescriptor`

`struct cudlaSignalEvents`

`struct cudlaTask`

`struct cudlaWaitEvents`

`enum cudlaAccessPermissionFlags`

Access permission flags for importing NvSciBuffers

#### Values

**CUDLA\_READ\_WRITE\_PERM = 0**

Flag to import memory with read-write permission

**CUDLA\_READ\_ONLY\_PERM = 1**

Flag to import memory with read-only permission

**CUDLA\_TASK\_STATISTICS = 1<<1**

Flag to indicate buffer as layerwise statistics buffer.

`enum cudlaDevAttributeType`

Device attribute type.

#### Values

**CUDLA\_UNIFIED\_ADDRESSING = 0**

Flag to check for support for UVA.

**CUDLA\_DEVICE\_VERSION = 1**

Flag to check for DLA HW version.

## enum cudlaFenceType

Supported fence types.

### Values

**CUDLA\_NVSCISYNC\_FENCE = 1**

NvSciSync fence type for EOF.

**CUDLA\_NVSCISYNC\_FENCE\_SOF = 2**

## enum cudlaMode

Device creation modes.

### Values

**CUDLA\_CUDA\_DLA = 0**

Hybrid mode.

**CUDLA\_STANDALONE = 1**

Standalone mode.

## enum cudlaModuleAttributeType

Module attribute types.

### Values

**CUDLA\_NUM\_INPUT\_TENSORS = 0**

Flag to retrieve number of input tensors.

**CUDLA\_NUM\_OUTPUT\_TENSORS = 1**

Flag to retrieve number of output tensors.

**CUDLA\_INPUT\_TENSOR\_DESCRIPTOR = 2**

Flag to retrieve all the input tensor descriptors.

**CUDLA\_OUTPUT\_TENSOR\_DESCRIPTOR = 3**

Flag to retrieve all the output tensor descriptors.

**CUDLA\_NUM\_OUTPUT\_TASK\_STATISTICS = 4**

Flag to retrieve total number of output task statistics buffer.

**CUDLA\_OUTPUT\_TASK\_STATISTICS\_DESCRIPTOR = 5**

Flag to retrieve all the output task statistics descriptors.

## enum cudlaModuleLoadFlags

Module load flags for [cudlaModuleLoadFromMemory](#).

## Values

**CUDLA\_MODULE\_DEFAULT = 0**

Default flag.

**CUDLA\_MODULE\_ENABLE\_FAULT\_DIAGNOSTICS = 1**

Flag to load a module that is used to perform permanent fault diagnostics for DLA HW.

## enum cudlaNvSciSyncAttributes

cuDLA NvSciSync attributes.

### Values

**CUDLA\_NVSCISYNC\_ATTR\_WAIT = 1**

Wait attribute.

**CUDLA\_NVSCISYNC\_ATTR\_SIGNAL = 2**

Signal attribute.

## enum cudlaStatus

Error codes.

### Values

**cudlaSuccess = 0**

The API call returned with no errors.

**cudlaErrorInvalidParam = 1**

This indicates that one or more parameters passed to the API is/are incorrect.

**cudlaErrorOutOfResources = 2**

This indicates that the API call failed due to lack of underlying resources.

**cudlaErrorCreationFailed = 3**

This indicates that an internal error occurred during creation of device handle.

**cudlaErrorInvalidAddress = 4**

This indicates that the memory object being passed in the API call has not been registered before.

**cudlaErrorOs = 5**

This indicates that an OS error occurred.

**cudlaErrorCuda = 6**

This indicates that there was an error in a CUDA operation as part of the API call.

**cudlaErrorUmd = 7**

This indicates that there was an error in the DLA runtime for the API call.

**cudlaErrorInvalidDevice = 8**

This indicates that the device handle passed to the API call is invalid.

**cudlaErrorInvalidAttribute = 9**

This indicates that an invalid attribute is being requested.

**cudlaErrorIncompatibleDlaSWVersion = 10**



This indicates that the underlying DLA runtime is incompatible with the current cuDLA version.

**cudaErrorMemoryRegistered = 11**

This indicates that the memory object is already registered.

**cudaErrorInvalidModule = 12**

This indicates that the module being passed is invalid.

**cudaErrorUnsupportedOperation = 13**

This indicates that the operation being requested by the API call is unsupported.

**cudaErrorNvSci = 14**

This indicates that the NvSci operation requested by the API call failed.

**cudaErrorDlaErrInvalidInput = 0x40000001**

DLA HW Error.

**cudaErrorDlaErrInvalidPreAction = 0x40000002**

DLA HW Error.

**cudaErrorDlaErrNoMem = 0x40000003**

DLA HW Error.

**cudaErrorDlaErrProcessorBusy = 0x40000004**

DLA HW Error.

**cudaErrorDlaErrTaskStatusMismatch = 0x40000005**

DLA HW Error.

**cudaErrorDlaErrEngineTimeout = 0x40000006**

DLA HW Error.

**cudaErrorDlaErrDataMismatch = 0x40000007**

DLA HW Error.

**cudaErrorUnknown = 0x7fffffff**

This indicates that an unknown error has occurred.

## enum cudaSubmissionFlags

Task submission flags for [cudaSubmitTask](#).

### Values

**CUDA\_SUBMIT\_NOOP = 1**

Flag to specify that the submitted task must be bypassed for execution.

**CUDA\_SUBMIT\_SKIP\_LOCK\_ACQUIRE = 1<<1**

Flag to specify that the global lock acquire must be skipped.

**CUDA\_SUBMIT\_DIAGNOSTICS\_TASK = 1<<2**

Flag to specify that the submitted task is to run permanent fault diagnostics for DLA HW.

## typedef cudaDevHandle\_t \*cudaDevHandle

cuDLA Device Handle

```
typedef cudlaModule_t *cudlaModule
```

cuDLA Module Handle

## 1.2. Modules

Here is a list of all modules:

- ▶ [Data types used by cuDLA driver](#)
- ▶ [cuDLA API](#)

### 1.1. Data types used by cuDLA driver

```
union cudlaDevAttribute
```

```
struct cudlaExternalMemoryHandleDesc_t
```

```
struct cudlaExternalSemaphoreHandleDesc_t
```

```
struct CudlaFence
```

```
union cudlaModuleAttribute
```

```
struct cudlaModuleTensorDescriptor
```

```
struct cudlaSignalEvents
```

```
struct cudlaTask
```

```
struct cudlaWaitEvents
```

```
enum cudlaAccessPermissionFlags
```

Access permission flags for importing NvSciBuffers

Values

```
CUDLA_READ_WRITE_PERM = 0
```

Flag to import memory with read-write permission

```
CUDLA_READ_ONLY_PERM = 1
```

Flag to import memory with read-only permission

**CUDLA\_TASK\_STATISTICS = 1<<1**

Flag to indicate buffer as layerwise statistics buffer.

## enum cudlaDevAttributeType

Device attribute type.

### Values

**CUDLA\_UNIFIED\_ADDRESSING = 0**

Flag to check for support for UVA.

**CUDLA\_DEVICE\_VERSION = 1**

Flag to check for DLA HW version.

## enum cudlaFenceType

Supported fence types.

### Values

**CUDLA\_NVSCISYNC\_FENCE = 1**

NvSciSync fence type for EOF.

**CUDLA\_NVSCISYNC\_FENCE\_SOF = 2**

## enum cudlaMode

Device creation modes.

### Values

**CUDLA\_CUDA\_DLA = 0**

Hybrid mode.

**CUDLA\_STANDALONE = 1**

Standalone mode.

## enum cudlaModuleAttributeType

Module attribute types.

### Values

**CUDLA\_NUM\_INPUT\_TENSORS = 0**

Flag to retrieve number of input tensors.

**CUDLA\_NUM\_OUTPUT\_TENSORS = 1**

Flag to retrieve number of output tensors.

**CUDLA\_INPUT\_TENSOR\_DESCRIPTOR = 2**

Flag to retrieve all the input tensor descriptors.

**CUDLA\_OUTPUT\_TENSOR\_DESCRIPTOR = 3**

Flag to retrieve all the output tensor descriptors.

**CUDLA\_NUM\_OUTPUT\_TASK\_STATISTICS = 4**

Flag to retrieve total number of output task statistics buffer.

**CUDLA\_OUTPUT\_TASK\_STATISTICS\_DESCRIPTOR = 5**

Flag to retrieve all the output task statistics descriptors.

## enum cudlaModuleLoadFlags

Module load flags for [cudlaModuleLoadFromMemory](#).

### Values

**CUDLA\_MODULE\_DEFAULT = 0**

Default flag.

**CUDLA\_MODULE\_ENABLE\_FAULT\_DIAGNOSTICS = 1**

Flag to load a module that is used to perform permanent fault diagnostics for DLA HW.

## enum cudlaNvSciSyncAttributes

cuDLA NvSciSync attributes.

### Values

**CUDLA\_NVSCISYNC\_ATTR\_WAIT = 1**

Wait attribute.

**CUDLA\_NVSCISYNC\_ATTR\_SIGNAL = 2**

Signal attribute.

## enum cudlaStatus

Error codes.

### Values

**cudlaSuccess = 0**

The API call returned with no errors.

**cudlaErrorInvalidParam = 1**

This indicates that one or more parameters passed to the API is/are incorrect.

**cudlaErrorOutOfResources = 2**

This indicates that the API call failed due to lack of underlying resources.

**cudlaErrorCreationFailed = 3**

This indicates that an internal error occurred during creation of device handle.

**cudlaErrorInvalidAddress = 4**

This indicates that the memory object being passed in the API call has not been registered before.

**cudlaErrorOs = 5**

This indicates that an OS error occurred.

**cudaErrorCuda = 6**

This indicates that there was an error in a CUDA operation as part of the API call.

**cudaErrorUmd = 7**

This indicates that there was an error in the DLA runtime for the API call.

**cudaErrorInvalidDevice = 8**

This indicates that the device handle passed to the API call is invalid.

**cudaErrorInvalidAttribute = 9**

This indicates that an invalid attribute is being requested.

**cudaErrorIncompatibleDlaSWVersion = 10**

This indicates that the underlying DLA runtime is incompatible with the current cuDLA version.

**cudaErrorMemoryRegistered = 11**

This indicates that the memory object is already registered.

**cudaErrorInvalidModule = 12**

This indicates that the module being passed is invalid.

**cudaErrorUnsupportedOperation = 13**

This indicates that the operation being requested by the API call is unsupported.

**cudaErrorNvSci = 14**

This indicates that the NvSci operation requested by the API call failed.

**cudaErrorDlaErrInvalidInput = 0x40000001**

DLA HW Error.

**cudaErrorDlaErrInvalidPreAction = 0x40000002**

DLA HW Error.

**cudaErrorDlaErrNoMem = 0x40000003**

DLA HW Error.

**cudaErrorDlaErrProcessorBusy = 0x40000004**

DLA HW Error.

**cudaErrorDlaErrTaskStatusMismatch = 0x40000005**

DLA HW Error.

**cudaErrorDlaErrEngineTimeout = 0x40000006**

DLA HW Error.

**cudaErrorDlaErrDataMismatch = 0x40000007**

DLA HW Error.

**cudaErrorUnknown = 0x7fffffff**

This indicates that an unknown error has occurred.

## enum cudaSubmissionFlags

Task submission flags for [cudaSubmitTask](#).

### Values

**CUDA\_SUBMIT\_NOOP = 1**

Flag to specify that the submitted task must be bypassed for execution.

**CUDA\_SUBMIT\_SKIP\_LOCK\_ACQUIRE = 1<<1**

Flag to specify that the global lock acquire must be skipped.

**CUDLA\_SUBMIT\_DIAGNOSTICS\_TASK = 1<<2**

Flag to specify that the submitted task is to run permanent fault diagnostics for DLA HW.

**typedef cudlaDevHandle\_t \*cudlaDevHandle**

cuDLA Device Handle

**typedef cudlaModule\_t \*cudlaModule**

cuDLA Module Handle

---

## Chapter 2. Data Structures

Here are the data structures with brief descriptions:

[cudaDevAttribute](#)

[cudaExternalMemoryHandleDesc](#)

[cudaExternalSemaphoreHandleDesc](#)

[CudlaFence](#)

[cudaModuleAttribute](#)

[cudaModuleTensorDescriptor](#)

[cudaSignalEvents](#)

[cudaTask](#)

[cudaWaitEvents](#)

### 2.1. `cudaDevAttribute` Union Reference

Device attribute.

`uint32_t cudaDevAttribute::deviceVersion`

DLA device version. Xavier has 1.0 and Orin has 2.0.

`uint8_t cudaDevAttribute::unifiedAddressingSupported`

Returns 0 if unified addressing is not supported.

### 2.2. `cudaExternalMemoryHandleDesc_t` Struct Reference

External memory handle descriptor.

`const void`

`*cudaExternalMemoryHandleDesc_t::extBufObject`

A handle representing an external memory object.

`unsigned long long`

`cudaExternalMemoryHandleDesc_t::size`

Size of the memory allocation

## 2.3. `cudaExternalSemaphoreHandleDesc_t` Struct Reference

External semaphore handle descriptor.

`const void`

`*cudaExternalSemaphoreHandleDesc_t::extSyncObject`

A handle representing an external synchronization object.

## 2.4. `CudlaFence` Struct Reference

Fence description.

`void *CudlaFence::fence`

Fence.

`cudaFenceType CudlaFence::type`

Fence type.

## 2.5. `cudaModuleAttribute` Union Reference

Module attribute.



## `cudaModuleTensorDescriptor`

`*cudaModuleAttribute::inputTensorDesc`

Returns an array of input tensor descriptors.

`uint32_t cudaModuleAttribute::numInputTensors`

Returns the number of input tensors.

`uint32_t cudaModuleAttribute::numOutputTensors`

Returns the number of output tensors.

## `cudaModuleTensorDescriptor`

`*cudaModuleAttribute::outputTensorDesc`

Returns an array of output tensor descriptors.

## 2.6. `cudaModuleTensorDescriptor` Struct Reference

Tensor descriptor.

## 2.7. `cudaSignalEvents` Struct Reference

Signal events for [cudaSubmitTask](#)

`const **cudaSignalEvents::devPtrs`

Array of registered synchronization objects (via [cudaImportExternalSemaphore](#)).

`CudlaFence *cudaSignalEvents::eofFences`

Array of fences pointers for all the signal events corresponding to the synchronization objects.

`uint32_t cudaSignalEvents::numEvents`

Total number of signal events.

## 2.8. cudlaTask Struct Reference

Structure of Task.

**const \*\*cudlaTask::inputTensor**

Array of input tensors.

**cudlaModule cudlaTask::moduleHandle**

cuDLA module handle.

**uint32\_t cudlaTask::numInputTensors**

Number of input tensors.

**uint32\_t cudlaTask::numOutputTensors**

Number of output tensors.

**const \*\*cudlaTask::outputTensor**

Array of output tensors.

**cudlaSignalEvents \*cudlaTask::signalEvents**

Signal events.

**const cudlaWaitEvents \*cudlaTask::waitEvents**

Wait events.

## 2.9. cudlaWaitEvents Struct Reference

Wait events for [cudlaSubmitTask](#).

**uint32\_t cudlaWaitEvents::numEvents**

Total number of wait events.

```
const CudlaFence *cudlaWaitEvents::preFences
```

Array of fence pointers for all the wait events.

---

# Chapter 3. Data Fields

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

**deviceVersion**

[cudlaDevAttribute](#)

**devPtrs**

[cudlaSignalEvents](#)

**eofFences**

[cudlaSignalEvents](#)

**extBufObject**

[cudlaExternalMemoryHandleDesc](#)

**extSyncObject**

[cudlaExternalSemaphoreHandleDesc](#)

**fence**

[CudlaFence](#)

**inputTensor**

[cudlaTask](#)

**inputTensorDesc**

[cudlaModuleAttribute](#)

**moduleHandle**

[cudlaTask](#)

**numEvents**

[cudlaWaitEvents](#)

[cudlaSignalEvents](#)

**numInputTensors**

[cudlaTask](#)

[cudlaModuleAttribute](#)

**numOutputTensors**

[cudlaTask](#)

[cudlaModuleAttribute](#)

**outputTensor**

[cudlaTask](#)

**outputTensorDesc**

[cudlaModuleAttribute](#)

**preFences**[cudaWaitEvents](#)**signalEvents**[cudaTask](#)**size**[cudaExternalMemoryHandleDesc](#)**type**[CudlaFence](#)**unifiedAddressingSupported**[cudaDevAttribute](#)**waitEvents**[cudaTask](#)

## 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

© 2021-2024 NVIDIA Corporation & affiliates. All rights reserved.