



# OPTIX API REFERENCE

v3.9

**API Reference Manual**



## 1 Module Documentation

### 1.1 OpenGL Texture Formats

The following OpenGL texture formats are available for interoperability with OptiX.

R8I
R8UI
RG8I
RG8UI
RGBA8
RGBA8I
RGBA8UI
R16I
R16UI
RG16I
RG16UI
RGBA16
RGBA16I
RGBA16UI
R32I
R32UI
RG32I
RG32UI
RGBA32I
RGBA32UI
R32F
RG32F
RGBA32F

## 1.2 Direct3D Texture Formats

The following Direct3D texture formats are available for interoperability with OptiX.

L8
A8
A8L8
V8U8
A8R8G8B8
X8R8G8B8
A8B8G8R8
X8B8G8R8
Q8W8V8U8
L16
G16R16
V16U16
A16B16G16R16
Q16W16V16U16
R32F
G32R32F
A32B32G32R32F

### 1.3 DXGI Texture Formats

The following DXGI texture formats are available for interoperability with OptiX.

R8_SINT
R8_SNORM
R8_UINT
R8_UNORM
R16_SINT
R16_SNORM
R16_UINT
R16_UNORM
R32_SINT
R32_UINT
R32_FLOAT
R8G8_SINT
R8G8_SNORM
R8G8_UINT
R8G8_UNORM
R16G16_SINT
R16G16_SNORM
R16G16_UINT
R16G16_UNORM
R32G32_SINT
R32G32_UINT
R32G32_FLOAT
R8G8B8A8_SINT
R8G8B8A8_SNORM
R8G8B8A8_UINT
R8G8B8A8_UNORM
R16G16B16A16_SINT
R16G16B16A16_SNORM
R16G16B16A16_UINT
R16G16B16A16_UNORM
R32G32B32A32_SINT
R32G32B32A32_UINT
R32G32B32A32_FLOAT

## 2 Class Documentation

### 2.1 optix::Aabb Class Reference

#### 2.1.1 Detailed Description

Axis-aligned bounding box.

#### Description

[Aabb](#) is a utility class for computing and manipulating axis-aligned bounding boxes (aabbs). [Aabb](#) is primarily useful in the bounding box program associated with geometry objects. [Aabb](#) may also be useful in other computation and can be used in both host and device code.

#### History

[Aabb](#) was introduced in OptiX 1.0.

**See also** [RT\\_PROGRAM](#), [rtGeometrySetBoundingBoxProgram](#)

#### Public Member Functions

- RT\_HOSTDEVICE [Aabb](#) ()
- RT\_HOSTDEVICE [Aabb](#) (const float3 &min, const float3 &max)
- RT\_HOSTDEVICE [Aabb](#) (const float3 &v0, const float3 &v1, const float3 &v2)
- RT\_HOSTDEVICE bool [operator==](#) (const [Aabb](#) &other) const
- RT\_HOSTDEVICE float3 & [operator\[\]](#) (int i)
- RT\_HOSTDEVICE const float3 & [operator\[\]](#) (int i) const
- RT\_HOSTDEVICE void [set](#) (const float3 &min, const float3 &max)
- RT\_HOSTDEVICE void [set](#) (const float3 &v0, const float3 &v1, const float3 &v2)
- RT\_HOSTDEVICE void [invalidate](#) ()
- RT\_HOSTDEVICE bool [valid](#) () const
- RT\_HOSTDEVICE bool [contains](#) (const float3 &p) const
- RT\_HOSTDEVICE bool [contains](#) (const [Aabb](#) &bb) const
- RT\_HOSTDEVICE void [include](#) (const float3 &p)
- RT\_HOSTDEVICE void [include](#) (const [Aabb](#) &other)
- RT\_HOSTDEVICE void [include](#) (const float3 &min, const float3 &max)
- RT\_HOSTDEVICE float3 [center](#) () const
- RT\_HOSTDEVICE float [center](#) (int dim) const
- RT\_HOSTDEVICE float3 [extent](#) () const
- RT\_HOSTDEVICE float [extent](#) (int dim) const
- RT\_HOSTDEVICE float [volume](#) () const
- RT\_HOSTDEVICE float [area](#) () const
- RT\_HOSTDEVICE float [halfArea](#) () const
- RT\_HOSTDEVICE int [longestAxis](#) () const
- RT\_HOSTDEVICE float [maxExtent](#) () const
- RT\_HOSTDEVICE bool [intersects](#) (const [Aabb](#) &other) const
- RT\_HOSTDEVICE void [intersection](#) (const [Aabb](#) &other)
- RT\_HOSTDEVICE void [enlarge](#) (float amount)
- RT\_HOSTDEVICE bool [isFlat](#) () const
- RT\_HOSTDEVICE float [distance](#) (const float3 &x) const
- RT\_HOSTDEVICE float [distance2](#) (const float3 &x) const
- RT\_HOSTDEVICE float [signedDistance](#) (const float3 &x) const

#### Public Attributes

- float3 [m\\_min](#)
- float3 [m\\_max](#)

## 2.1.2 Constructor & Destructor Documentation

### 2.1.2.1 OPTIXU\_INLINE RT\_HOSTDEVICE optix::Aabb::Aabb ( )

Construct an invalid box

### 2.1.2.2 OPTIXU\_INLINE RT\_HOSTDEVICE optix::Aabb::Aabb ( const float3 & *min*, const float3 & *max* )

Construct from min and max vectors

### 2.1.2.3 OPTIXU\_INLINE RT\_HOSTDEVICE optix::Aabb::Aabb ( const float3 & *v0*, const float3 & *v1*, const float3 & *v2* )

Construct from three points (e.g. triangle)

## 2.1.3 Member Function Documentation

### 2.1.3.1 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::area ( ) const

Compute the surface area of the box

### 2.1.3.2 OPTIXU\_INLINE RT\_HOSTDEVICE float3 optix::Aabb::center ( ) const

Compute the box center

### 2.1.3.3 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::center ( int *dim* ) const

Compute the box center in the given dimension

### 2.1.3.4 OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::Aabb::contains ( const float3 & *p* ) const

Check if the point is in the box

### 2.1.3.5 OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::Aabb::contains ( const Aabb & *bb* ) const

Check if the box is fully contained in the box

### 2.1.3.6 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::distance ( const float3 & *x* ) const

Compute the minimum Euclidean distance from a point on the surface of this [Aabb](#) to the point of interest

### 2.1.3.7 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::distance2 ( const float3 & *x* ) const

Compute the minimum squared Euclidean distance from a point on the surface of this [Aabb](#) to the point of interest

### 2.1.3.8 OPTIXU\_INLINE RT\_HOSTDEVICE void optix::Aabb::enlarge ( float *amount* )

Enlarge the box by moving both min and max by 'amount'

### 2.1.3.9 OPTIXU\_INLINE RT\_HOSTDEVICE float3 optix::Aabb::extent ( ) const

Compute the box extent

### 2.1.3.10 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::extent ( int *dim* ) const

Compute the box extent in the given dimension

### 2.1.3.11 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::halfArea ( ) const

Compute half the surface area of the box

2.1.3.12 OPTIXU\_INLINE RT\_HOSTDEVICE void optix::Aabb::include ( const float3 & *p* )

Extend the box to include the given point

2.1.3.13 OPTIXU\_INLINE RT\_HOSTDEVICE void optix::Aabb::include ( const Aabb & *other* )

Extend the box to include the given box

2.1.3.14 OPTIXU\_INLINE RT\_HOSTDEVICE void optix::Aabb::include ( const float3 & *min*, const float3 & *max* )

Extend the box to include the given box

2.1.3.15 OPTIXU\_INLINE RT\_HOSTDEVICE void optix::Aabb::intersection ( const Aabb & *other* )

Make the current box be the intersection between this one and another one

2.1.3.16 OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::Aabb::intersects ( const Aabb & *other* ) const

Check for intersection with another box

2.1.3.17 OPTIXU\_INLINE RT\_HOSTDEVICE void optix::Aabb::invalidate ( )

Invalidate the box

2.1.3.18 OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::Aabb::isFlat ( ) const

Check if the box is flat in at least one dimension

2.1.3.19 OPTIXU\_INLINE RT\_HOSTDEVICE int optix::Aabb::longestAxis ( ) const

Get the index of the longest axis

2.1.3.20 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::maxExtent ( ) const

Get the extent of the longest axis

2.1.3.21 OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::Aabb::operator== ( const Aabb & *other* ) const

Exact equality

2.1.3.22 OPTIXU\_INLINE RT\_HOSTDEVICE float3 & optix::Aabb::operator[] ( int *i* )

Array access

2.1.3.23 OPTIXU\_INLINE RT\_HOSTDEVICE const float3 & optix::Aabb::operator[] ( int *i* ) const

Const array access

2.1.3.24 OPTIXU\_INLINE RT\_HOSTDEVICE void optix::Aabb::set ( const float3 & *min*, const float3 & *max* )

Set using two vectors

2.1.3.25 OPTIXU\_INLINE RT\_HOSTDEVICE void optix::Aabb::set ( const float3 & *v0*, const float3 & *v1*, const float3 & *v2* )

Set using three points (e.g. triangle)

2.1.3.26 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::signedDistance ( const float3 & *x* ) const

Compute the minimum Euclidean distance from a point on the surface of this [Aabb](#) to the point of interest. If the point of interest lies inside this [Aabb](#), the result is negative

## 2.1.3.27 OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::Aabb::valid ( ) const

Check if the box is valid

## 2.1.3.28 OPTIXU\_INLINE RT\_HOSTDEVICE float optix::Aabb::volume ( ) const

Compute the volume of the box

## 2.1.4 Member Data Documentation

## 2.1.4.1 float3 optix::Aabb::m\_max

Max bound

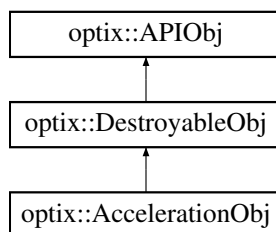
## 2.1.4.2 float3 optix::Aabb::m\_min

Min bound

## 2.2 optix::AccelerationObj Class Reference

## 2.2.1 Detailed Description

Acceleration wraps the OptiX C API RTacceleration opaque type and its associated function set. Inheritance diagram for optix::AccelerationObj:



## Public Member Functions

- void [destroy](#) ( )
- void [validate](#) ( )
- [Context](#) [getContext](#) ( ) const
- [RTacceleration](#) [get](#) ( )
  
- void [markDirty](#) ( )
- bool [isDirty](#) ( ) const
  
- void [setProperty](#) (const std::string &name, const std::string &value)
- std::string [getProperty](#) (const std::string &name) const
- void [setBuilder](#) (const std::string &builder)
- std::string [getBuilder](#) ( ) const
- void [setTraverser](#) (const std::string &traverser)
- std::string [getTraverser](#) ( ) const
  
- RTsize [getDataSize](#) ( ) const
- void [getData](#) (void \*data) const
- void [setData](#) (const void \*data, RTsize size)



## Friends

- class **Handle**< **AccelerationObj** >

## Additional Inherited Members

## 2.2.2 Member Function Documentation

## 2.2.2.1 RTsize optix::AccelerationObj::getDataSize ( ) const [inline]

Query the size of the marshalled acceleration data. See [rtAccelerationGetDataSize](#).

## 2.2.2.2 std::string optix::AccelerationObj::getProperty ( const std::string &amp; name ) const [inline]

Query properties specifying Acceleration builder/traverser behavior. See [rtAccelerationGetProperty](#).

## 2.2.2.3 void optix::AccelerationObj::markDirty ( ) [inline]

Mark the acceleration as needing a rebuild. See [rtAccelerationMarkDirty](#).

## 2.2.2.4 void optix::AccelerationObj::setProperty ( const std::string &amp; name, const std::string &amp; value ) [inline]

Set properties specifying Acceleration builder/traverser behavior. See [rtAccelerationSetProperty](#).

## 2.3 optix::APIObj Class Reference

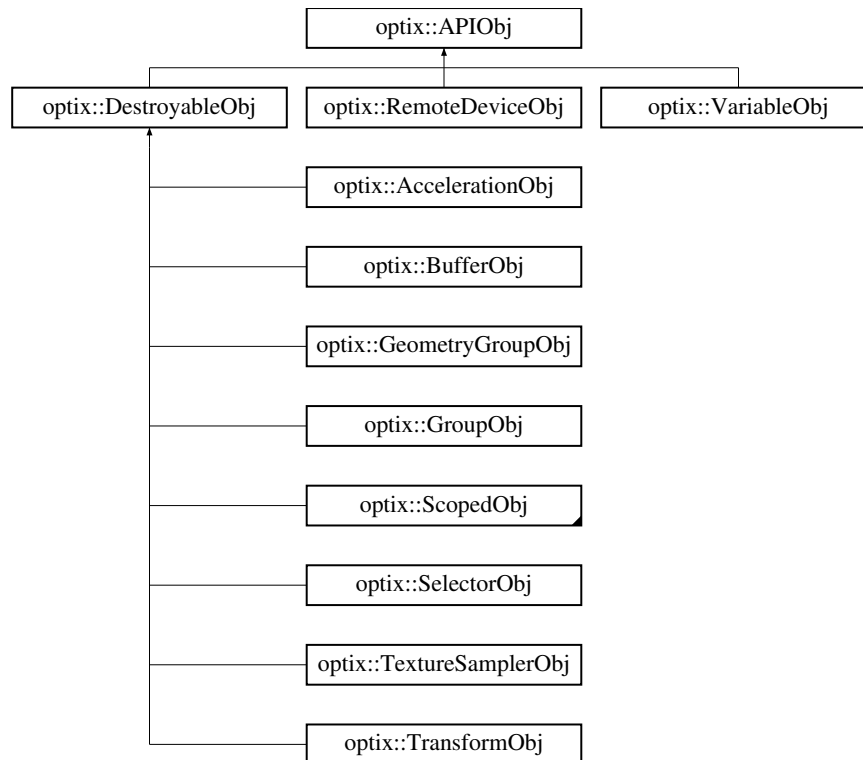
## 2.3.1 Detailed Description

Base class for all reference counted wrappers around OptiX C API opaque types.

Wraps:

- RTcontext
- RTbuffer
- RTgeometry
- RTgeometryinstance
- RTgeometrygroup
- RTgroup
- RTmaterial
- RTprogram
- RTselector
- RTtexturesampler
- RTtransform
- RTvariable

Inheritance diagram for optix::APIObj:



#### Public Member Functions

- void [addReference](#) ()
- int [removeReference](#) ()
- virtual [Context](#) [getContext](#) () const =0
- virtual void [checkError](#) ([RResult](#) code) const
- virtual void **checkError** ([RResult](#) code, [Context](#) context) const
- void **checkErrorNoGetContext** ([RResult](#) code) const

#### Static Public Member Functions

- static [Exception](#) [makeException](#) ([RResult](#) code, [RTcontext](#) context)

### 2.3.2 Member Function Documentation

#### 2.3.2.1 void optix::APIObj::checkError ( [RResult](#) code ) const [inline],[virtual]

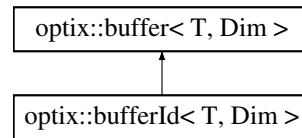
Check the given result code and throw an error with appropriate message if the code is not RTsuccess

Reimplemented in [optix::ContextObj](#).

## 2.4 optix::boundCallableProgramId< T > Class Template Reference

## 2.5 optix::buffer< T, Dim > Struct Template Reference

Inheritance diagram for optix::buffer< T, Dim >:



## Classes

- struct [type](#)

## Public Types

- typedef [VectorTypes](#)< size\_t, Dim > **WrapperType**
- typedef [VectorTypes](#)< size\_t, Dim >::Type **IndexType**

## Public Member Functions

- `__device__ __forceinline__ IndexType size () const`
- `__device__ __forceinline__ T & operator[] (IndexType i)`

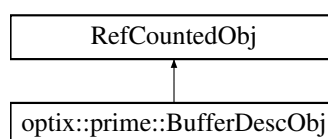
## Static Protected Member Functions

- `__inline__ static __device__ size_t4 make_index (size_t v0)`
- `__inline__ static __device__ size_t4 make_index (size_t2 v0)`
- `__inline__ static __device__ size_t4 make_index (size_t3 v0)`
- `__inline__ static __device__ size_t4 make_index (size_t4 v0)`
- `template<typename T2 > __device__ static __forceinline__ void * create (type< T2 >, void *v)`
- `template<typename T2, int Dim2> __device__ static __forceinline__ void * create (type< bufferId< T2, Dim2 > >, void *v)`

## 2.6 optix::prime::BufferDescObj Class Reference

### 2.6.1 Detailed Description

Encapsulates an OptiX Prime buffer descriptor. The purpose of a buffer descriptor is to provide information about a buffer's type, format, and location. It also describes the region of the buffer to use. Inheritance diagram for optix::prime::BufferDescObj:



## Public Member Functions

- [Context](#) `getContext ()`
- void `setRange` (RTPsize begin, RTPsize end)
- void `setStride` (unsigned strideBytes)
- void `setCudaDeviceNumber` (unsigned deviceNumber)
- [RTPbufferdesc](#) `getRTPbufferdesc ()`

## Friends

- class **ContextObj**
- class **ModelObj**
- class **QueryObj**

## 2.7 optix::bufferId&lt; T, Dim &gt; Struct Template Reference

## 2.7.1 Detailed Description

`template<typename T, int Dim>struct optix::bufferId< T, Dim >`

`bufferId` is a host version of the device side `bufferId`.

Use `bufferId` to define types that can be included from both the host and device code. This class provides a container that can be used to transport the buffer id back and forth between host and device code. The `bufferId` class is useful, because it can take a buffer id obtained from `rtBufferGetId` and provide accessors similar to the `buffer` class.

"bindless\_type.h" used by both host and device code:

```
#include <optix_world.h>
struct BufInfo {
    int val;
    rtBufferId<int, 1> data;
};
```

Host code:

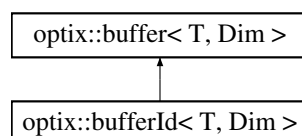
```
#include "bindless_type.h"
BufInfo input_buffer_info;
input_buffer_info.val = 0;
input_buffer_info.data = rtBufferId<int,1>(inputBuffer0->getId());
context["input_buffer_info"]->setUserData(sizeof(BufInfo), &input_buffer_info);
```

Device code:

```
#include "bindless_type.h"
rtBuffer<int,1> result;
rtDeclareVariable(BufInfo, input_buffer_info, ,);

RT_PROGRAM void bindless()
{
    int value = input_buffer_info.data[input_buffer_info.val];
    result[0] = value;
}
```

Inheritance diagram for `optix::bufferId< T, Dim >`:



## Public Types

- typedef `buffer`< T, Dim >  
::`WrapperType` **WrapperType**
- typedef `buffer`< T, Dim >::`IndexType` **IndexType**

## Public Member Functions

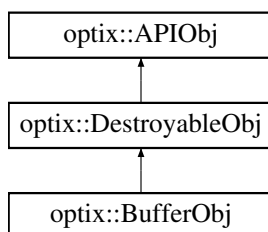
- `__device__ __forceinline__` **bufferId** (`RTbufferidnull` nullid)
- `__device__ __forceinline__` **bufferId** (int id)
- `__device__ __forceinline__`  
**bufferId** & **operator=** (`RTbufferidnull` nullid)
- `__device__ __forceinline__`  
`IndexType` **size** () const
- `__device__ __forceinline__` T & **operator[]** (`IndexType` i) const
- `__device__ __forceinline__` int **getId** () const
- `__device__ __forceinline__` **operator bool** () const
- **bufferId** (int id)
- int **getId** () const

## Additional Inherited Members

## 2.8 optix::BufferObj Class Reference

## 2.8.1 Detailed Description

Buffer wraps the OptiX C API `RTbuffer` opaque type and its associated function set. Inheritance diagram for `optix::BufferObj`:



## Public Member Functions

- void **destroy** ()
- void **validate** ()
- `Context` **getContext** () const
- `RTbuffer` **get** ()
- void **setFormat** (`RTformat` format)
- `RTformat` **getFormat** () const
- void **setElementSize** (`RTsize` size\_of\_element)
- `RTsize` **getElementSize** () const
- void **getDevicePointer** (unsigned int optix\_device\_number, `CUdeviceptr` \*device\_pointer)
- `CUdeviceptr` **getDevicePointer** (unsigned int optix\_device\_number)
- void **setDevicePointer** (unsigned int optix\_device\_number, `CUdeviceptr` device\_pointer)
- void **markDirty** ()
- void **setSize** (`RTsize` width)

- void [getSize](#) (RTsize &width) const
- void [getMipLevelSize](#) (unsigned int level, RTsize &width) const
- void [setSize](#) (RTsize width, RTsize height)
- void [getSize](#) (RTsize &width, RTsize &height) const
- void [getMipLevelSize](#) (unsigned int level, RTsize &width, RTsize &height) const
- void [setSize](#) (RTsize width, RTsize height, RTsize depth)
- void [getSize](#) (RTsize &width, RTsize &height, RTsize &depth) const
- void [getMipLevelSize](#) (unsigned int level, RTsize &width, RTsize &height, RTsize &depth) const
- void [setSize](#) (unsigned int dimensionality, const RTsize \*dims)
- void [getSize](#) (unsigned int dimensionality, RTsize \*dims) const
- unsigned int [getDimensionality](#) () const
- void [setMipLevelCount](#) (unsigned int levels)
- unsigned int [getMipLevelCount](#) () const
  
- int [getId](#) () const
  
- unsigned int [getGLBOld](#) () const
- void [registerGLBuffer](#) ()
- void [unregisterGLBuffer](#) ()
  
- void [setAttribute](#) (RTbufferattribute attrib, RTsize size, void \*p)
- void [getAttribute](#) (RTbufferattribute attrib, RTsize size, void \*p)
  
- void [registerD3D9Buffer](#) ()
- void [registerD3D10Buffer](#) ()
- void [registerD3D11Buffer](#) ()
- void [unregisterD3D9Buffer](#) ()
- void [unregisterD3D10Buffer](#) ()
- void [unregisterD3D11Buffer](#) ()
- IDirect3DResource9 \* [getD3D9Resource](#) ()
- ID3D10Resource \* [getD3D10Resource](#) ()
- ID3D11Resource \* [getD3D11Resource](#) ()
  
- void \* [map](#) ()
- void [unmap](#) ()
- void \* [map](#) (unsigned int level)
- void [unmap](#) (unsigned int level)
  
- void [bindProgressiveStream](#) (Buffer source)
- void [getProgressiveUpdateReady](#) (int \*ready, unsigned int \*subframe\_count, unsigned int \*max\_subframes)

#### Friends

- class **Handle**< **BufferObj** >

#### Additional Inherited Members

### 2.8.2 Member Function Documentation

#### 2.8.2.1 void optix::BufferObj::bindProgressiveStream ( Buffer source ) [inline]

Bind a buffer as source for a progressive stream. See [rtBufferBindProgressiveStream](#).

2.8.2.2 `CUdeviceptr optix::BufferObj::getDevicePointer ( unsigned int optix_device_number ) [inline]`

Set the data format for the buffer. See [rtBufferSetFormat](#).

2.8.2.3 `unsigned int optix::BufferObj::getGLBOld ( ) const [inline]`

Queries the OpenGL Buffer Object ID associated with this buffer. See [rtBufferGetGLBOld](#).

2.8.2.4 `int optix::BufferObj::getId ( ) const [inline]`

Queries an id suitable for referencing the buffer in an another buffer. See [rtBufferGetId](#).

2.8.2.5 `void * optix::BufferObj::map ( ) [inline]`

Maps a buffer object for host access. See [rtBufferMap](#).

2.8.2.6 `void optix::BufferObj::registerD3D9Buffer ( ) [inline]`

Declare the texture's buffer as mutable and inaccessible by OptiX. See [rtBufferD3D9Register](#).

2.8.2.7 `void optix::BufferObj::setAttribute ( RTbufferattribute attrib, RTsize size, void * p ) [inline]`

Set a Buffer Attribute. See [rtBufferSetAttribute](#).

2.8.2.8 `void optix::BufferObj::setFormat ( RTformat format ) [inline]`

Set the data format for the buffer. See [rtBufferSetFormat](#).

2.8.2.9 `void optix::BufferObj::setSize ( RTsize width, RTsize height, RTsize depth ) [inline]`

Set buffer dimensionality to three and buffer dimensions to specified width,height,depth. See [rtBufferSetSize3D](#).

## 2.9 RTPinternals\_3070::BvhNode Struct Reference

### Public Attributes

- float **bbmin0** [3]
- float **bbmax0** [3]
- int **index0** [2]
- float **bbmin1** [3]
- float **bbmax1** [3]
- int **index1** [2]

## 2.10 optix::callableProgramId< T > Class Template Reference

## 2.11 rti\_internal\_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T > Class Template Reference

### Public Member Functions

- `__device__ __forceinline__ ReturnT operator() ( )`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4)`

- `__device__ __forceinline__ ReturnT operator()` (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5)
- `__device__ __forceinline__ ReturnT operator()` (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5, Arg6T arg6)
- `__device__ __forceinline__ ReturnT operator()` (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5, Arg6T arg6, Arg7T arg7)
- `__device__ __forceinline__ ReturnT operator()` (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5, Arg6T arg6, Arg7T arg7, Arg8T arg8)
- `__device__ __forceinline__ ReturnT operator()` (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5, Arg6T arg6, Arg7T arg7, Arg8T arg8, Arg9T arg9)

## Protected Attributes

- `int m_id`

## 2.12 rti\_internal\_callableprogram::check\_is\_CPArgVoid&lt; Condition, Dummy &gt; Struct Template Reference

## Public Types

- `typedef bool result`

## 2.13 rti\_internal\_callableprogram::check\_is\_CPArgVoid&lt; false, IntentionalError &gt; Struct Template Reference

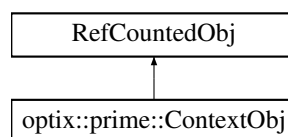
## Public Types

- `typedef IntentionalError::does_not_exist result`

## 2.14 optix::prime::ContextObj Class Reference

## 2.14.1 Detailed Description

Wraps the OptiX Prime C API [RTPcontext](#) opaque type and its associated function set representing an OptiX Prime context. Inheritance diagram for `optix::prime::ContextObj`:



## Public Member Functions

- [BufferDesc createBufferDesc](#) ([RTPbufferformat](#) format, [RTPbuffertype](#) type, void \*buffer)
- [Model createModel](#) ()
- void [setCudaDeviceNumbers](#) (const std::vector< unsigned > &deviceNumbers)
- void [setCudaDeviceNumbers](#) (unsigned deviceCount, const unsigned \*deviceNumbers)
- void [setCpuThreads](#) (unsigned numThreads)
- std::string [getLastErrorMessage](#) ()
- [RTPcontext getRTPcontext](#) ()



## Static Public Member Functions

- static [Context create](#) ([RTPcontexttype](#) type)

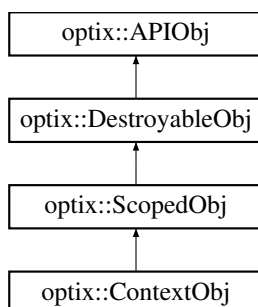
## Friends

- class **QueryObj**
- class **ModelObj**
- class **BufferDescObj**

## 2.15 optix::ContextObj Class Reference

## 2.15.1 Detailed Description

Context object wraps the OptiX C API RTcontext opaque type and its associated function set. Inheritance diagram for optix::ContextObj:



## Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context getContext](#) () const
- void [compile](#) ()
- void [setRemoteDevice](#) ([RemoteDevice](#) remote\_device)
- int [getRunningState](#) () const
- [RTcontext get](#) ()
- void [checkError](#) ([RTresult](#) code) const
- std::string [getErrorString](#) ([RTresult](#) code) const
- [Acceleration createAcceleration](#) (const char \*builder, const char \*traverser)
- [Buffer createBuffer](#) (unsigned int type)
- [Buffer createBuffer](#) (unsigned int type, [RTformat](#) format)
- [Buffer createBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width)
- [Buffer createMipmappedBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, unsigned int levels)
- [Buffer createBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height)
- [Buffer createMipmappedBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height, unsigned int levels)
- [Buffer createBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height, RTsize depth)
- [Buffer createMipmappedBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height, RTsize depth, unsigned int levels)
- [Buffer create1DLayeredBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize layers, unsigned int levels)

- [Buffer create2DLayeredBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height, RTsize layers, unsigned int levels)
- [Buffer createCubeBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height, unsigned int levels)
- [Buffer createCubeLayeredBuffer](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height, RTsize faces, unsigned int levels)
- [Buffer createBufferForCUDA](#) (unsigned int type)
- [Buffer createBufferForCUDA](#) (unsigned int type, [RTformat](#) format)
- [Buffer createBufferForCUDA](#) (unsigned int type, [RTformat](#) format, RTsize width)
- [Buffer createBufferForCUDA](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height)
- [Buffer createBufferForCUDA](#) (unsigned int type, [RTformat](#) format, RTsize width, RTsize height, RTsize depth)
- [Buffer createBufferFromGLBO](#) (unsigned int type, unsigned int vbo)
- [TextureSampler createTextureSamplerFromGLImage](#) (unsigned int id, [RTgltarget](#) target)
- [Buffer createBufferFromD3D9Resource](#) (unsigned int type, [IDirect3DResource9](#) \*pResource)
- [Buffer createBufferFromD3D10Resource](#) (unsigned int type, [ID3D10Resource](#) \*pResource)
- [Buffer createBufferFromD3D11Resource](#) (unsigned int type, [ID3D11Resource](#) \*pResource)
- [TextureSampler createTextureSamplerFromD3D9Resource](#) ([IDirect3DResource9](#) \*pResource)
- [TextureSampler createTextureSamplerFromD3D10Resource](#) ([ID3D10Resource](#) \*pResource)
- [TextureSampler createTextureSamplerFromD3D11Resource](#) ([ID3D11Resource](#) \*pResource)
- [Buffer getBufferFromId](#) (int buffer\_id)
- [Program getProgramFromId](#) (int program\_id)
- [TextureSampler getTextureSamplerFromId](#) (int sampler\_id)
- [Geometry createGeometry](#) ()
- [GeometryInstance createGeometryInstance](#) ()
- `template<class Iterator >`  
[GeometryInstance createGeometryInstance](#) ([Geometry](#) geometry, Iterator matlbegin, Iterator matlend)
- [Group createGroup](#) ()
- `template<class Iterator >`  
[Group createGroup](#) (Iterator childbegin, Iterator childend)
- [GeometryGroup createGeometryGroup](#) ()
- `template<class Iterator >`  
[GeometryGroup createGeometryGroup](#) (Iterator childbegin, Iterator childend)
- [Transform createTransform](#) ()
- [Material createMaterial](#) ()
- [Program createProgramFromPTXFile](#) (const std::string &ptx, const std::string &program\_name)
- [Program createProgramFromPTXString](#) (const std::string &ptx, const std::string &program\_name)
- [Selector createSelector](#) ()
- [TextureSampler createTextureSampler](#) ()
  
- `template<class Iterator >`  
void [setDevices](#) (Iterator begin, Iterator end)
- void [setD3D9Device](#) ([IDirect3DDevice9](#) \*device)
- void [setD3D10Device](#) ([ID3D10Device](#) \*device)
- void [setD3D11Device](#) ([ID3D11Device](#) \*device)
- std::vector< int > [getEnabledDevices](#) () const
- unsigned int [getEnabledDeviceCount](#) () const
  
- int [getMaxTextureCount](#) () const
- int [getCPUNumThreads](#) () const
- RTsize [getUsedHostMemory](#) () const
- int [getGPUPagingActive](#) () const
- int [getGPUPagingForcedOff](#) () const
- RTsize [getAvailableDeviceMemory](#) (int ordinal) const
  
- void [setCPUNumThreads](#) (int cpu\_num\_threads)

- void [setGPUPagingForcedOff](#) (int gpu\_paging\_forced\_off)
- template<class T >  
void [setAttribute](#) (RTcontextattribute attribute, const T &val)
- void [setStackSize](#) (RTsize stack\_size\_bytes)
- RTsize [getStackSize](#) () const
- void [setTimeoutCallback](#) (RTtimeoutcallback callback, double min\_polling\_seconds)
- void [setEntryPointCount](#) (unsigned int num\_entry\_points)
- unsigned int [getEntryPointCount](#) () const
- void [setRayTypeCount](#) (unsigned int num\_ray\_types)
- unsigned int [getRayTypeCount](#) () const
- void [setRayGenerationProgram](#) (unsigned int entry\_point\_index, [Program](#) program)
- [Program](#) [getRayGenerationProgram](#) (unsigned int entry\_point\_index) const
- void [setExceptionProgram](#) (unsigned int entry\_point\_index, [Program](#) program)
- [Program](#) [getExceptionProgram](#) (unsigned int entry\_point\_index) const
- void [setExceptionEnabled](#) (RTexception exception, bool enabled)
- bool [getExceptionEnabled](#) (RTexception exception) const
- void [setMissProgram](#) (unsigned int ray\_type\_index, [Program](#) program)
- [Program](#) [getMissProgram](#) (unsigned int ray\_type\_index) const
- void [launch](#) (unsigned int entry\_point\_index, RTsize image\_width)
- void [launch](#) (unsigned int entry\_point\_index, RTsize image\_width, RTsize image\_height)
- void [launch](#) (unsigned int entry\_point\_index, RTsize image\_width, RTsize image\_height, RTsize image\_depth)
- void [launchProgressive](#) (unsigned int entry\_point\_index, RTsize image\_width, RTsize image\_height, unsigned int max\_subframes)
- void [stopProgressive](#) ()
- void [setPrintEnabled](#) (bool enabled)
- bool [getPrintEnabled](#) () const
- void [setPrintBufferSize](#) (RTsize buffer\_size\_bytes)
- RTsize [getPrintBufferSize](#) () const
- void [setPrintLaunchIndex](#) (int x, int y=-1, int z=-1)
- optix::int3 [getPrintLaunchIndex](#) () const
- [Variable](#) [declareVariable](#) (const std::string &name)
- [Variable](#) [queryVariable](#) (const std::string &name) const
- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const

#### Static Public Member Functions

- static unsigned int [getDeviceCount](#) ()
- static std::string [getDeviceName](#) (int ordinal)
- static void [getDeviceAttribute](#) (int ordinal, RTdeviceattribute attrib, RTsize size, void \*p)
- static [Context](#) [create](#) ()

#### Friends

- class **Handle**< **ContextObj** >

## 2.15.2 Member Function Documentation

**2.15.2.1** `void optix::ContextObj::checkError ( RTresult code ) const` `[inline], [virtual]`

See [APIObj::checkError](#)

Reimplemented from [optix::APIObj](#).

**2.15.2.2** `Buffer optix::ContextObj::create1DLayeredBuffer ( unsigned int type, RTformat format, RTsize width, RTsize layers, unsigned int levels )` `[inline]`

Create a 1D layered mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

**2.15.2.3** `Buffer optix::ContextObj::create2DLayeredBuffer ( unsigned int type, RTformat format, RTsize width, RTsize height, RTsize layers, unsigned int levels )` `[inline]`

Create a 2D layered mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

**2.15.2.4** `Acceleration optix::ContextObj::createAcceleration ( const char * builder, const char * traverser )` `[inline]`

See [rtAccelerationCreate](#)

**2.15.2.5** `Buffer optix::ContextObj::createBuffer ( unsigned int type, RTformat format, RTsize width )` `[inline]`

Create a buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize1D](#).

**2.15.2.6** `Buffer optix::ContextObj::createBuffer ( unsigned int type, RTformat format, RTsize width, RTsize height )` `[inline]`

Create a buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize2D](#).

**2.15.2.7** `Buffer optix::ContextObj::createBuffer ( unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth )` `[inline]`

Create a buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize3D](#).

**2.15.2.8** `Buffer optix::ContextObj::createBufferForCUDA ( unsigned int type, RTformat format, RTsize width )` `[inline]`

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize1D](#).

**2.15.2.9** `Buffer optix::ContextObj::createBufferForCUDA ( unsigned int type, RTformat format, RTsize width, RTsize height )` `[inline]`

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize2D](#).

**2.15.2.10** `Buffer optix::ContextObj::createBufferForCUDA ( unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth )` `[inline]`

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize3D](#).

**2.15.2.11 Buffer** `optix::ContextObj::createCubeBuffer ( unsigned int type, RTformat format, RTsize width, RTsize height, unsigned int levels ) [inline]`

Create a cube mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

**2.15.2.12 Buffer** `optix::ContextObj::createCubeLayeredBuffer ( unsigned int type, RTformat format, RTsize width, RTsize height, RTsize faces, unsigned int levels ) [inline]`

Create a cube layered mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

**2.15.2.13 template<class Iterator > GeometryGroup** `optix::ContextObj::createGeometryGroup ( Iterator childbegin, Iterator childend ) [inline]`

Create a GeometryGroup with a set of child nodes. See [rtGeometryGroupCreate](#), [rtGeometryGroupSetChildCount](#) and [rtGeometryGroupSetChild](#)

**2.15.2.14 template<class Iterator > GeometryInstance** `optix::ContextObj::createGeometryInstance ( Geometry geometry, Iterator matlbegin, Iterator matlend )`

Create a geometry instance with a Geometry object and a set of associated materials. See [rtGeometryInstanceCreate](#), [rtGeometryInstanceSetMaterialCount](#), and [rtGeometryInstanceSetMaterial](#)

**2.15.2.15 template<class Iterator > Group** `optix::ContextObj::createGroup ( Iterator childbegin, Iterator childend ) [inline]`

Create a Group with a set of child nodes. See [rtGroupCreate](#), [rtGroupSetChildCount](#) and [rtGroupSetChild](#)

**2.15.2.16 Buffer** `optix::ContextObj::createMipmappedBuffer ( unsigned int type, RTformat format, RTsize width, unsigned int levels ) [inline]`

Create a mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize1DMipmapped](#).

**2.15.2.17 Buffer** `optix::ContextObj::createMipmappedBuffer ( unsigned int type, RTformat format, RTsize width, RTsize height, unsigned int levels ) [inline]`

Create a mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize2DMipmapped](#).

**2.15.2.18 Buffer** `optix::ContextObj::createMipmappedBuffer ( unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth, unsigned int levels ) [inline]`

Create a mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize3DMipmapped](#).

**2.15.2.19 Variable** `optix::ContextObj::declareVariable ( const std::string & name ) [inline], [virtual]`

Declare a variable associated with this object. See [rt\[ObjectType\]DeclareVariable](#). Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implements [optix::ScopedObj](#).

**2.15.2.20 Buffer** `optix::ContextObj::getBufferFromId ( int buffer_id ) [inline]`

Queries the Buffer object from a given buffer id obtained from a previous call to [BufferObj::getId](#). See [BufferObj::getId](#) and [rtContextGetBufferFromId](#).

**2.15.2.21 Context** `optix::ContextObj::getContext ( ) const [inline], [virtual]`

Retrieve the Context object associated with this APIObject. In this case, simply returns itself.

Implements [optix::APIObj](#).

2.15.2.22 `unsigned int optix::ContextObj::getEnabledDeviceCount ( ) const [inline]`

See [rtContextGetDeviceCount](#). As opposed to `getDeviceCount`, this returns only the number of enabled devices.

2.15.2.23 `int optix::ContextObj::getMaxTextureCount ( ) const [inline]`

See [rtContextGetAttribute](#)

2.15.2.24 `Program optix::ContextObj::getProgramFromId ( int program_id ) [inline]`

Queries the Program object from a given program id obtained from a previous call to [ProgramObj::getId](#). See [ProgramObj::getId](#) and [rtContextGetProgramFromId](#).

2.15.2.25 `TextureSampler optix::ContextObj::getTextureSamplerFromId ( int sampler_id ) [inline]`

Queries the TextureSampler object from a given sampler id obtained from a previous call to [TextureSamplerObj::getId](#). See [TextureSamplerObj::getId](#) and [rtContextGetTextureSamplerFromId](#).

2.15.2.26 `unsigned int optix::ContextObj::getVariableCount ( ) const [inline],[virtual]`

Query the number of variables associated with this object. Used along with [ScopedObj::getVariable](#) to iterate over variables in an object. See [rt\[ObjectType\]GetVariableCount](#)

Implements [optix::ScopedObj](#).

2.15.2.27 `void optix::ContextObj::launch ( unsigned int entry_point_index, RTsize image_width ) [inline]`

See [rtContextLaunch](#)

2.15.2.28 `void optix::ContextObj::launchProgressive ( unsigned int entry_point_index, RTsize image_width, RTsize image_height, unsigned int max_subframes ) [inline]`

See [rtContextLaunchProgressive](#)

2.15.2.29 `Variable optix::ContextObj::queryVariable ( const std::string & name ) const [inline],[virtual]`

Query a variable associated with this object by name. See [rt\[ObjectType\]QueryVariable](#). Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implements [optix::ScopedObj](#).

2.15.2.30 `void optix::ContextObj::setCPUNumThreads ( int cpu_num_threads ) [inline]`

See [rtContextSetAttribute](#)

2.15.2.31 `template<class Iterator > void optix::ContextObj::setDevices ( Iterator begin, Iterator end ) [inline]`

See [rtContextSetDevices](#)

2.15.2.32 `void optix::ContextObj::setPrintEnabled ( bool enabled ) [inline]`

See [rtContextSetPrintEnabled](#)

2.15.2.33 `void optix::ContextObj::setRayGenerationProgram ( unsigned int entry_point_index, Program program ) [inline]`

See [rtContextSetRayGenerationProgram](#)

2.15.2.34 `void optix::ContextObj::setStackSize ( RTsize stack_size_bytes ) [inline]`

See [rtContextSetStackSize](#)

2.15.2.35 void optix::ContextObj::setTimeoutCallback ( RTtimeoutcallback *callback*, double *min\_polling\_seconds* )  
[inline]

See [rtContextSetTimeoutCallback](#) RTtimeoutcallback is defined as typedef int (\*RTtimeoutcallback)(void).

## 2.16 rti\_internal\_callableprogram::CPArgVoid Class Reference

## 2.17 optix::DestroyableObj Class Reference

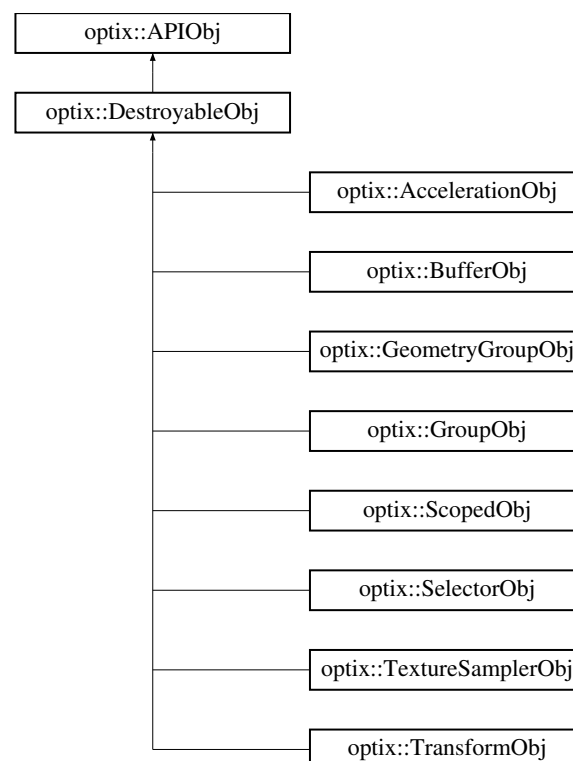
### 2.17.1 Detailed Description

Base class for all wrapper objects which can be destroyed and validated.

Wraps:

- RTcontext
- RTgeometry
- RTgeometryinstance
- RTgeometrygroup
- RTgroup
- RTmaterial
- RTprogram
- RTselector
- RTtexturesampler
- RTtransform

Inheritance diagram for optix::DestroyableObj:



## Public Member Functions

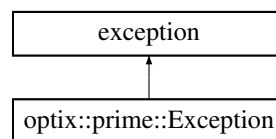
- virtual void [destroy](#) ()=0
- virtual void [validate](#) ()=0

## Additional Inherited Members

## 2.18 optix::prime::Exception Class Reference

## 2.18.1 Detailed Description

Encapsulates an OptiX Prime exception. Inheritance diagram for optix::prime::Exception:



## Public Member Functions

- [RTPresult](#) [getErrorCode](#) () const
- const std::string & [getErrorString](#) () const
- virtual const char \* **what** () const throw ()

## Static Public Member Functions

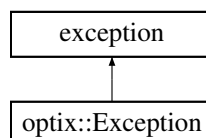
- static [Exception](#) [makeException](#) ([RTPresult](#) code)
- static [Exception](#) [makeException](#) ([RTPresult](#) code, [RTPresultContext](#) context)

## 2.19 optix::Exception Class Reference

## 2.19.1 Detailed Description

[Exception](#) class for error reporting from the OptiXpp API.

Encapsulates an error message, often the direct result of a failed OptiX C API function call and subsequent `rt-ContextGetErrorString` call. Inheritance diagram for `optix::Exception`:



## Public Member Functions

- [Exception](#) (const std::string &message, [RTPresult](#) error\_code=[RT\\_ERROR\\_UNKNOWN](#))
- virtual [~Exception](#) () throw ()
- const std::string & [getErrorString](#) () const
- [RTPresult](#) [getErrorCode](#) () const
- virtual const char \* **what** () const throw ()



### Static Public Member Functions

- static [Exception](#) [makeException](#) ([RTresult](#) code, [RTcontext](#) context)

### 2.19.2 Constructor & Destructor Documentation

#### 2.19.2.1 virtual optix::Exception::~~Exception ( ) throw ) [inline],[virtual]

Virtual destructor (needed for virtual function calls inherited from std::exception).

### 2.19.3 Member Function Documentation

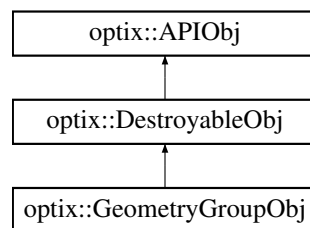
#### 2.19.3.1 Exception optix::Exception::makeException ( RTresult code, RTcontext context ) [inline],[static]

Helper for creating exceptions from an RTresult code origination from an OptiX C API function call.

## 2.20 optix::GeometryGroupObj Class Reference

### 2.20.1 Detailed Description

GeometryGroup wraps the OptiX C API RTgeometrygroup opaque type and its associated function set. Inheritance diagram for optix::GeometryGroupObj:



### Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTgeometrygroup](#) [get](#) ()
- void [setAcceleration](#) ([Acceleration](#) acceleration)
- [Acceleration](#) [getAcceleration](#) () const
- void [setChildCount](#) (unsigned int count)
- unsigned int [getChildCount](#) () const
- void [setChild](#) (unsigned int index, [GeometryInstance](#) geometryinstance)
- [GeometryInstance](#) [getChild](#) (unsigned int index) const
- unsigned int [addChild](#) ([GeometryInstance](#) child)
- unsigned int [removeChild](#) ([GeometryInstance](#) child)
- void [removeChild](#) (int index)
- void [removeChild](#) (unsigned int index)
- unsigned int [getChildIndex](#) ([GeometryInstance](#) child) const

## Friends

- class **Handle**< **GeometryGroupObj** >

## Additional Inherited Members

## 2.20.2 Member Function Documentation

2.20.2.1 unsigned int optix::GeometryGroupObj::removeChild ( **GeometryInstance** *child* ) [inline]

Remove a child in this group. Note: this function is not order-preserving. Returns the position of the removed element if succeeded. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid.

2.20.2.2 void optix::GeometryGroupObj::removeChild ( int *index* ) [inline]

Remove a child in this group. Note: this function is not order-preserving. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid.

2.20.2.3 void optix::GeometryGroupObj::removeChild ( unsigned int *index* ) [inline]

Remove a child in this group. Note: this function is not order-preserving. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid.

2.20.2.4 void optix::GeometryGroupObj::setAcceleration ( **Acceleration** *acceleration* ) [inline]

Set the Acceleration structure for this group. See [rtGeometryGroupSetAcceleration](#).

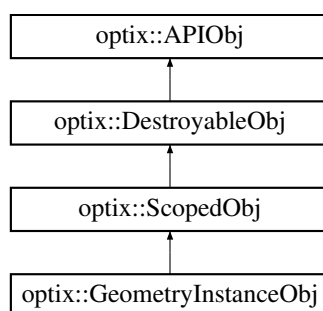
2.20.2.5 void optix::GeometryGroupObj::setChildCount ( unsigned int *count* ) [inline]

Set the number of children for this group. See [rtGeometryGroupSetChildCount](#).

## 2.21 optix::GeometryInstanceObj Class Reference

## 2.21.1 Detailed Description

GeometryInstance wraps the OptiX C API RTgeometryinstance acceleration opaque type and its associated function set. Inheritance diagram for optix::GeometryInstanceObj:



## Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTgeometryinstance](#) [get](#) ()

- void [setGeometry](#) ([Geometry](#) geometry)
- [Geometry](#) [getGeometry](#) () const
- void [setMaterialCount](#) (unsigned int count)
- unsigned int [getMaterialCount](#) () const
- void [setMaterial](#) (unsigned int idx, [Material](#) material)
- [Material](#) [getMaterial](#) (unsigned int idx) const
- unsigned int [addMaterial](#) ([Material](#) material)
- [Variable](#) [declareVariable](#) (const std::string &name)
- [Variable](#) [queryVariable](#) (const std::string &name) const
- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const

#### Friends

- class **Handle**< **GeometryInstanceObj** >

#### Additional Inherited Members

#### 2.21.2 Member Function Documentation

##### 2.21.2.1 Variable `optix::GeometryInstanceObj::declareVariable ( const std::string & name )` `[inline]`, `[virtual]`

Declare a variable associated with this object. See `rt[ObjectType]DeclareVariable`. Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implements [optix::ScopedObj](#).

##### 2.21.2.2 unsigned int `optix::GeometryInstanceObj::getVariableCount ( )` const `[inline]`, `[virtual]`

Query the number of variables associated with this object. Used along with [ScopedObj::getVariable](#) to iterate over variables in an object. See `rt[ObjectType]GetVariableCount`

Implements [optix::ScopedObj](#).

##### 2.21.2.3 Variable `optix::GeometryInstanceObj::queryVariable ( const std::string & name )` const `[inline]`, `[virtual]`

Query a variable associated with this object by name. See `rt[ObjectType]QueryVariable`. Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implements [optix::ScopedObj](#).

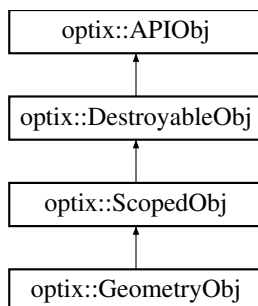
##### 2.21.2.4 void `optix::GeometryInstanceObj::setGeometry ( Geometry geometry )` `[inline]`

Set the geometry object associated with this instance. See [rtGeometryInstanceSetGeometry](#).

## 2.22 optix::GeometryObj Class Reference

### 2.22.1 Detailed Description

Geometry wraps the OptiX C API `RTgeometry` opaque type and its associated function set. Inheritance diagram for `optix::GeometryObj`:



### Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTGeometry](#) [get](#) ()
  
- void [markDirty](#) ()
- bool [isDirty](#) () const
  
- void [setPrimitiveCount](#) (unsigned int num\_primitives)
- unsigned int [getPrimitiveCount](#) () const
  
- void [setPrimitiveIndexOffset](#) (unsigned int index\_offset)
- unsigned int [getPrimitiveIndexOffset](#) () const
  
- void [setBoundingBoxProgram](#) ([Program](#) program)
- [Program](#) [getBoundingBoxProgram](#) () const
- void [setIntersectionProgram](#) ([Program](#) program)
- [Program](#) [getIntersectionProgram](#) () const
  
- [Variable](#) [declareVariable](#) (const std::string &name)
- [Variable](#) [queryVariable](#) (const std::string &name) const
- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const

### Friends

- class **Handle**< **GeometryObj** >

### Additional Inherited Members

#### 2.22.2 Member Function Documentation

##### 2.22.2.1 Variable `optix::GeometryObj::declareVariable ( const std::string & name )` `[inline]`, `[virtual]`

Declare a variable associated with this object. See `rt[ObjectType]DeclareVariable`. Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implements [optix::ScopedObj](#).

2.22.2.2 `unsigned int optix::GeometryObj::getPrimitiveCount ( ) const [inline]`

Query the number of primitives in this geometry object (eg, number of triangles in mesh). See [rtGeometryGetPrimitiveCount](#)

2.22.2.3 `unsigned int optix::GeometryObj::getPrimitiveIndexOffset ( ) const [inline]`

Query the primitive index offset for this geometry object. See [rtGeometryGetPrimitiveIndexOffset](#)

2.22.2.4 `unsigned int optix::GeometryObj::getVariableCount ( ) const [inline],[virtual]`

Query the number of variables associated with this object. Used along with [ScopedObj::getVariable](#) to iterate over variables in an object. See [rt\[ObjectType\]GetVariableCount](#)

Implements [optix::ScopedObj](#).

2.22.2.5 `void optix::GeometryObj::markDirty ( ) [inline]`

Mark this geometry as dirty, causing rebuild of parent groups acceleration. See [rtGeometryMarkDirty](#).

2.22.2.6 `Variable optix::GeometryObj::queryVariable ( const std::string & name ) const [inline],[virtual]`

Query a variable associated with this object by name. See [rt\[ObjectType\]QueryVariable](#). Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implements [optix::ScopedObj](#).

2.22.2.7 `void optix::GeometryObj::setBoundingBoxProgram ( Program program ) [inline]`

Set the bounding box program for this geometry. See [rtGeometrySetBoundingBoxProgram](#).

2.22.2.8 `void optix::GeometryObj::setPrimitiveCount ( unsigned int num_primitives ) [inline]`

Set the number of primitives in this geometry object (eg, number of triangles in mesh). See [rtGeometrySetPrimitiveCount](#)

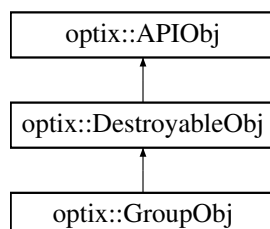
2.22.2.9 `void optix::GeometryObj::setPrimitiveIndexOffset ( unsigned int index_offset ) [inline]`

Set the primitive index offset for this geometry object. See [rtGeometrySetPrimitiveIndexOffset](#)

## 2.23 optix::GroupObj Class Reference

### 2.23.1 Detailed Description

Group wraps the OptiX C API RTgroup opaque type and its associated function set. Inheritance diagram for `optix::GroupObj`:



#### Public Member Functions

- void [destroy](#) ( )
- void [validate](#) ( )

- [Context](#) `getContext ()` const
- [RTgroup](#) `get ()`
- void [setAcceleration](#) ([Acceleration](#) acceleration)
- [Acceleration](#) `getAcceleration ()` const
- void [setChildCount](#) (unsigned int count)
- unsigned int [getChildCount](#) () const
- template<typename T >  
void [setChild](#) (unsigned int index, T child)
- template<typename T >  
T [getChild](#) (unsigned int index) const
- [RObjecttype](#) `getChildType` (unsigned int index) const
- template<typename T >  
unsigned int [addChild](#) (T child)
- template<typename T >  
unsigned int [removeChild](#) (T child)
- void [removeChild](#) (int index)
- void [removeChild](#) (unsigned int index)
- template<typename T >  
unsigned int [getChildIndex](#) (T child) const

#### Friends

- class **Handle**< **GroupObj** >

#### Additional Inherited Members

### 2.23.2 Member Function Documentation

#### 2.23.2.1 template<typename T > unsigned int optix::GroupObj::removeChild ( T *child* ) [inline]

Remove a child in this group. Note: this function is not order-preserving. Returns the position of the removed element if succeeded. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid.

#### 2.23.2.2 void optix::GroupObj::removeChild ( int *index* ) [inline]

Remove a child in this group. Note: this function is not order-preserving. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid.

#### 2.23.2.3 void optix::GroupObj::removeChild ( unsigned int *index* ) [inline]

Remove a child in this group. Note: this function is not order-preserving. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid.

#### 2.23.2.4 void optix::GroupObj::setAcceleration ( [Acceleration](#) *acceleration* ) [inline]

Set the Acceleration structure for this group. See [rtGroupSetAcceleration](#).

#### 2.23.2.5 void optix::GroupObj::setChildCount ( unsigned int *count* ) [inline]

Set the number of children for this group. See [rtGroupSetChildCount](#).

## 2.24 optix::Handle< T > Class Template Reference

### 2.24.1 Detailed Description

template<class T>class optix::Handle< T >

The [Handle](#) class is a reference counted handle class used to manipulate API objects.

All interaction with API objects should be done via these handles and the associated typedefs rather than direct usage of the objects.

#### Public Member Functions

- [Handle](#) ()
- [Handle](#) (T \*ptr)
- template<class U >  
  [Handle](#) (U \*ptr)
- [Handle](#) (const [Handle](#)< T > &copy)
- template<class U >  
  [Handle](#) (const [Handle](#)< U > &copy)
- [Handle](#)< T > & [operator=](#) (const [Handle](#)< T > &copy)
- template<class U >  
  [Handle](#)< T > & [operator=](#) (const [Handle](#)< U > &copy)
- [~Handle](#) ()
- T \* [operator->](#) ()
- const T \* [operator->](#) () const
- T \* [get](#) ()
- const T \* [get](#) () const
- [operator bool](#) () const
- [Handle](#)< [VariableObj](#) > [operator\[\]](#) (const std::string &varname)
- [Handle](#)< [VariableObj](#) > [operator\[\]](#) (const char \*varname)

#### Static Public Member Functions

- static [Handle](#)< T > [take](#) (typename T::api\_t p)
- static [Handle](#)< T > [take](#) (RObject p)
- static [Handle](#)< T > [create](#) ()
- static [Handle](#)< T > [create](#) (const std::string &a, const std::string &b, const std::string &c)
- static unsigned int [getDeviceCount](#) ()

### 2.24.2 Member Function Documentation

2.24.2.1 template<class T > [Handle](#)< [VariableObj](#) > optix::Handle< T >::operator[] ( const std::string & varname )

Variable access operator. This operator will query the API object for a variable with the given name, creating a new variable instance if necessary. Only valid for ScopedObjs.

2.24.2.2 template<class T > [Handle](#)< [VariableObj](#) > optix::Handle< T >::operator[] ( const char \* varname )

Variable access operator. Identical to [operator\[\]](#)(const std::string& varname)

Explicitly define char\* version to avoid ambiguities between builtin operator[](int, char\*) and Handle::operator[](std::string ). The problem lies in that a [Handle](#) can be cast to a bool then to an int which implies that:

```
Context context;
context["var"];
```

can be interpreted as either

```
1["var"]; // Strange but legal way to index into a string (same as "var"[1] )
```

or

```
context[ std::string("var") ];
```

### 2.24.2.3 template<class T> static Handle<T> optix::Handle< T >::take( RObject p ) [inline],[static]

Special version that takes an RObject which must be cast up to the appropriate OptiX API opaque type.

## 2.25 rti\_internal\_callableprogram::is\_CPArgVoid< T1 > Struct Template Reference

### Static Public Attributes

- static const bool **result** = false

## 2.26 rti\_internal\_callableprogram::is\_CPArgVoid< CPArgVoid > Struct Template Reference

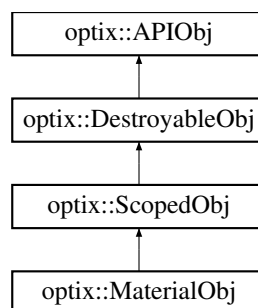
### Static Public Attributes

- static const bool **result** = true

## 2.27 optix::MaterialObj Class Reference

### 2.27.1 Detailed Description

Material wraps the OptiX C API RTmaterial opaque type and its associated function set. Inheritance diagram for optix::MaterialObj:



### Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTmaterial](#) [get](#) ()
- void [setClosestHitProgram](#) (unsigned int ray\_type\_index, [Program](#) program)
- [Program](#) [getClosestHitProgram](#) (unsigned int ray\_type\_index) const
- void [setAnyHitProgram](#) (unsigned int ray\_type\_index, [Program](#) program)
- [Program](#) [getAnyHitProgram](#) (unsigned int ray\_type\_index) const



- Variable `declareVariable` (const std::string &name)
- Variable `queryVariable` (const std::string &name) const
- void `removeVariable` (Variable v)
- unsigned int `getVariableCount` () const
- Variable `getVariable` (unsigned int index) const

#### Friends

- class `Handle< MaterialObj >`

#### Additional Inherited Members

### 2.27.2 Member Function Documentation

#### 2.27.2.1 Variable `optix::MaterialObj::declareVariable` ( const std::string & *name* ) [inline], [virtual]

Declare a variable associated with this object. See `rt[ObjectType]DeclareVariable`. Note that this function is wrapped by the convenience function `Handle::operator[]`.

Implements `optix::ScopedObj`.

#### 2.27.2.2 unsigned int `optix::MaterialObj::getVariableCount` ( ) const [inline], [virtual]

Query the number of variables associated with this object. Used along with `ScopedObj::getVariable` to iterate over variables in an object. See `rt[ObjectType]GetVariableCount`

Implements `optix::ScopedObj`.

#### 2.27.2.3 Variable `optix::MaterialObj::queryVariable` ( const std::string & *name* ) const [inline], [virtual]

Query a variable associated with this object by name. See `rt[ObjectType]QueryVariable`. Note that this function is wrapped by the convenience function `Handle::operator[]`.

Implements `optix::ScopedObj`.

#### 2.27.2.4 void `optix::MaterialObj::setClosestHitProgram` ( unsigned int *ray\_type\_index*, Program *program* ) [inline]

Set closest hit program for this material at the given *ray\_type* index. See `rtMaterialSetClosestHitProgram`.

## 2.28 `optix::Matrix< M, N >` Class Template Reference

### 2.28.1 Detailed Description

```
template<unsigned int M, unsigned int N>class optix::Matrix< M, N >
```

A matrix with M rows and N columns.

#### Description

`Matrix` provides a utility class for small-dimension floating-point matrices, such as transformation matrices. `Matrix` may also be useful in other computation and can be used in both host and device code. Typedefs are provided for 2x2 through 4x4 matrices.

#### History

`Matrix` was introduced in OptiX 1.0.

See also `rtVariableSetMatrix*`

## Public Types

- typedef [VectorDim](#)< N >::VectorType **floatN**
- typedef [VectorDim](#)< M >::VectorType **floatM**

## Public Member Functions

- RT\_HOSTDEVICE [Matrix](#) ()
- RT\_HOSTDEVICE [Matrix](#) (const float data[M \*N])
- RT\_HOSTDEVICE [Matrix](#) (const [Matrix](#) &m)
- RT\_HOSTDEVICE [Matrix](#) & **operator=** (const [Matrix](#) &b)
- RT\_HOSTDEVICE float **operator[]** (unsigned int i) const
- RT\_HOSTDEVICE float & **operator[]** (unsigned int i)
- RT\_HOSTDEVICE floatN **getRow** (unsigned int m) const
- RT\_HOSTDEVICE **floatM** **getCol** (unsigned int n) const
- RT\_HOSTDEVICE float \* **getData** ()
- RT\_HOSTDEVICE const float \* **getData** () const
- RT\_HOSTDEVICE void **setRow** (unsigned int m, const floatN &r)
- RT\_HOSTDEVICE void **setCol** (unsigned int n, const **floatM** &c)
- RT\_HOSTDEVICE [Matrix](#)< N, M > **transpose** () const
- RT\_HOSTDEVICE [Matrix](#)< 4, 4 > **inverse** () const
- RT\_HOSTDEVICE float **det** () const
- RT\_HOSTDEVICE bool **operator<** (const [Matrix](#)< M, N > &rhs) const
- template<>  
OPTIXU\_INLINE RT\_HOSTDEVICE float **det** () const
- template<>  
OPTIXU\_INLINE RT\_HOSTDEVICE float **det** () const
- template<>  
OPTIXU\_INLINE RT\_HOSTDEVICE  
[Matrix](#)< 4, 4 > **inverse** () const
- template<>  
OPTIXU\_INLINE RT\_HOSTDEVICE  
[Matrix](#)< 4, 4 > **rotate** (const float radians, const float3 &axis)
- template<>  
OPTIXU\_INLINE RT\_HOSTDEVICE  
[Matrix](#)< 4, 4 > **translate** (const float3 &vec)
- template<>  
OPTIXU\_INLINE RT\_HOSTDEVICE  
[Matrix](#)< 4, 4 > **scale** (const float3 &vec)

## Static Public Member Functions

- static RT\_HOSTDEVICE [Matrix](#)< 4, 4 > **rotate** (const float radians, const float3 &axis)
- static RT\_HOSTDEVICE [Matrix](#)< 4, 4 > **translate** (const float3 &vec)
- static RT\_HOSTDEVICE [Matrix](#)< 4, 4 > **scale** (const float3 &vec)
- static RT\_HOSTDEVICE [Matrix](#)< N, N > **identity** ()

## 2.28.2 Constructor &amp; Destructor Documentation

## 2.28.2.1 template&lt;unsigned int M, unsigned int N&gt; OPTIXU\_INLINE RT\_HOSTDEVICE optix::Matrix&lt; M, N &gt;::Matrix ( )

A column of the matrix.

Create an uninitialized matrix

2.28.2.2 `template<unsigned int M, unsigned int N> RT_HOSTDEVICE optix::Matrix< M, N >::Matrix ( const float data[M * N] ) [inline], [explicit]`

Create a matrix from the specified float array

2.28.2.3 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE optix::Matrix< M, N >::Matrix ( const Matrix< M, N > & m )`

Copy the matrix

### 2.28.3 Member Function Documentation

2.28.3.1 `template<unsigned int M, unsigned int N> RT_HOSTDEVICE float optix::Matrix< M, N >::det ( ) const`

Returns the determinant of the matrix

2.28.3.2 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N >::floatM optix::Matrix< M, N >::getCol ( unsigned int n ) const`

Access the specified column 0..N. Returns float, float2, float3 or float4 depending on the matrix size

2.28.3.3 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE float * optix::Matrix< M, N >::getData ( )`

Returns a pointer to the internal data array. The data array is stored in row-major order.

2.28.3.4 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE const float * optix::Matrix< M, N >::getData ( ) const`

Returns a const pointer to the internal data array. The data array is stored in row-major order.

2.28.3.5 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N >::floatN optix::Matrix< M, N >::getRow ( unsigned int m ) const`

Access the specified row 0..M. Returns float, float2, float3 or float4 depending on the matrix size

2.28.3.6 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< N, N > optix::Matrix< M, N >::identity ( ) [static]`

Returns the identity matrix

2.28.3.7 `template<unsigned int M, unsigned int N> RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::inverse ( ) const`

Returns the inverse of the matrix

2.28.3.8 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE bool optix::Matrix< M, N >::operator< ( const Matrix< M, N > & rhs ) const`

Ordered comparison operator so that the matrix can be used in an STL container

2.28.3.9 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N > & optix::Matrix< M, N >::operator= ( const Matrix< M, N > & b )`

Assignment operator

2.28.3.10 `template<unsigned int M, unsigned int N> RT_HOSTDEVICE float optix::Matrix< M, N >::operator[] ( unsigned int i ) const [inline]`

Access the specified element 0..N\*M-1

2.28.3.11 `template<unsigned int M, unsigned int N> RT_HOSTDEVICE float& optix::Matrix< M, N >::operator[] ( unsigned int i ) [inline]`

Access the specified element 0..N\*M-1

2.28.3.12 `template<unsigned int M, unsigned int N> static RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::rotate ( const float radians, const float3 & axis ) [static]`

Returns a rotation matrix

2.28.3.13 `template<unsigned int M, unsigned int N> static RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::scale ( const float3 & vec ) [static]`

Returns a scale matrix

2.28.3.14 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE void optix::Matrix< M, N >::setCol ( unsigned int n, const floatM & c )`

Assign the specified column 0..N. Takes a float, float2, float3 or float4 depending on the matrix size

2.28.3.15 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE void optix::Matrix< M, N >::setRow ( unsigned int m, const floatN & r )`

Assign the specified row 0..M. Takes a float, float2, float3 or float4 depending on the matrix size

2.28.3.16 `template<unsigned int M, unsigned int N> static RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::translate ( const float3 & vec ) [static]`

Returns a translation matrix

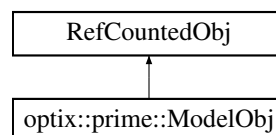
2.28.3.17 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< N, M > optix::Matrix< M, N >::transpose ( ) const`

Returns the transpose of the matrix

## 2.29 optix::prime::ModelObj Class Reference

### 2.29.1 Detailed Description

Encapsulates an OptiX Prime model. The purpose of a model is to represent a set of triangles and an acceleration structure. Inheritance diagram for optix::prime::ModelObj:



### Public Member Functions

- Query `createQuery` (`RTPQuerytype` queryType)
- Context `getContext` ()
- void `finish` ()
- int `isFinished` ()
- void `update` (unsigned hints)
- void `copy` (const `Model` &srcModel)
- void `setTriangles` (RTPsize triCount, `RTPbuffertype` type, const void \*vertPtr, unsigned stride=0)

- void [setTriangles](#) (RTPsize triCount, [RTPbuffertype](#) type, const void \*indexPtr, RTPsize vertCount, [RT-Pbuffertype](#) vertType, const void \*vertPtr, unsigned stride=0)
- void [setTriangles](#) (const [BufferDesc](#) &vertices)
- void [setTriangles](#) (const [BufferDesc](#) &indices, const [BufferDesc](#) &vertices)
- void [setInstances](#) (RTPsize count, [RTPbuffertype](#) instanceType, const [RTPmodel](#) \*instanceList, [RT-Pbufferformat](#) transformFormat, [RTPbuffertype](#) transformType, const void \*transformList)
- void [setInstances](#) (const [BufferDesc](#) &instances, const [BufferDesc](#) &transforms)
- void [setBuilderParameter](#) ([RTPbuilderparam](#) param, RTPsize size, const void \*p)
- template<typename T >  
void [setBuilderParameter](#) ([RTPbuilderparam](#) param, const T &val)
- [RTPmodel](#) [getRTPmodel](#) ()

#### Friends

- class [ContextObj](#)
- class [QueryObj](#)

#### 2.29.2 Member Function Documentation

**2.29.2.1** void [optix::prime::ModelObj::setBuilderParameter](#) ( [RTPbuilderparam](#) *param*, RTPsize *size*, const void \* *p* )  
[inline]

Sets a model build parameter See [rtpModelSetBuilderParameter](#) for additional information

**2.29.2.2** template<typename T > void [optix::prime::ModelObj::setBuilderParameter](#) ( [RTPbuilderparam](#) *param*, const T &  
*val* )

Sets a model build parameter See [rtpModelSetBuilderParameter](#) for additional information

**2.29.2.3** void [optix::prime::ModelObj::setInstances](#) ( RTPsize *count*, [RTPbuffertype](#) *instanceType*, const [RTPmodel](#) \*  
*instanceList*, [RTPbufferformat](#) *transformFormat*, [RTPbuffertype](#) *transformType*, const void \* *transformList* )  
[inline]

Sets the instance data for a model. This function creates buffer descriptors of the specified types and formats, populates them with the supplied data and assigns them to the model. See [rtpModelSetInstances](#) for additional information

**2.29.2.4** void [optix::prime::ModelObj::setInstances](#) ( const [BufferDesc](#) & *instances*, const [BufferDesc](#) & *transforms* )  
[inline]

Sets the instance data for a model using the supplied buffer descriptors. See [rtpModelSetInstances](#) for additional information

**2.29.2.5** void [optix::prime::ModelObj::setTriangles](#) ( RTPsize *triCount*, [RTPbuffertype](#) *type*, const void \* *vertPtr*, unsigned  
*stride* = 0 ) [inline]

Sets the triangle data for a model. This function creates a buffer descriptor of the specified type, populates it with the supplied data and assigns it to the model. The list of vertices is assumed to be a flat list of triangles and each three vertices form a single triangle. See [rtpModelSetTriangles](#) for additional information

**2.29.2.6** void [optix::prime::ModelObj::setTriangles](#) ( RTPsize *triCount*, [RTPbuffertype](#) *type*, const void \* *indexPtr*, RTPsize  
*vertCount*, [RTPbuffertype](#) *vertType*, const void \* *vertPtr*, unsigned *stride* = 0 ) [inline]

Sets the triangle data for a model. This function creates buffer descriptors of the specified types, populates them with the supplied data and assigns them to the model. The list of vertices uses the indices list to determine the triangles. See [rtpModelSetTriangles](#) for additional information

2.29.2.7 `void optix::prime::ModelObj::setTriangles ( const BufferDesc & vertices ) [inline]`

Sets the triangle data for a model using the supplied buffer descriptor of vertices. The list of vertices is assumed to be a flat list of triangles and each three vertices shape a single triangle. See [rtpModelSetTriangles](#) for additional information

2.29.2.8 `void optix::prime::ModelObj::setTriangles ( const BufferDesc & indices, const BufferDesc & vertices ) [inline]`

Sets the triangle data for a model using the supplied buffer descriptor of vertices. The list of vertices uses the indices list to determine the triangles. See [rtpModelSetTriangles](#) for additional information

## 2.30 `optix::Onb` Struct Reference

### 2.30.1 Detailed Description

Orthonormal basis

#### Public Member Functions

- OPTIXU\_INLINE RT\_HOSTDEVICE **Onb** (const float3 &normal)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **inverse\_transform** (float3 &p) const

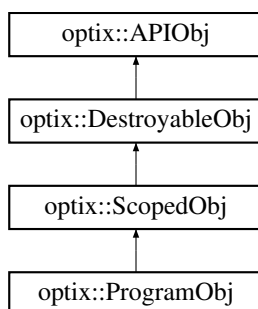
#### Public Attributes

- float3 **m\_tangent**
- float3 **m\_binormal**
- float3 **m\_normal**

## 2.31 `optix::ProgramObj` Class Reference

### 2.31.1 Detailed Description

Program object wraps the OptiX C API RTprogram opaque type and its associated function set. Inheritance diagram for `optix::ProgramObj`:



#### Public Member Functions

- void **destroy** ()
- void **validate** ()
- **Context** **getContext** () const
- **Variable** **declareVariable** (const std::string &name)
- **Variable** **queryVariable** (const std::string &name) const

- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const
- [RTprogram](#) [get](#) ()
- int [getId](#) () const

#### Friends

- class **Handle**< **ProgramObj** >

#### Additional Inherited Members

##### 2.31.2 Member Function Documentation

###### 2.31.2.1 Variable `optix::ProgramObj::declareVariable ( const std::string & name )` `[inline]`, `[virtual]`

Declare a variable associated with this object. See `rt[ObjectType]DeclareVariable`. Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implements [optix::ScopedObj](#).

###### 2.31.2.2 `int optix::ProgramObj::getId ( ) const` `[inline]`

Returns the device-side ID of this program object. See [rtProgramGetId](#)

###### 2.31.2.3 `unsigned int optix::ProgramObj::getVariableCount ( ) const` `[inline]`, `[virtual]`

Query the number of variables associated with this object. Used along with [ScopedObj::getVariable](#) to iterate over variables in an object. See `rt[ObjectType]GetVariableCount`

Implements [optix::ScopedObj](#).

###### 2.31.2.4 Variable `optix::ProgramObj::queryVariable ( const std::string & name ) const` `[inline]`, `[virtual]`

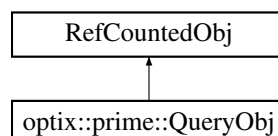
Query a variable associated with this object by name. See `rt[ObjectType]QueryVariable`. Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implements [optix::ScopedObj](#).

## 2.32 optix::prime::QueryObj Class Reference

### 2.32.1 Detailed Description

Encapsulates an OptiX Prime query. The purpose of a query is to coordinate the intersection of rays with a model. Inheritance diagram for `optix::prime::QueryObj`:



#### Public Member Functions

- [Context](#) [getContext](#) ()

- void `finish` ()
- int `isFinished` ()
- void `setCudaStream` (cudaStream\_t stream)
- void `setRays` (RTPsize count, RTPbufferformat format, RTPbuffertype type, void \*rays)
- void `setRays` (const BufferDesc &rays)
- void `setHits` (RTPsize count, RTPbufferformat format, RTPbuffertype type, void \*hits)
- void `setHits` (const BufferDesc &hits)
- void `execute` (unsigned hint)
- RTPQuery `getRTPQuery` ()

#### Friends

- class **ContextObj**
- class **ModelObj**

## 2.33 Ray Struct Reference

### 2.33.1 Detailed Description

`Ray` class.

#### Description

`Ray` is an encapsulation of a ray mathematical entity. The origin and direction members specify the ray, while the `ray_type` member specifies which closest-hit/any-hit pair will be used when the ray hits a geometry object. The `tmin/tmax` members specify the interval over which the ray is valid.

To avoid numerical range problems, the value `RT_DEFAULT_MAX` can be used to specify an infinite extent.

During C++ compilation, `Ray` is contained within the `optix::` namespace but has global scope during C compilation. `Ray`'s constructors are not available during C compilation.

#### Members

```
// The origin of the ray
float3 origin;

// The direction of the ray
float3 direction;

// The ray type associated with this ray
unsigned int ray_type;

// The min and max extents associated with this ray
float tmin;
float tmax;
```

#### Constructors

```
// Create a Ray with undefined member values
Ray( void );

// Create a Ray copied from an exemplar
Ray( const Ray &r );

// Create a ray with a specified origin, direction, ray_type, and min/max extents.
// When tmax is not given, it defaults to @ref RT_DEFAULT_MAX.
Ray( float3 origin, float3 direction, unsigned int ray_type,
      float tmin, float tmax = RT_DEFAULT_MAX );
```

#### Functions

```
// Create a ray with a specified origin, direction, ray type, and min/max extents.
Ray make_Ray( float3 origin,
              float3 direction,
              unsigned int ray_type,
              float tmin,
              float tmax );
```



## History

`Ray` was introduced in OptiX 1.0.

**See also** `rtContextSetRayTypeCount`, `rtMaterialSetAnyHitProgram`, `rtMaterialSetClosestHitProgram`

## Public Attributes

- float3 `origin`
- float3 `direction`
- unsigned int `ray_type`
- float `tmin`
- float `tmax`

### 2.33.2 Member Data Documentation

#### 2.33.2.1 float3 `Ray::direction`

The direction of the ray

#### 2.33.2.2 float3 `Ray::origin`

The origin of the ray

#### 2.33.2.3 unsigned int `Ray::ray_type`

The ray type associated with this ray

#### 2.33.2.4 float `Ray::tmax`

The max extent associated with this ray

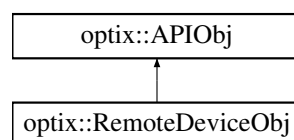
#### 2.33.2.5 float `Ray::tmin`

The min extent associated with this ray

## 2.34 `optix::RemoteDeviceObj` Class Reference

### 2.34.1 Detailed Description

`RemoteDevice` wraps the OptiX C API `RTremotedevice` opaque type and its associated function set. Inheritance diagram for `optix::RemoteDeviceObj`:



## Public Member Functions

- void **destroy** ()
- void **reserve** (unsigned int num\_nodes, unsigned int configuration\_idx)
- void **release** ()
- void **getAttribute** (`RTremotedeviceattribute` attrib, `RTsize` size, void \*p)
- std::string **getConfiguration** (unsigned int index)
- `RTremotedevice` **get** ()

## Static Public Member Functions

- static `RemoteDevice create` (const std::string &url, const std::string &username, const std::string &password)

## Friends

- class `Handle< RemoteDeviceObj >`

2.35 `optix::rt_print_t< T >` Struct Template Reference

## Static Public Attributes

- static const int `desc` = 0

2.36 `optix::rt_print_t< double >` Struct Template Reference

## Static Public Attributes

- static const int `desc` = 3

2.37 `optix::rt_print_t< float >` Struct Template Reference

## Static Public Attributes

- static const int `desc` = 2

2.38 `optix::rt_print_t< long long >` Struct Template Reference

## Static Public Attributes

- static const int `desc` = 1

2.39 `optix::rt_print_t< unsigned long long >` Struct Template Reference

## Static Public Attributes

- static const int `desc` = 1

2.40 `rtCallableProgramSizeofWrapper< T >` Struct Template Reference

## Static Public Attributes

- static const size\_t `value` = sizeof(T)

2.41 `rtCallableProgramSizeofWrapper< void >` Struct Template Reference

## Static Public Attributes

- static const size\_t `value` = 0

## 2.42 `rti_internal_typeinfo::rti_typeenum< T >` Struct Template Reference

### Static Public Attributes

- static const int **m\_typeenum** = `_OPTIX_TYPE_ENUM_UNKNOWN`

## 2.43 `rti_internal_typeinfo::rti_typeenum< optix::boundCallableProgramId< T > >` Struct Template Reference

### Static Public Attributes

- static const int **m\_typeenum** = `_OPTIX_TYPE_ENUM_PROGRAM_AS_ID`

## 2.44 `rti_internal_typeinfo::rti_typeenum< optix::callableProgramId< T > >` Struct Template Reference

### Static Public Attributes

- static const int **m\_typeenum** = `_OPTIX_TYPE_ENUM_PROGRAM_ID`

## 2.45 `rti_internal_typeinfo::rti_typeinfo` Struct Reference

### Public Attributes

- unsigned int **kind**
- unsigned int **size**

## 2.46 `rtObject` Struct Reference

### 2.46.1 Detailed Description

Opaque handle to a OptiX object.

#### Description

`rtObject` is an opaque handle to an OptiX object of any type. To set or query the variable value, use `rtVariableSetObject` and `rtVariableGetObject`.

Depending on how exactly the variable is used, only certain concrete types may make sense. For example, when used as an argument to `rtTrace`, the variable must be set to any OptiX type of `RTgroup`, `RTselector`, `RTgeometrygroup`, or `RTtransform`.

Note that for certain OptiX types, there are more specialized handles available to access a variable. For example, to access an OptiX object of type `RTtexturesampler`, a handle of type `rtTextureSampler` provides more functionality than one of the generic type `rtObject`.

#### History

`rtObject` was introduced in OptiX 1.0.

**See also** `rtVariableSetObject`, `rtVariableGetObject`, `rtTrace`, `rtTextureSampler`, `rtBuffer`

### Protected Member Functions

- void **never\_call** ()

## Protected Attributes

- unsigned int **handle**

## 2.47 RTPinternals\_3070 Struct Reference

## Classes

- struct [BvhNode](#)
- struct [WoopTriangle](#)

## Public Attributes

- int **numNodes**
- int **numEntities**
- [BvhNode](#) \* **nodes**
- int \* **remap**
- [WoopTriangle](#) \* **woopTriangles**
- int **indexStride**
- int **vertexStride**
- int \* **indices**
- float \* **vertices**
- int **matrixStride**
- float \* **invMatrices**
- int \* **instanceToModelId**

## 2.48 RTUtraversalresult Struct Reference

## 2.48.1 Detailed Description

Traversal API allowing batch raycasting queries utilizing either OptiX or the CPU.

The OptiX traversal API is demonstrated in the traversal sample within the OptiX SDK.

Structure encapsulating the result of a single ray query

## Public Attributes

- int [prim\\_id](#)
- float [t](#)

## 2.48.2 Member Data Documentation

## 2.48.2.1 int RTUtraversalresult::prim\_id

Index of the intereseected triangle, -1 for miss

## 2.48.2.2 float RTUtraversalresult::t

[Ray](#) t parameter of hit point

## 2.49 optix::ScopedObj Class Reference

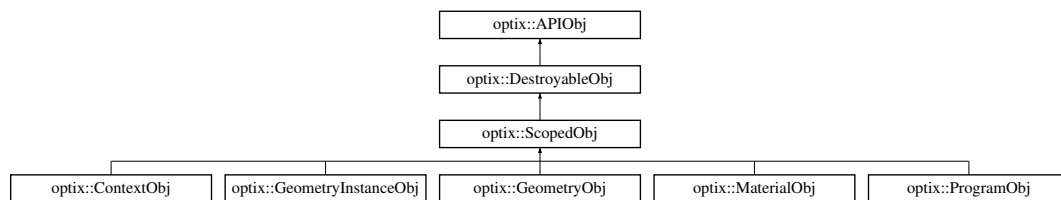
### 2.49.1 Detailed Description

Base class for all objects which are OptiX variable containers.

Wraps:

- RTcontext
- RTgeometry
- RTgeometryinstance
- RTmaterial
- RTprogram

Inheritance diagram for optix::ScopedObj:



### Public Member Functions

- virtual [Variable declareVariable](#) (const std::string &name)=0
- virtual [Variable queryVariable](#) (const std::string &name) const =0
- virtual void [removeVariable](#) ([Variable](#) v)=0
- virtual unsigned int [getVariableCount](#) () const =0
- virtual [Variable getVariable](#) (unsigned int index) const =0

### Additional Inherited Members

### 2.49.2 Member Function Documentation

#### 2.49.2.1 virtual [Variable](#) optix::ScopedObj::declareVariable ( const std::string & name ) [pure virtual]

Declare a variable associated with this object. See `rt[ObjectType]DeclareVariable`. Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implemented in [optix::MaterialObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::ProgramObj](#), and [optix::ContextObj](#).

#### 2.49.2.2 virtual unsigned int optix::ScopedObj::getVariableCount ( ) const [pure virtual]

Query the number of variables associated with this object. Used along with [ScopedObj::getVariable](#) to iterate over variables in an object. See `rt[ObjectType]GetVariableCount`

Implemented in [optix::MaterialObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::ProgramObj](#), and [optix::ContextObj](#).

2.49.2.3 virtual Variable optix::ScopedObj::queryVariable ( const std::string & name ) const [pure virtual]

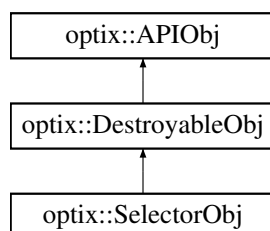
Query a variable associated with this object by name. See rt[ObjectType]QueryVariable. Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implemented in [optix::MaterialObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::ProgramObj](#), and [optix::ContextObj](#).

## 2.50 optix::SelectorObj Class Reference

### 2.50.1 Detailed Description

Selector wraps the OptiX C API RTselector opaque type and its associated function set. Inheritance diagram for optix::SelectorObj:



### Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTselector](#) [get](#) ()
  
- void [setVisitProgram](#) ([Program](#) program)
- [Program](#) [getVisitProgram](#) () const
  
- void [setChildCount](#) (unsigned int count)
- unsigned int [getChildCount](#) () const
- template<typename T >  
void [setChild](#) (unsigned int index, T child)
- template<typename T >  
T [getChild](#) (unsigned int index) const
- [RTobjecttype](#) [getChildType](#) (unsigned int index) const
- template<typename T >  
unsigned int [addChild](#) (T child)
- template<typename T >  
unsigned int [removeChild](#) (T child)
- void [removeChild](#) (int index)
- void [removeChild](#) (unsigned int index)
- template<typename T >  
unsigned int [getChildIndex](#) (T child) const
  
- [Variable](#) [declareVariable](#) (const std::string &name)
- [Variable](#) [queryVariable](#) (const std::string &name) const
- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const

## Friends

- class **Handle**< **SelectorObj** >

## Additional Inherited Members

## 2.50.2 Member Function Documentation

2.50.2.1 `template<typename T> unsigned int optix::SelectorObj::removeChild ( T child ) [inline]`

Remove a child in this group and returns the index to the deleted element in case of success. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid. Note: this function shifts down all the elements next to the removed one.

2.50.2.2 `void optix::SelectorObj::removeChild ( int index ) [inline]`

Remove a child in this group by its index. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid. Note: this function shifts down all the elements next to the removed one.

2.50.2.3 `void optix::SelectorObj::removeChild ( unsigned int index ) [inline]`

Remove a child in this group by its index. Throws [RT\\_ERROR\\_INVALID\\_VALUE](#) if the parameter is invalid. Note: this function shifts down all the elements next to the removed one.

2.50.2.4 `void optix::SelectorObj::setChildCount ( unsigned int count ) [inline]`

Set the number of children for this group. See [rtSelectorSetChildCount](#).

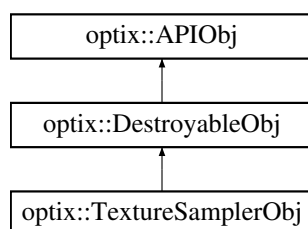
2.50.2.5 `void optix::SelectorObj::setVisitProgram ( Program program ) [inline]`

Set the visitor program for this selector. See [rtSelectorSetVisitProgram](#)

## 2.51 optix::TextureSamplerObj Class Reference

## 2.51.1 Detailed Description

TextureSampler wraps the OptiX C API `RTtexturesampler` opaque type and its associated function set. Inheritance diagram for `optix::TextureSamplerObj`:



## Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTtexturesampler](#) [get](#) ()
- void [setMipLevelCount](#) (unsigned int num\_mip\_levels)

- unsigned int `getMipLevelCount` () const
  - void `setArraySize` (unsigned int num\_textures\_in\_array)
  - unsigned int `getArraySize` () const
  - void `setWrapMode` (unsigned int dim, `RTwrapmode` wrapmode)
  - `RTwrapmode` `getWrapMode` (unsigned int dim) const
  - void `setFilteringModes` (`RTfiltermode` minification, `RTfiltermode` magnification, `RTfiltermode` mipmapping)
  - void `getFilteringModes` (`RTfiltermode` &minification, `RTfiltermode` &magnification, `RTfiltermode` &mipmapping) const
  - void `setMaxAnisotropy` (float value)
  - float `getMaxAnisotropy` () const
  - void `setMipLevelClamp` (float minLevel, float maxLevel)
  - void `getMipLevelClamp` (float &minLevel, float &maxLevel) const
  - void `setMipLevelBias` (float value)
  - float `getMipLevelBias` () const
  - void `setReadMode` (`RTtexturereadmode` readmode)
  - `RTtexturereadmode` `getReadMode` () const
  - void `setIndexingMode` (`RTtextureindexmode` indexmode)
  - `RTtextureindexmode` `getIndexingMode` () const
- 
- int `getId` () const
- 
- void `setBuffer` (unsigned int texture\_array\_idx, unsigned int mip\_level, `Buffer` buffer)
  - `Buffer` `getBuffer` (unsigned int texture\_array\_idx, unsigned int mip\_level) const
  - void `setBuffer` (`Buffer` buffer)
  - `Buffer` `getBuffer` () const
- 
- void `registerGLTexture` ()
  - void `unregisterGLTexture` ()
- 
- void `registerD3D9Texture` ()
  - void `registerD3D10Texture` ()
  - void `registerD3D11Texture` ()
  - void `unregisterD3D9Texture` ()
  - void `unregisterD3D10Texture` ()
  - void `unregisterD3D11Texture` ()

#### Friends

- class `Handle`< `TextureSamplerObj` >

#### Additional Inherited Members

##### 2.51.2 Member Function Documentation

###### 2.51.2.1 `int optix::TextureSamplerObj::getId ( ) const` `[inline]`

Returns the device-side ID of this sampler. See `rtTextureSamplerGetId`

###### 2.51.2.2 `void optix::TextureSamplerObj::registerD3D9Texture ( )` `[inline]`

Declare the texture's buffer as immutable and accessible by OptiX. See `rtTextureSamplerD3D9Register`.

###### 2.51.2.3 `void optix::TextureSamplerObj::registerGLTexture ( )` `[inline]`

Declare the texture's buffer as immutable and accessible by OptiX. See `rtTextureSamplerGLRegister`.



2.51.2.4 void optix::TextureSamplerObj::setBuffer ( unsigned int *texture\_array\_idx*, unsigned int *mip\_level*, Buffer *buffer* )  
[inline]

**Deprecated in OptiX 4.0** Set the underlying buffer used for texture storage. See [rtTextureSamplerSetBuffer](#).

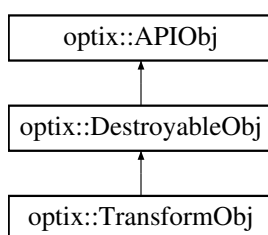
2.51.2.5 void optix::TextureSamplerObj::setMipLevelCount ( unsigned int *num\_mip\_levels* ) [inline]

**Deprecated in OptiX 4.0** Set the number of mip levels for this sampler. See [rtTextureSamplerSetMipLevelCount](#).

## 2.52 optix::TransformObj Class Reference

### 2.52.1 Detailed Description

Transform wraps the OptiX C API RTtransform opaque type and its associated function set. Inheritance diagram for optix::TransformObj:



### Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- Context [getContext](#) () const
- RTtransform [get](#) ()
  
- template<typename T >  
void [setChild](#) ( T child)
- template<typename T >  
T [getChild](#) () const
- RObjecttype [getChildType](#) () const
  
- void [setMatrix](#) (bool transpose, const float \*matrix, const float \*inverse\_matrix)
- void [getMatrix](#) (bool transpose, float \*matrix, float \*inverse\_matrix) const

### Friends

- class **Handle**< **TransformObj** >

### Additional Inherited Members

### 2.52.2 Member Function Documentation

2.52.2.1 template<typename T > void optix::TransformObj::setChild ( T *child* ) [inline]

Set the child node of this transform. See [rtTransformSetChild](#).

2.52.2.2 `void optix::TransformObj::setMatrix ( bool transpose, const float * matrix, const float * inverse_matrix )`  
`[inline]`

Set the transform matrix for this node. See [rtTransformSetMatrix](#).

## 2.53 `optix::buffer< T, Dim >::type< T2 >` Struct Template Reference

## 2.54 `optix::VariableObj` Class Reference

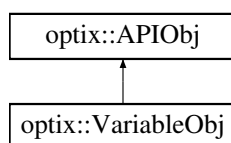
### 2.54.1 Detailed Description

Variable object wraps OptiX C API RTvariable type and its related function set.

See [OptiXApiReference](#) for complete description of the usage and behavior of RTvariable objects. Creation and querying of Variables can be performed via the [Handle::operator\[\]](#) function of the scope object associated with the variable. For example:

```
my_context["new_variable"]->setFloat( 1.0f );
```

will create a variable named `new_variable` on the object `my_context` if it does not already exist. It will then set the value of that variable to be a float 1.0f. Inheritance diagram for `optix::VariableObj`:



### Public Member Functions

- [Context](#) `getContext ()` const
- `std::string` `getName ()` const
- `std::string` `getAnnotation ()` const
- [RObjecttype](#) `getType ()` const
- [RTvariable](#) `get ()`
- `RTsize` `getSize ()` const

### Float setters

*Set variable to have a float value.*

- void [setFloat](#) (float f1)
- void [setFloat](#) (optix::float2 f)
- void [setFloat](#) (float f1, float f2)
- void [setFloat](#) (optix::float3 f)
- void [setFloat](#) (float f1, float f2, float f3)
- void [setFloat](#) (optix::float4 f)
- void [setFloat](#) (float f1, float f2, float f3, float f4)
- void [set1fv](#) (const float \*f)
- void [set2fv](#) (const float \*f)
- void [set3fv](#) (const float \*f)
- void [set4fv](#) (const float \*f)

### Int setters

*Set variable to have an int value.*

- void [setInt](#) (int i1)

- void **setInt** (int i1, int i2)
- void **setInt** (optix::int2 i)
- void **setInt** (int i1, int i2, int i3)
- void **setInt** (optix::int3 i)
- void **setInt** (int i1, int i2, int i3, int i4)
- void **setInt** (optix::int4 i)
- void **set1iv** (const int \*i)
- void **set2iv** (const int \*i)
- void **set3iv** (const int \*i)
- void **set4iv** (const int \*i)

### Unsigned int setters

*Set variable to have an unsigned int value.*

- void **setUInt** (unsigned int u1)
- void **setUInt** (unsigned int u1, unsigned int u2)
- void **setUInt** (unsigned int u1, unsigned int u2, unsigned int u3)
- void **setUInt** (unsigned int u1, unsigned int u2, unsigned int u3, unsigned int u4)
- void **setUInt** (optix::uint2 u)
- void **setUInt** (optix::uint3 u)
- void **setUInt** (optix::uint4 u)
- void **set1uiv** (const unsigned int \*u)
- void **set2uiv** (const unsigned int \*u)
- void **set3uiv** (const unsigned int \*u)
- void **set4uiv** (const unsigned int \*u)

### Matrix setters

*Set variable to have a [Matrix](#) value*

- void **setMatrix2x2fv** (bool transpose, const float \*m)
- void **setMatrix2x3fv** (bool transpose, const float \*m)
- void **setMatrix2x4fv** (bool transpose, const float \*m)
- void **setMatrix3x2fv** (bool transpose, const float \*m)
- void **setMatrix3x3fv** (bool transpose, const float \*m)
- void **setMatrix3x4fv** (bool transpose, const float \*m)
- void **setMatrix4x2fv** (bool transpose, const float \*m)
- void **setMatrix4x3fv** (bool transpose, const float \*m)
- void **setMatrix4x4fv** (bool transpose, const float \*m)

### Numeric value getters

*Query value of a variable with numeric value*

- float **getFloat** () const
- optix::float2 **getFloat2** () const
- optix::float3 **getFloat3** () const
- optix::float4 **getFloat4** () const
- void **getFloat** (float &f1) const
- void **getFloat** (float &f1, float &f2) const
- void **getFloat** (float &f1, float &f2, float &f3) const
- void **getFloat** (float &f1, float &f2, float &f3, float &f4) const
- unsigned **getUInt** () const
- optix::uint2 **getUInt2** () const
- optix::uint3 **getUInt3** () const
- optix::uint4 **getUInt4** () const
- void **getUInt** (unsigned &u1) const
- void **getUInt** (unsigned &u1, unsigned &u2) const
- void **getUInt** (unsigned &u1, unsigned &u2, unsigned &u3) const
- void **getUInt** (unsigned &u1, unsigned &u2, unsigned &u3, unsigned &u4) const
- int **getInt** () const
- optix::int2 **getInt2** () const
- optix::int3 **getInt3** () const
- optix::int4 **getInt4** () const

- void **getInt** (int &i1) const
- void **getInt** (int &i1, int &i2) const
- void **getInt** (int &i1, int &i2, int &i3) const
- void **getInt** (int &i1, int &i2, int &i3, int &i4) const
- void **getMatrix2x2** (bool transpose, float \*m) const
- void **getMatrix2x3** (bool transpose, float \*m) const
- void **getMatrix2x4** (bool transpose, float \*m) const
- void **getMatrix3x2** (bool transpose, float \*m) const
- void **getMatrix3x3** (bool transpose, float \*m) const
- void **getMatrix3x4** (bool transpose, float \*m) const
- void **getMatrix4x2** (bool transpose, float \*m) const
- void **getMatrix4x3** (bool transpose, float \*m) const
- void **getMatrix4x4** (bool transpose, float \*m) const

### OptiX API object setters

Set variable to have an OptiX API object as its value

- void **setBuffer** ([Buffer](#) buffer)
- void **set** ([Buffer](#) buffer)
- void **setTextureSampler** ([TextureSampler](#) texturesample)
- void **set** ([TextureSampler](#) texturesample)
- void **set** ([GeometryGroup](#) group)
- void **set** ([Group](#) group)
- void **set** ([Program](#) program)
- void **setProgramId** ([Program](#) program)
- void **set** ([Selector](#) selector)
- void **set** ([Transform](#) transform)

### OptiX API object getters

Retrieve OptiX API object value from a variable

- [Buffer](#) **getBuffer** () const
- [GeometryGroup](#) **getGeometryGroup** () const
- [GeometryInstance](#) **getGeometryInstance** () const
- [Group](#) **getGroup** () const
- [Program](#) **getProgram** () const
- [Selector](#) **getSelector** () const
- [TextureSampler](#) **getTextureSampler** () const
- [Transform](#) **getTransform** () const

### User data variable accessors

- void **setUserData** (RTsize size, const void \*ptr)
- void **getUserData** (RTsize size, void \*ptr) const

### Friends

- class **Handle**< [VariableObj](#) >

### Additional Inherited Members

## 2.55 optix::VectorDim< DIM > Struct Template Reference

## 2.56 optix::VectorDim< 2 > Struct Template Reference

### Public Types

- typedef float2 **VectorType**

## 2.57 `optix::VectorDim< 3 >` Struct Template Reference

### Public Types

- typedef float3 **VectorType**

## 2.58 `optix::VectorDim< 4 >` Struct Template Reference

### Public Types

- typedef float4 **VectorType**

## 2.59 `optix::VectorTypes< T, Dim >` Struct Template Reference

## 2.60 `optix::VectorTypes< float, 1 >` Struct Template Reference

### Public Types

- typedef float **Type**

### Static Public Member Functions

- template<class S >  
static `__device__`  
`__forceinline__` Type **make** (S s)

## 2.61 `optix::VectorTypes< float, 2 >` Struct Template Reference

### Public Types

- typedef float2 **Type**

### Static Public Member Functions

- template<class S >  
static `__device__`  
`__forceinline__` Type **make** (S s)

## 2.62 `optix::VectorTypes< float, 3 >` Struct Template Reference

### Public Types

- typedef float3 **Type**

### Static Public Member Functions

- template<class S >  
static `__device__`  
`__forceinline__` Type **make** (S s)

## 2.63 `optix::VectorTypes< float, 4 >` Struct Template Reference

### Public Types

- typedef float4 **Type**

### Static Public Member Functions

- template<class S >  
static `__device__`  
`__forceinline__` Type **make** (S s)

## 2.64 `optix::VectorTypes< int, 1 >` Struct Template Reference

### Public Types

- typedef int **Type**

### Static Public Member Functions

- template<class S >  
static `__device__`  
`__forceinline__` Type **make** (S s)

## 2.65 `optix::VectorTypes< int, 2 >` Struct Template Reference

### Public Types

- typedef int2 **Type**

### Static Public Member Functions

- template<class S >  
static `__device__`  
`__forceinline__` Type **make** (S s)

## 2.66 `optix::VectorTypes< int, 3 >` Struct Template Reference

### Public Types

- typedef int3 **Type**

### Static Public Member Functions

- template<class S >  
static `__device__`  
`__forceinline__` Type **make** (S s)

## 2.67 `optix::VectorTypes< int, 4 >` Struct Template Reference

### Public Types

- typedef int4 **Type**

## Static Public Member Functions

- `template<class S >`  
`static __device__`  
`__forceinline__ Type make (S s)`

2.68 `optix::VectorTypes< unsigned int, 1 >` Struct Template Reference

## Public Types

- `typedef unsigned int Type`

## Static Public Member Functions

- `static __device__`  
`__forceinline__ Type make (unsigned int s)`
- `template<class S >`  
`static __device__`  
`__forceinline__ Type make (S s)`

2.69 `optix::VectorTypes< unsigned int, 2 >` Struct Template Reference

## Public Types

- `typedef uint2 Type`

## Static Public Member Functions

- `template<class S >`  
`static __device__`  
`__forceinline__ Type make (S s)`

2.70 `optix::VectorTypes< unsigned int, 3 >` Struct Template Reference

## Public Types

- `typedef uint3 Type`

## Static Public Member Functions

- `template<class S >`  
`static __device__`  
`__forceinline__ Type make (S s)`

2.71 `optix::VectorTypes< unsigned int, 4 >` Struct Template Reference

## Public Types

- `typedef uint4 Type`

### Static Public Member Functions

- `template<class S >`  
`static __device__`  
`__forceinline__ Type make (S s)`

## 2.72 RTPinternals\_3070::WoopTriangle Struct Reference

### Public Attributes

- `float t [4]`
- `float u [4]`
- `float v [4]`

## 3 File Documentation

### 3.1 optix.h File Reference

#### 3.1.1 Detailed Description

OptiX public API header.

#### Author

NVIDIA Corporation Includes the host api if compiling host code, includes the cuda api if compiling device code. For the math library routines include [optix\\_math.h](#)

#### Macros

- `#define OPTIX_VERSION`

#### 3.1.2 Macro Definition Documentation

##### 3.1.2.1 #define OPTIX\_VERSION

#### Value:

```
3090 /* 3.9.0 (major = OPTIX_VERSION/1000,          *
          *      minor = (OPTIX_VERSION%1000)/10,   *
          *      micro = OPTIX_VERSION%10          */
```

### 3.2 optix\_cuda\_interop.h File Reference

#### 3.2.1 Detailed Description

OptiX public API declarations CUDAInterop.

#### Author

NVIDIA Corporation OptiX public API declarations for CUDA interoperability

#### Typedefs

- `typedef unsigned int CUdeviceptr`



## Functions

- [RTresult](#) RTAPI [rtBufferCreateForCUDA](#) ([RTcontext](#) context, unsigned int bufferdesc, [RTbuffer](#) \*buffer)
- [RTresult](#) RTAPI [rtBufferGetDevicePointer](#) ([RTbuffer](#) buffer, unsigned int optix\_device\_number, void \*\*device\_pointer)
- [RTresult](#) RTAPI [rtBufferMarkDirty](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferSetDevicePointer](#) ([RTbuffer](#) buffer, unsigned int optix\_device\_number, CUdeviceptr device\_pointer)

## 3.2.2 Function Documentation

3.2.2.1 [RTresult](#) RTAPI [rtBufferCreateForCUDA](#) ( [RTcontext](#) context, unsigned int bufferdesc, [RTbuffer](#) \* buffer )

Creates a new buffer object that will later rely on user-side CUDA allocation.

**Description**

[rtBufferCreateForCUDA](#) allocates and returns a new handle to a new buffer object in \*buffer associated with context. This buffer will function like a normal OptiX buffer created with [rtBufferCreate](#), except OptiX will not allocate or upload data for it.

After a buffer object has been created with [rtBufferCreateForCUDA](#), the user needs to call [rtBufferSetDevicePointer](#) to provide one or more device pointers to the buffer data. When the user provides a single device's data pointer for a buffer prior to calling [rtContextLaunch](#), OptiX will allocate memory on the other devices and copy the data there. Setting pointers for more than one but fewer than all devices is not supported.

If [rtBufferSetDevicePointer](#) or [rtBufferGetDevicePointer](#) have been called for a single device for a given buffer, the user can change the buffer's content on that device. OptiX must then synchronize the new buffer contents to all devices. These synchronization copies occur at every [rtContextLaunch](#), unless the buffer is declared with [RT\\_BUFFER\\_COPY\\_ON\\_DIRTY](#). In this case, use [rtBufferMarkDirty](#) to notify OptiX that the buffer has been dirtied and must be synchronized.

The backing storage of the buffer is managed by OptiX. A buffer is specified by a bitwise or combination of a type and flags in bufferdesc. The supported types are:

- [RT\\_BUFFER\\_INPUT](#)
- [RT\\_BUFFER\\_OUTPUT](#)
- [RT\\_BUFFER\\_INPUT\\_OUTPUT](#)

The type values are used to specify the direction of data flow from the host to the OptiX devices. [RT\\_BUFFER\\_INPUT](#) specifies that the host may only write to the buffer and the device may only read from the buffer. [RT\\_BUFFER\\_OUTPUT](#) specifies the opposite, read only access on the host and write only access on the device. Devices and the host may read and write from buffers of type [RT\\_BUFFER\\_INPUT\\_OUTPUT](#). Reading or writing to a buffer of the incorrect type (e.g., the host writing to a buffer of type [RT\\_BUFFER\\_OUTPUT](#)) is undefined.

The supported flags are:

- [RT\\_BUFFER\\_GPU\\_LOCAL](#)
- [RT\\_BUFFER\\_COPY\\_ON\\_DIRTY](#)

Flags can be used to optimize data transfers between the host and its devices. The flag [RT\\_BUFFER\\_GPU\\_LOCAL](#) can only be used in combination with [RT\\_BUFFER\\_INPUT\\_OUTPUT](#). [RT\\_BUFFER\\_INPUT\\_OUTPUT](#) and [RT\\_BUFFER\\_GPU\\_LOCAL](#) used together specify a buffer that allows the host to **only** write, and the device to read **and** write data. The written data will be never visible on the host side.

## Parameters

in	<i>context</i>	The context to create the buffer in
in	<i>bufferdesc</i>	Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer
out	<i>buffer</i>	The return handle for the buffer object

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtBufferCreateForCUDA](#) was introduced in OptiX 3.0.

**See also** [rtBufferCreate](#), [rtBufferSetDevicePointer](#), [rtBufferMarkDirty](#), [rtBufferDestroy](#)

**3.2.2.2** **RTresult** RTAPI [rtBufferGetDevicePointer](#) ( **RTbuffer** *buffer*, unsigned int *optix\_device\_number*, void \*\*  
*device\_pointer* )

Gets the pointer to the buffer's data on the given device.

## Description

[rtBufferGetDevicePointer](#) returns the pointer to the data of *buffer* on device *optix\_device\_number* in \*\**device\_pointer*.

If [rtBufferGetDevicePointer](#) has been called for a single device for a given buffer, the user can change the buffer's content on that device. OptiX must then synchronize the new buffer contents to all devices. These synchronization copies occur at every [rtContextLaunch](#), unless the buffer is declared with [RT\\_BUFFER\\_COPY\\_ON\\_DIRTY](#). In this case, use [rtBufferMarkDirty](#) to notify OptiX that the buffer has been dirtied and must be synchronized.

## Parameters

in	<i>buffer</i>	The buffer to be queried for its device pointer
in	<i>optix_device_ - number</i>	The number of OptiX device
out	<i>device_pointer</i>	The return handle to the buffer's device pointer

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtBufferGetDevicePointer](#) was introduced in OptiX 3.0.

**See also** [rtBufferMarkDirty](#), [rtBufferSetDevicePointer](#)

**3.2.2.3** **RTresult** RTAPI [rtBufferMarkDirty](#) ( **RTbuffer** *buffer* )

Sets a buffer as dirty.

## Description

If [rtBufferSetDevicePointer](#) or [rtBufferGetDevicePointer](#) have been called for a single device for a given buffer, the user can change the buffer's content on that device. OptiX must then synchronize the new buffer contents to all

devices. These synchronization copies occur at every `rtContextLaunch`, unless the buffer is declared with `RT_BUFFER_COPY_ON_DIRTY`. In this case, use `rtBufferMarkDirty` to notify OptiX that the buffer has been dirtied and must be synchronized.

Note that `RT_BUFFER_COPY_ON_DIRTY` currently only applies to CUDA Interop buffers (buffers for which the application has a device pointer).

#### Parameters

<i>in</i>	<i>buffer</i>	The buffer to be marked dirty
-----------	---------------	-------------------------------

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

#### History

`rtBufferMarkDirty` was introduced in OptiX 3.0.

**See also** `rtBufferGetDevicePointer`, `rtBufferSetDevicePointer`, `RT_BUFFER_COPY_ON_DIRTY`

**3.2.2.4** `RTresult RTAPI rtBufferSetDevicePointer ( RTbuffer buffer, unsigned int optix_device_number, CUdeviceptr device_pointer )`

Sets the pointer to the buffer's data on the given device.

#### Description

`rtBufferSetDevicePointer` sets the pointer to the data of *buffer* on device *optix\_device\_number* to *device\_pointer*.

The buffer needs to be allocated with `rtBufferCreateForCUDA` in order for the call to `rtBufferSetDevicePointer` to be valid. Likewise, before providing a device pointer for the buffer, the application must first specify the size and format of the buffer.

If `rtBufferSetDevicePointer` has been called for a single device for a given buffer, the user can change the buffer's content on that device. OptiX must then synchronize the new buffer contents to all devices. These synchronization copies occur at every `rtContextLaunch`, unless the buffer is declared with `RT_BUFFER_COPY_ON_DIRTY`. In this case, use `rtBufferMarkDirty` to notify OptiX that the buffer has been dirtied and must be synchronized.

#### Parameters

<i>in</i>	<i>buffer</i>	The buffer for which the device pointer is to be set
<i>in</i>	<i>optix_device_number</i>	The number of OptiX device
<i>in</i>	<i>device_pointer</i>	The pointer to the data on the specified device

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_INVALID_CONTEXT`

#### History

`rtBufferSetDevicePointer` was introduced in OptiX 3.0.

**See also** `rtBufferMarkDirty`, `rtBufferGetDevicePointer`

### 3.3 optix\_d3d10\_interop.h File Reference

### 3.3.1 Detailed Description

OptiX public API declarations D3D10 interop.

#### Author

NVIDIA Corporation OptiX public API declarations for D3D10 interoperability

#### Typedefs

- typedef struct IDXGIAAdapter **IDXGIAAdapter**
- typedef struct ID3D10Device **ID3D10Device**
- typedef struct ID3D10Resource **ID3D10Resource**

#### Functions

- [RTresult](#) RTAPI [rtContextSetD3D10Device](#) ([RTcontext](#) context, ID3D10Device \*device)
- [RTresult](#) RTAPI [rtDeviceGetD3D10Device](#) (int \*device, IDXGIAAdapter \*pAdapter)
- [RTresult](#) RTAPI [rtBufferCreateFromD3D10Resource](#) ([RTcontext](#) context, unsigned int bufferdesc, ID3D10Resource \*resource, [RTbuffer](#) \*buffer)
- [RTresult](#) RTAPI [rtTextureSamplerCreateFromD3D10Resource](#) ([RTcontext](#) context, ID3D10Resource \*resource, [RTtexturesampler](#) \*textureSampler)
- [RTresult](#) RTAPI [rtBufferGetD3D10Resource](#) ([RTbuffer](#) buffer, ID3D10Resource \*\*resource)
- [RTresult](#) RTAPI [rtTextureSamplerGetD3D10Resource](#) ([RTtexturesampler](#) textureSampler, ID3D10Resource \*\*resource)
- [RTresult](#) RTAPI [rtBufferD3D10Register](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferD3D10Unregister](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtTextureSamplerD3D10Register](#) ([RTtexturesampler](#) textureSampler)
- [RTresult](#) RTAPI [rtTextureSamplerD3D10Unregister](#) ([RTtexturesampler](#) textureSampler)

### 3.3.2 Function Documentation

**3.3.2.1** [RTresult](#) RTAPI [rtBufferCreateFromD3D10Resource](#) ( [RTcontext](#) context, unsigned int bufferdesc, ID3D10Resource \* resource, [RTbuffer](#) \* buffer )

Creates a new buffer object from a D3D10 resource.

#### Description

[rtBufferCreateFromD3D10Resource](#) allocates and returns a handle to a new buffer object in \*buffer associated with context. If the allocated size of the D3D resource is 0, [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#) will be returned. Supported D3D10 buffer types are:

- ID3D10Buffer

These buffers can be used to share data with D3D10; changes of the content in buffer, either done by D3D10 or OptiX, will be reflected automatically in both APIs. If the size, or format, of a D3D10 buffer is changed, appropriate OptiX calls have to be used to update buffer accordingly. OptiX keeps only a reference to D3D10 data, when buffer is destroyed, the state of resource is unaltered.

The type of this buffer is specified by one of the following values in bufferdesc:

- [RT\\_BUFFER\\_INPUT](#)
- [RT\\_BUFFER\\_OUTPUT](#)
- [RT\\_BUFFER\\_INPUT\\_OUTPUT](#)

The type values are used to specify the direction of data flow from the host to the OptiX devices. [RT\\_BUFFER\\_INPUT](#) specifies that the host may only write to the buffer and the device may only read from the buffer. [RT\\_BUFFER\\_OUTPUT](#) specifies the opposite, read only access on the host and write only access on the device. Devices and the host may read and write from buffers of type [RT\\_BUFFER\\_INPUT\\_OUTPUT](#). Reading or writing to a buffer of the incorrect type (e.g., the host writing to a buffer of type [RT\\_BUFFER\\_OUTPUT](#)) is undefined.

Flags can be used to optimize data transfers between the host and its devices. Currently no *flags* are supported for interop buffers.

#### Parameters

in	<i>context</i>	The context to create the buffer in
in	<i>bufferdesc</i>	Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer
in	<i>resource</i>	The D3D10 resource handle for use in OptiX
out	<i>buffer</i>	The return handle for the buffer object

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtBufferCreateFromD3D10Resource](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreate](#), [rtBufferDestroy](#)

#### 3.3.2.2 RTresult RTAPI rtBufferD3D10Register ( RTbuffer buffer )

Declares a D3D10 buffer as immutable and accessible by OptiX.

#### Description

An OptiX buffer in an unregistered state can be registered to OptiX again via [rtBufferD3D10Register](#). Once registered, properties like the size of the original D3D10 resource cannot be modified anymore. Calls to the corresponding D3D10 functions will return with an error code. However, the data of the D3D10 resource can still be read and written by the appropriate D3D10 commands. When a buffer is already in a registered state [rtBufferD3D10Register](#) will return [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#). A resource must be registered in order to be used by OptiX. If a resource is not registered [RT\\_ERROR\\_INVALID\\_VALUE](#) will be returned.

#### Parameters

in	<i>buffer</i>	The handle for the buffer object
----	---------------	----------------------------------

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#)

#### History

[rtBufferD3D10Register](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D11Resource](#)

### 3.3.2.3 RTresult RTAPI rtBufferD3D10Unregister ( RTbuffer *buffer* )

Declares a D3D10 buffer as mutable and inaccessible by OptiX.

#### Description

An OptiX buffer in a registered state can be unregistered via [rtBufferD3D10Register](#). Once unregistered, properties like the size of the original D3D10 resource can be changed. As long as a resource is unregistered, OptiX will not be able to access the data and will fail with [RT\\_ERROR\\_INVALID\\_VALUE](#). When a buffer is already in an unregistered state [rtBufferD3D10Unregister](#) will return [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#).

#### Parameters

<i>in</i>	<i>buffer</i>	The handle for the buffer object
-----------	---------------	----------------------------------

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#)

#### History

[rtBufferD3D10Unregister](#) was introduced in OptiX 2.0.

See also [rtBufferCreateFromD3D11Resource](#)

### 3.3.2.4 RTresult RTAPI rtBufferGetD3D10Resource ( RTbuffer *buffer*, ID3D10Resource \*\* *resource* )

Gets the D3D10 resource associated with this buffer.

#### Description

[rtBufferGetD3D10Resource](#) stores the D3D10 resource pointer in *\*\*resource* if *buffer* was created with [rtBufferCreateFromD3D10Resource](#). If *buffer* was not created from a D3D10 resource *\*\*resource* will be 0 after the call and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned.

#### Parameters

<i>in</i>	<i>buffer</i>	The buffer to be queried for its D3D10 resource
<i>out</i>	<i>resource</i>	The return handle for the resource

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

#### History

[rtBufferGetD3D10Resource](#) was introduced in OptiX 2.0.

See also [rtBufferCreateFromD3D10Resource](#)

### 3.3.2.5 RTresult RTAPI rtContextSetD3D10Device ( RTcontext *context*, ID3D10Device \* *device* )

Binds a D3D10 device to a context and enables interop.

#### Description

[rtContextSetD3D10Device](#) binds *device* to *context* and enables D3D10 interop capabilities in *context*. This function must be executed once for *context* before any call to [rtBufferCreateFromD3D10Resource](#) or [rtTextureSamplerCreateFromD3D10Resource](#) can take place. A context can only be bound to one device. Once *device* is bound to *context*, the binding is immutable and remains upon destruction of *context*.

#### Parameters

in	<i>context</i>	The context to bind the device with
in	<i>device</i>	The D3D10 device to be used for interop with the associated context

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

#### History

[rtContextSetD3D10Device](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D10Resource](#), [rtTextureSamplerCreateFromD3D10Resource](#)

#### 3.3.2.6 RTresult RTAPI rtDeviceGetD3D10Device ( int \* *device*, IDXGIAdapter \* *pAdapter* )

Returns the OptiX device number associated with the pointer to a D3D10 adapter.

#### Description

[rtDeviceGetD3D10Device](#) returns in *device* the OptiX device ID of the adapter represented by *d3d10Device*. *d3d10Device* is a pointer returned from *D3D10CreateDeviceAndSwapChain*. In combination with [rtContextSetDevices](#), this function can be used to restrict OptiX to use only one device. The same device the D3D10 commands will be sent to.

This function is only supported on Windows platforms.

#### Parameters

in	<i>device</i>	A handle to the memory location where the OptiX device ordinal associated with <i>d3d10Device</i> will be stored
out	<i>pAdapter</i>	A pointer to an <i>ID3D10Device</i> as returned from <i>D3D10CreateDeviceAndSwapChain</i>

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtDeviceGetD3D10Device](#) was introduced in OptiX 2.5.

**See also** [rtDeviceGetDeviceCount](#)

#### 3.3.2.7 RTresult RTAPI rtTextureSamplerCreateFromD3D10Resource ( RTcontext *context*, ID3D10Resource \* *resource*, RTtexturesampler \* *textureSampler* )

Creates a new texture sampler object from a D3D10 resource.

#### Description

[rtTextureSamplerCreateFromD3D10Resource](#) allocates and returns a handle to a new texture sampler object in *\*texturesampler* associated with *context*. If the allocated size of the D3D resource is 0, [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#) will be returned. Supported D3D10 texture types are:

- ID3D10Texture1D
- ID3D10Texture2D
- ID3D10Texture3D

These texture samplers can be used to share data with D3D10; changes of the content and size of *texturesampler* done by D3D10 will be reflected automatically in OptiX. Currently texture sampler data are read only in OptiX programs. OptiX keeps only a reference to D3D10 data, when *texturesampler* is destroyed, the state of the *resource* is unaltered.

The array size and number of mipmap levels can't be changed for texture samplers that encapsulate a D3D10 resource. Furthermore no buffer objects can be queried. Please refer to the InteropTypes for a complete list of supported texture formats.

#### Parameters

in	<i>context</i>	The context to create the texture sampler in
in	<i>resource</i>	The D3D10 resource handle for use in OptiX
out	<i>textureSampler</i>	The return handle for the texture sampler object

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtTextureSamplerCreateFromD3D10Resource](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreate](#), [rtTextureSamplerDestroy](#)

#### 3.3.2.8 RTresult RTAPI rtTextureSamplerD3D10Register ( RTtexturesampler *textureSampler* )

Declares a D3D10 texture as immutable and accessible by OptiX.

#### Description

An OptiX texture sampler in an unregistered state can be registered to OptiX again via [rtTextureSamplerD3D10Register](#). Once registered, properties like the size of the original D3D10 resource cannot be modified anymore. Calls to the corresponding D3D10 functions will return with an error code. However, the data of the D3D10 resource can still be read and written by the appropriate D3D10 commands. When a texture sampler is already in a registered state [rtTextureSamplerD3D10Register](#) will return [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#). A resource must be registered in order to be used by OptiX. If a resource is not registered [RT\\_ERROR\\_INVALID\\_VALUE](#) will be returned.

#### Parameters

in	<i>textureSampler</i>	The handle for the texture object
----	-----------------------	-----------------------------------

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#)



## History

[rtTextureSamplerD3D10Register](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromD3D10Resource](#)

### 3.3.2.9 RTresult RTAPI rtTextureSamplerD3D10Unregister ( RTtexturesampler *textureSampler* )

Declares a D3D10 texture as mutable and inaccessible by OptiX.

## Description

An OptiX texture sampler in a registered state can be unregistered via [rtTextureSamplerD3D10Unregister](#). Once unregistered, properties like the size of the original D3D10 resource can be changed. As long as a resource is unregistered, OptiX will not be able to access the data and will fail with [RT\\_ERROR\\_INVALID\\_VALUE](#). When a buffer is already in an unregistered state [rtBufferD3D10Unregister](#) will return [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#).

## Parameters

in	<i>textureSampler</i>	The handle for the texture object
----	-----------------------	-----------------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#)

## History

[rtTextureSamplerD3D10Unregister](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromD3D10Resource](#)

### 3.3.2.10 RTresult RTAPI rtTextureSamplerGetD3D10Resource ( RTtexturesampler *textureSampler*, ID3D10Resource \*\**resource* )

Gets the D3D10 resource associated with this texture sampler.

## Description

[rtTextureSamplerGetD3D10Resource](#) stores the D3D10 resource pointer in *\*\*resource* if *sampler* was created with [rtTextureSamplerGetD3D10Resource](#). If *sampler* was not created from a D3D10 resource *resource* will be 0 after the call and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned

## Parameters

in	<i>textureSampler</i>	The texture sampler to be queried for its D3D10 resource
out	<i>resource</i>	The return handle for the resource

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

## History

[rtTextureSamplerGetD3D10Resource](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D10Resource](#)

## 3.4 optix\_d3d11\_interop.h File Reference

### 3.4.1 Detailed Description

OptiX public API declarations D3D11 interop.

#### Author

NVIDIA Corporation OptiX public API declarations for D3D11 interoperability

#### Typedefs

- typedef struct IDXGIAAdapter **IDXGIAAdapter**
- typedef struct ID3D11Device **ID3D11Device**
- typedef struct ID3D11Resource **ID3D11Resource**

#### Functions

- [RTresult RTAPI rtContextSetD3D11Device](#) ([RTcontext](#) context, ID3D11Device \*device)
- [RTresult RTAPI rtDeviceGetD3D11Device](#) (int \*device, IDXGIAAdapter \*pAdapter)
- [RTresult RTAPI rtBufferCreateFromD3D11Resource](#) ([RTcontext](#) context, unsigned int bufferdesc, ID3D11Resource \*resource, [RTbuffer](#) \*buffer)
- [RTresult RTAPI rtTextureSamplerCreateFromD3D11Resource](#) ([RTcontext](#) context, ID3D11Resource \*resource, [RTtexturesampler](#) \*textureSampler)
- [RTresult RTAPI rtBufferGetD3D11Resource](#) ([RTbuffer](#) buffer, ID3D11Resource \*\*resource)
- [RTresult RTAPI rtTextureSamplerGetD3D11Resource](#) ([RTtexturesampler](#) textureSampler, ID3D11Resource \*\*resource)
- [RTresult RTAPI rtBufferD3D11Register](#) ([RTbuffer](#) buffer)
- [RTresult RTAPI rtBufferD3D11Unregister](#) ([RTbuffer](#) buffer)
- [RTresult RTAPI rtTextureSamplerD3D11Register](#) ([RTtexturesampler](#) textureSampler)
- [RTresult RTAPI rtTextureSamplerD3D11Unregister](#) ([RTtexturesampler](#) textureSampler)

### 3.4.2 Function Documentation

**3.4.2.1** [RTresult RTAPI rtBufferCreateFromD3D11Resource](#) ( [RTcontext](#) context, unsigned int bufferdesc, ID3D11Resource \* resource, [RTbuffer](#) \* buffer )

Creates a new buffer object from a D3D11 resource.

#### Description

[rtBufferCreateFromD3D11Resource](#) allocates and returns a handle to a new buffer object in \*buffer associated with context. If the allocated size of the D3D resource is 0, [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#) will be returned. Supported D3D11 buffer types are:

- ID3D11Buffer

These buffers can be used to share data with D3D11; changes of the content in buffer, either done by D3D11 or OptiX, will be reflected automatically in both APIs. If the size, or format, of a D3D11 buffer is changed, appropriate OptiX calls have to be used to update buffer accordingly. OptiX keeps only a reference to D3D11 data, when buffer is destroyed, the state of resource is unaltered.

The type of this buffer is specified by one of the following values in bufferdesc:

- [RT\\_BUFFER\\_INPUT](#)
- [RT\\_BUFFER\\_OUTPUT](#)

- [RT\\_BUFFER\\_INPUT\\_OUTPUT](#)

The type values are used to specify the direction of data flow from the host to the OptiX devices. [RT\\_BUFFER\\_INPUT](#) specifies that the host may only write to the buffer and the device may only read from the buffer. [RT\\_BUFFER\\_OUTPUT](#) specifies the opposite, read only access on the host and write only access on the device. Devices and the host may read and write from buffers of type [RT\\_BUFFER\\_INPUT\\_OUTPUT](#). Reading or writing to a buffer of the incorrect type (e.g., the host writing to a buffer of type [RT\\_BUFFER\\_OUTPUT](#)) is undefined.

Flags can be used to optimize data transfers between the host and its devices. Currently no *flags* are supported for interop buffers.

#### Parameters

in	<i>context</i>	The context to create the buffer in
in	<i>bufferdesc</i>	Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer
in	<i>resource</i>	The D3D11 resource handle for use in OptiX
out	<i>buffer</i>	The return handle for the buffer object

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtBufferCreateFromD3D11Resource](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreate](#), [rtBufferDestroy](#)

#### 3.4.2.2 RTresult RTAPI rtBufferD3D11Register ( RTbuffer *buffer* )

Declares a D3D11 buffer as immutable and accessible by OptiX.

#### Description

An OptiX buffer in an unregistered state can be registered to OptiX again via [rtBufferD3D11Register](#). Once registered, properties like the size of the original D3D11 resource cannot be modified anymore. Calls to the corresponding D3D11 functions will return with an error code. However, the data of the D3D11 resource can still be read and written by the appropriate D3D11 commands. When a buffer is already in a registered state [rtBufferD3D11Register](#) will return [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#). A resource must be registered in order to be used by OptiX. If a resource is not registered [RT\\_ERROR\\_INVALID\\_VALUE](#) will be returned.

#### Parameters

in	<i>buffer</i>	The handle for the buffer object
----	---------------	----------------------------------

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#)

#### History

[rtBufferD3D11Register](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D11Resource](#)

### 3.4.2.3 RTresult RTAPI rtBufferD3D11Unregister ( RTbuffer *buffer* )

Declares a D3D11 buffer as mutable and inaccessible by OptiX.

#### Description

An OptiX buffer in a registered state can be unregistered via [rtBufferD3D11Register](#). Once unregistered, properties like the size of the original D3D11 resource can be changed. As long as a resource is unregistered, OptiX will not be able to access the data and will fail with [RT\\_ERROR\\_INVALID\\_VALUE](#). When a buffer is already in an unregistered state [rtBufferD3D11Unregister](#) will return [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#).

#### Parameters

in	<i>buffer</i>	The handle for the buffer object
----	---------------	----------------------------------

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#)

#### History

[rtBufferD3D11Unregister](#) was introduced in OptiX 2.0.

See also [rtBufferCreateFromD3D11Resource](#)

### 3.4.2.4 RTresult RTAPI rtBufferGetD3D11Resource ( RTbuffer *buffer*, ID3D11Resource \*\* *resource* )

Gets the D3D11 resource associated with this buffer.

#### Description

[rtBufferGetD3D11Resource](#) stores the D3D11 resource pointer in *\*\*resource* if *buffer* was created with [rtBufferCreateFromD3D11Resource](#). If *buffer* was not created from a D3D11 resource *\*\*resource* will be 0 after the call and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned.

#### Parameters

in	<i>buffer</i>	The buffer to be queried for its D3D11 resource
out	<i>resource</i>	The return handle for the resource

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

#### History

[rtBufferGetD3D11Resource](#) was introduced in OptiX 2.0.

See also [rtBufferCreateFromD3D11Resource](#)

### 3.4.2.5 RTresult RTAPI rtContextSetD3D11Device ( RTcontext *context*, ID3D11Device \* *device* )

Binds a D3D11 device to a context and enables interop.

#### Description

[rtContextSetD3D11Device](#) binds *device* to *context* and enables D3D11 interop capabilities in *context*. This function must be executed once for *context* before any call to [rtBufferCreateFromD3D11Resource](#) or [rtTextureSamplerCreateFromD3D11Resource](#) can take place. A context can only be bound to one device. Once *device* is bound to *context*, the binding is immutable and remains upon destruction of *context*.

#### Parameters

in	<i>context</i>	The context to bind the device with
in	<i>device</i>	The D3D11 device to be used for interop with the associated context

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

#### History

[rtContextSetD3D11Device](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D11Resource](#), [rtTextureSamplerCreateFromD3D11Resource](#)

#### 3.4.2.6 RTresult RTAPI rtDeviceGetD3D11Device ( int \* *device*, IDXGIAdapter \* *pAdapter* )

Returns the OptiX device number associated with the pointer to a D3D11 adapter.

#### Description

[rtDeviceGetD3D11Device](#) returns in *device* the OptiX device ID of the adapter represented by *D3D11Device*. *D3D11Device* is a pointer returned from *D3D11CreateDeviceAndSwapChain*. In combination with [rtContextSetDevices](#), this function can be used to restrict OptiX to use only one device. The same device the D3D11 commands will be sent to.

This function is only supported on Windows platforms.

#### Parameters

in	<i>device</i>	A handle to the memory location where the OptiX device ordinal associated with <i>D3D11Device</i> will be stored
in	<i>pAdapter</i>	A pointer to an <i>ID3D11Device</i> as returned from <i>D3D11CreateDeviceAndSwapChain</i>

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtDeviceGetD3D11Device](#) was introduced in OptiX 2.5.

**See also** [rtDeviceGetDeviceCount](#)

#### 3.4.2.7 RTresult RTAPI rtTextureSamplerCreateFromD3D11Resource ( RTcontext *context*, ID3D11Resource \* *resource*, RTtexturesampler \* *textureSampler* )

Creates a new texture sampler object from a D3D11 resource.

#### Description

[rtTextureSamplerCreateFromD3D11Resource](#) allocates and returns a handle to a new texture sampler object in *\*texturesampler* associated with *context*. If the allocated size of the D3D resource is 0, [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#) will be returned. Supported D3D11 texture types are:

- ID3D11Texture1D
- ID3D11Texture2D
- ID3D11Texture3D

These texture samplers can be used to share data with D3D11; changes of the content and size of *texturesampler* done by D3D11 will be reflected automatically in OptiX. Currently texture sampler data are read only in OptiX programs. OptiX keeps only a reference to D3D11 data, when *texturesampler* is destroyed, the state of the *resource* is unaltered.

The array size and number of mipmap levels can't be changed for texture samplers that encapsulate a D3D11 resource. Furthermore no buffer objects can be queried. Please refer to the InteropTypes for a complete list of supported texture formats.

#### Parameters

in	<i>context</i>	The context to create the texture sampler in
in	<i>resource</i>	The D3D11 resource handle for use in OptiX
out	<i>textureSampler</i>	The return handle for the texture sampler object

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtTextureSamplerCreateFromD3D11Resource](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreate](#), [rtTextureSamplerDestroy](#)

#### 3.4.2.8 RTresult RTAPI rtTextureSamplerD3D11Register ( RTtexturesampler *textureSampler* )

Declares a D3D11 texture as immutable and accessible by OptiX.

#### Description

An OptiX texture sampler in an unregistered state can be registered to OptiX again via [rtTextureSamplerD3D11Register](#). Once registered, properties like the size of the original D3D11 resource cannot be modified anymore. Calls to the corresponding D3D11 functions will return with an error code. However, the data of the D3D11 resource can still be read and written by the appropriate D3D11 commands. When a texture sampler is already in a registered state [rtTextureSamplerD3D11Register](#) will return [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#). A resource must be registered in order to be used by OptiX. If a resource is not registered [RT\\_ERROR\\_INVALID\\_VALUE](#) will be returned.

#### Parameters

in	<i>textureSampler</i>	The handle for the texture object
----	-----------------------	-----------------------------------

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#)

## History

[rtTextureSamplerD3D11Register](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromD3D11Resource](#)

### 3.4.2.9 RTresult RTAPI rtTextureSamplerD3D11Unregister ( RTtexturesampler *textureSampler* )

Declares a D3D11 texture as mutable and inaccessible by OptiX.

## Description

An OptiX texture sampler in a registered state can be unregistered via [rtTextureSamplerD3D11Unregister](#). Once unregistered, properties like the size of the original D3D11 resource can be changed. As long as a resource is unregistered, OptiX will not be able to access the data and will fail with [RT\\_ERROR\\_INVALID\\_VALUE](#). When a buffer is already in an unregistered state [rtBufferD3D11Unregister](#) will return [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#).

## Parameters

in	<i>textureSampler</i>	The handle for the texture object
----	-----------------------	-----------------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#)

## History

[rtTextureSamplerD3D11Unregister](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromD3D11Resource](#)

### 3.4.2.10 RTresult RTAPI rtTextureSamplerGetD3D11Resource ( RTtexturesampler *textureSampler*, ID3D11Resource \*\**resource* )

Gets the D3D11 resource associated with this texture sampler.

## Description

[rtTextureSamplerGetD3D11Resource](#) stores the D3D11 resource pointer in *\*\*resource* if *sampler* was created with [rtTextureSamplerGetD3D11Resource](#). If *sampler* was not created from a D3D11 resource *resource* will be 0 after the call and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned

## Parameters

in	<i>textureSampler</i>	The texture sampler to be queried for its D3D11 resource
out	<i>resource</i>	The return handle for the resource

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

## History

[rtTextureSamplerGetD3D11Resource](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D11Resource](#)

## 3.5 optix\_d3d9\_interop.h File Reference

### 3.5.1 Detailed Description

OptiX public API declarations D3D9 interop.

#### Author

NVIDIA Corporation OptiX public API declarations for D3D9 interoperability

#### Typedefs

- typedef struct [IDirect3DDevice9](#) [IDirect3DDevice9](#)
- typedef struct [IDirect3DResource9](#) [IDirect3DResource9](#)

#### Functions

- [RTresult](#) RTAPI [rtContextSetD3D9Device](#) ([RTcontext](#) context, [IDirect3DDevice9](#) \*device)
- [RTresult](#) RTAPI [rtDeviceGetD3D9Device](#) (int \*device, const char \*pszAdapterName)
- [RTresult](#) RTAPI [rtBufferCreateFromD3D9Resource](#) ([RTcontext](#) context, unsigned int bufferdesc, [IDirect3DResource9](#) \*resource, [RTbuffer](#) \*buffer)
- [RTresult](#) RTAPI [rtTextureSamplerCreateFromD3D9Resource](#) ([RTcontext](#) context, [IDirect3DResource9](#) \*resource, [RTtexturesampler](#) \*textureSampler)
- [RTresult](#) RTAPI [rtBufferGetD3D9Resource](#) ([RTbuffer](#) buffer, [IDirect3DResource9](#) \*\*resource)
- [RTresult](#) RTAPI [rtTextureSamplerGetD3D9Resource](#) ([RTtexturesampler](#) textureSampler, [IDirect3DResource9](#) \*\*pResource)
- [RTresult](#) RTAPI [rtBufferD3D9Register](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferD3D9Unregister](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtTextureSamplerD3D9Register](#) ([RTtexturesampler](#) textureSampler)
- [RTresult](#) RTAPI [rtTextureSamplerD3D9Unregister](#) ([RTtexturesampler](#) textureSampler)

### 3.5.2 Typedef Documentation

#### 3.5.2.1 typedef struct [IDirect3DDevice9](#) [IDirect3DDevice9](#)

[IDirect3DDevice9](#) structure

#### 3.5.2.2 typedef struct [IDirect3DResource9](#) [IDirect3DResource9](#)

[IDirect3DResource9](#) structure

### 3.5.3 Function Documentation

#### 3.5.3.1 [RTresult](#) RTAPI [rtBufferCreateFromD3D9Resource](#) ( [RTcontext](#) context, unsigned int bufferdesc, [IDirect3DResource9](#) \* resource, [RTbuffer](#) \* buffer )

Creates a new buffer object from a D3D9 resource.

#### Description

[rtBufferCreateFromD3D9Resource](#) allocates and returns a handle to a new buffer object in \*buffer associated with context. If the allocated size of the D3D resource is 0, [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#) will be returned. Supported D3D9 buffer types are:

- [IDirect3DVertexBuffer9](#)
- [IDirect3DIndexBuffer9](#)



These buffers can be used to share data with D3D9; changes of the content in *buffer*, either done by D3D9 or OptiX, will be reflected automatically in both APIs. If the size, or format, of a D3D9 buffer is changed, appropriate OptiX calls have to be used to update *buffer* accordingly. OptiX keeps only a reference to D3D9 data, when *buffer* is destroyed, the state of *resource* is unaltered.

The *type* of this buffer is specified by one of the following values in *bufferdesc*:

- [RT\\_BUFFER\\_INPUT](#)
- [RT\\_BUFFER\\_OUTPUT](#)
- [RT\\_BUFFER\\_INPUT\\_OUTPUT](#)

The type values are used to specify the direction of data flow from the host to the OptiX devices. [RT\\_BUFFER\\_INPUT](#) specifies that the host may only write to the buffer and the device may only read from the buffer. [RT\\_BUFFER\\_OUTPUT](#) specifies the opposite, read only access on the host and write only access on the device. Devices and the host may read and write from buffers of type [RT\\_BUFFER\\_INPUT\\_OUTPUT](#). Reading or writing to a buffer of the incorrect type (e.g., the host writing to a buffer of type [RT\\_BUFFER\\_OUTPUT](#)) is undefined.

Flags can be used to optimize data transfers between the host and it's devices. Currently no *flags* are supported for interop buffers.

#### Parameters

in	<i>context</i>	The context to create the buffer in
in	<i>bufferdesc</i>	Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer
in	<i>resource</i>	The D3D9 resource handle for use in OptiX
out	<i>buffer</i>	The return handle for the buffer object

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtBufferCreateFromD3D9Resource](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreate](#), [rtBufferDestroy](#)

#### 3.5.3.2 HRESULT WINAPI rtBufferD3D9Register ( RTbuffer buffer )

Declares a D3D9 buffer as immutable and accessible by OptiX.

#### Description

An OptiX buffer in an unregistered state can be registered to OptiX again via [rtBufferD3D9Register](#). Once registered, properties like the size of the original D3D9 resource cannot be modified anymore. Calls to the corresponding D3D9 functions will return with an error code. However, the data of the D3D9 resource can still be read and written by the appropriate D3D9 commands. When a buffer is already in a registered state [rtBufferD3D9Register](#) will return [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#). A resource must be registered in order to be used by OptiX. If a resource is not registered [RT\\_ERROR\\_INVALID\\_VALUE](#) will be returned.

#### Parameters

<i>in</i>	<i>buffer</i>	The handle for the buffer object
-----------	---------------	----------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#)

**History**

[rtBufferD3D9Register](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D11Resource](#)

**3.5.3.3 RTresult RTAPI rtBufferD3D9Unregister ( RTbuffer *buffer* )**

Declares a D3D9 buffer as mutable and inaccessible by OptiX.

**Description**

An OptiX buffer in a registered state can be unregistered via [rtBufferD3D9Register](#). Once unregistered, properties like the size of the original D3D9 resource can be changed. As long as a resource is unregistered, OptiX will not be able to access the data and will fail with [RT\\_ERROR\\_INVALID\\_VALUE](#). When a buffer is already in an unregistered state [rtBufferD3D9Unregister](#) will return [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#).

**Parameters**

<i>in</i>	<i>buffer</i>	The handle for the buffer object
-----------	---------------	----------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#)

**History**

[rtBufferD3D9Unregister](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D11Resource](#)

**3.5.3.4 RTresult RTAPI rtBufferGetD3D9Resource ( RTbuffer *buffer*, IDirect3DResource9 \*\* *resource* )**

Gets the D3D9 resource associated with this buffer.

**Description**

[rtBufferGetD3D9Resource](#) stores the D3D9 resource pointer in \*\**resource* if *buffer* was created with [rtBufferCreateFromD3D9Resource](#). If *buffer* was not created from a D3D9 resource \*\**resource* will be 0 after the call and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned.

**Parameters**

in	<i>buffer</i>	The buffer to be queried for its D3D9 resource
out	<i>resource</i>	The return handle for the resource

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

**History**

[rtBufferGetD3D9Resource](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D9Resource](#)

### 3.5.3.5 RTresult RTAPI rtContextSetD3D9Device ( RTcontext *context*, IDirect3DDevice9 \* *device* )

Binds a D3D9 device to a context and enables interop.

**Description**

[rtContextSetD3D9Device](#) binds *device* to *context* and enables D3D9 interop capabilities in *context*. This function must be executed once for *context* before any call to [rtBufferCreateFromD3D9Resource](#) or [rtTextureSamplerCreateFromD3D9Resource](#) can take place. A context can only be bound to one device. Once *device* is bound to *context*, the binding is immutable and remains upon destruction of *context*.

**Parameters**

in	<i>context</i>	The context to bind the device with
in	<i>device</i>	The D3D9 device to be used for interop with the associated context

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

**History**

[rtContextSetD3D9Device](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D9Resource](#), [rtTextureSamplerCreateFromD3D9Resource](#)

### 3.5.3.6 RTresult RTAPI rtDeviceGetD3D9Device ( int \* *device*, const char \* *pszAdapterName* )

Returns the OptiX device number associated with the specified name of a D3D9 adapter.

**Description**

[rtDeviceGetD3D9Device](#) returns in *device* the OptiX device ID of the adapter represented by *pszAdapterName*. *pszAdapterName* is the DeviceName field in the *D3DADAPTER\_IDENTIFIER9* struct. In combination with [rtContextSetDevices](#), this function can be used to restrict OptiX to use only one device. The same device the D3D9 commands will be sent to.

This function is only supported on Windows platforms.

## Parameters

in	<i>device</i>	A handle to the memory location where the OptiX device ordinal associated with <i>pszAdapterName</i> will be stored
out	<i>pszAdapterName</i>	The name of an adapter as can be found in the DeviceName field in the <i>D3D-ADAPTER_IDENTIFIER9</i> struct

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtDeviceGetD3D9Device](#) was introduced in OptiX 2.5.

See also [rtDeviceGetDeviceCount](#)

**3.5.3.7** **RTresult** RTAPI **rtTextureSamplerCreateFromD3D9Resource** ( **RTcontext** *context*, **IDirect3DResource9** \* *resource*, **RTtexturesampler** \* *textureSampler* )

Creates a new texture sampler object from a D3D9 resource.

## Description

[rtTextureSamplerCreateFromD3D9Resource](#) allocates and returns a handle to a new texture sampler object in \**texturesampler* associated with *context*. If the allocated size of the D3D resource is 0, [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#) will be returned. Supported D3D9 texture types are:

- **IDirect3DSurface9**
- (derivatives of) **IDirect3DBaseTexture9**

These texture samplers can be used to share data with D3D9; changes of the content and size of *texturesampler* done by D3D9 will be reflected automatically in OptiX. Currently texture sampler data are read only in OptiX programs. OptiX keeps only a reference to D3D9 data, when *texturesampler* is destroyed, the state of the *resource* is unaltered.

The array size and number of mipmap levels can't be changed for texture samplers that encapsulate a D3D9 resource. Furthermore no buffer objects can be queried. Please refer to the InteropTypes for a complete list of supported texture formats.

## Parameters

in	<i>context</i>	The context to create the texture sampler in
in	<i>resource</i>	The D3D9 resource handle for use in OptiX
out	<i>textureSampler</i>	The return handle for the texture sampler object

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTextureSamplerCreateFromD3D9Resource](#) was introduced in OptiX 2.0.

See also [rtTextureSamplerCreate](#), [rtTextureSamplerDestroy](#)

**3.5.3.8 RTresult RTAPI rtTextureSamplerD3D9Register ( RTtexturesampler textureSampler )**

Declares a D3D9 texture as immutable and accessible by OptiX.

**Description**

An OptiX texture sampler in an unregistered state can be registered to OptiX again via [rtTextureSamplerD3D9Register](#). Once registered, properties like the size of the original D3D9 resource cannot be modified anymore. Calls to the corresponding D3D9 functions will return with an error code. However, the data of the D3D9 resource can still be read and written by the appropriate D3D9 commands. When a texture sampler is already in a registered state [rtTextureSamplerD3D9Register](#) will return [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#). A resource must be registered in order to be used by OptiX. If a resource is not registered [RT\\_ERROR\\_INVALID\\_VALUE](#) will be returned.

**Parameters**

in	<i>textureSampler</i>	The handle for the texture object
----	-----------------------	-----------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#)

**History**

[rtTextureSamplerD3D9Register](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromD3D9Resource](#)

**3.5.3.9 RTresult RTAPI rtTextureSamplerD3D9Unregister ( RTtexturesampler textureSampler )**

Declares a D3D9 texture as mutable and inaccessible by OptiX.

**Description**

An OptiX texture sampler in a registered state can be unregistered via [rtTextureSamplerD3D9Unregister](#). Once unregistered, properties like the size of the original D3D9 resource can be changed. As long as a resource is unregistered, OptiX will not be able to access the data and will fail with [RT\\_ERROR\\_INVALID\\_VALUE](#). When a buffer is already in an unregistered state [rtBufferD3D9Unregister](#) will return [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#).

**Parameters**

in	<i>textureSampler</i>	The handle for the texture object
----	-----------------------	-----------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#)

**History**

[rtTextureSamplerD3D9Unregister](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromD3D9Resource](#)

### 3.5.3.10 RTresult RTAPI rtTextureSamplerGetD3D9Resource ( RTtexturesampler *textureSampler*, IDirect3DResource9 \*\* *pResource* )

Gets the D3D9 resource associated with this texture sampler.

#### Description

[rtTextureSamplerGetD3D9Resource](#) stores the D3D9 resource pointer in *\*\*resource* if *sampler* was created with [rtTextureSamplerGetD3D9Resource](#). If *sampler* was not created from a D3D9 resource *resource* will be 0 after the call and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned

#### Parameters

in	<i>textureSampler</i>	The texture sampler to be queried for its D3D9 resource
out	<i>pResource</i>	The return handle for the resource

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

#### History

[rtTextureSamplerGetD3D9Resource](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromD3D9Resource](#)

## 3.6 optix\_datatypes.h File Reference

### 3.6.1 Detailed Description

OptiX public API.

#### Author

NVIDIA Corporation OptiX public API Reference - Datatypes

#### Classes

- struct [Ray](#)

#### Macros

- #define [RT\\_DEFAULT\\_MAX](#) 1.e27f

#### Functions

- static \_\_inline\_\_ RT\_HOSTDEVICE [Ray make\\_Ray](#) (float3 origin, float3 direction, unsigned int ray\_type, float tmin, float tmax)

### 3.6.2 Macro Definition Documentation

#### 3.6.2.1 #define RT\_DEFAULT\_MAX 1.e27f

Max t for a ray

## 3.7 optix\_declarations.h File Reference

### 3.7.1 Detailed Description

OptiX public API declarations.

#### Author

NVIDIA Corporation OptiX public API declarations

#### Enumerations

- enum `RTformat` {  
`RT_FORMAT_UNKNOWN` = 0x100,  
`RT_FORMAT_FLOAT`,  
`RT_FORMAT_FLOAT2`,  
`RT_FORMAT_FLOAT3`,  
`RT_FORMAT_FLOAT4`,  
`RT_FORMAT_BYTE`,  
`RT_FORMAT_BYTE2`,  
`RT_FORMAT_BYTE3`,  
`RT_FORMAT_BYTE4`,  
`RT_FORMAT_UNSIGNED_BYTE`,  
`RT_FORMAT_UNSIGNED_BYTE2`,  
`RT_FORMAT_UNSIGNED_BYTE3`,  
`RT_FORMAT_UNSIGNED_BYTE4`,  
`RT_FORMAT_SHORT`,  
`RT_FORMAT_SHORT2`,  
`RT_FORMAT_SHORT3`,  
`RT_FORMAT_SHORT4`,  
`RT_FORMAT_UNSIGNED_SHORT`,  
`RT_FORMAT_UNSIGNED_SHORT2`,  
`RT_FORMAT_UNSIGNED_SHORT3`,  
`RT_FORMAT_UNSIGNED_SHORT4`,  
`RT_FORMAT_INT`,  
`RT_FORMAT_INT2`,  
`RT_FORMAT_INT3`,  
`RT_FORMAT_INT4`,  
`RT_FORMAT_UNSIGNED_INT`,  
`RT_FORMAT_UNSIGNED_INT2`,  
`RT_FORMAT_UNSIGNED_INT3`,  
`RT_FORMAT_UNSIGNED_INT4`,  
`RT_FORMAT_USER`,  
`RT_FORMAT_BUFFER_ID`,  
`RT_FORMAT_PROGRAM_ID`,  
`RT_FORMAT_HALF`,  
`RT_FORMAT_HALF2`,  
`RT_FORMAT_HALF3`,  
`RT_FORMAT_HALF4` }
- enum `RTobjecttype` {

```

RT_OBJECTTYPE_UNKNOWN = 0x200,
RT_OBJECTTYPE_GROUP,
RT_OBJECTTYPE_GEOMETRY_GROUP,
RT_OBJECTTYPE_TRANSFORM,
RT_OBJECTTYPE_SELECTOR,
RT_OBJECTTYPE_GEOMETRY_INSTANCE,
RT_OBJECTTYPE_BUFFER,
RT_OBJECTTYPE_TEXTURE_SAMPLER,
RT_OBJECTTYPE_OBJECT,
RT_OBJECTTYPE_MATRIX_FLOAT2x2,
RT_OBJECTTYPE_MATRIX_FLOAT2x3,
RT_OBJECTTYPE_MATRIX_FLOAT2x4,
RT_OBJECTTYPE_MATRIX_FLOAT3x2,
RT_OBJECTTYPE_MATRIX_FLOAT3x3,
RT_OBJECTTYPE_MATRIX_FLOAT3x4,
RT_OBJECTTYPE_MATRIX_FLOAT4x2,
RT_OBJECTTYPE_MATRIX_FLOAT4x3,
RT_OBJECTTYPE_MATRIX_FLOAT4x4,
RT_OBJECTTYPE_FLOAT,
RT_OBJECTTYPE_FLOAT2,
RT_OBJECTTYPE_FLOAT3,
RT_OBJECTTYPE_FLOAT4,
RT_OBJECTTYPE_INT,
RT_OBJECTTYPE_INT2,
RT_OBJECTTYPE_INT3,
RT_OBJECTTYPE_INT4,
RT_OBJECTTYPE_UNSIGNED_INT,
RT_OBJECTTYPE_UNSIGNED_INT2,
RT_OBJECTTYPE_UNSIGNED_INT3,
RT_OBJECTTYPE_UNSIGNED_INT4,
RT_OBJECTTYPE_USER,
RT_OBJECTTYPE_PROGRAM }

• enum RTwrapmode {
    RT_WRAP_REPEAT,
    RT_WRAP_CLAMP_TO_EDGE,
    RT_WRAP_MIRROR,
    RT_WRAP_CLAMP_TO_BORDER }

• enum RTfiltermode {
    RT_FILTER_NEAREST,
    RT_FILTER_LINEAR,
    RT_FILTER_NONE }

• enum RTtexturereadmode {
    RT_TEXTURE_READ_ELEMENT_TYPE = 0,
    RT_TEXTURE_READ_NORMALIZED_FLOAT = 1,
    RT_TEXTURE_READ_ELEMENT_TYPE_SRGB = 2,
    RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB = 3 }

• enum RTgltarget {
    RT_TARGET_GL_TEXTURE_2D,
    RT_TARGET_GL_TEXTURE_RECTANGLE,
    RT_TARGET_GL_TEXTURE_3D,
    RT_TARGET_GL_RENDER_BUFFER,
    RT_TARGET_GL_TEXTURE_1D,
    RT_TARGET_GL_TEXTURE_1D_ARRAY,
    RT_TARGET_GL_TEXTURE_2D_ARRAY,
    RT_TARGET_GL_TEXTURE_CUBE_MAP,
    RT_TARGET_GL_TEXTURE_CUBE_MAP_ARRAY }

• enum RTtextureindexmode {
    RT_TEXTURE_INDEX_NORMALIZED_COORDINATES,

```



- `RT_TEXTURE_INDEX_ARRAY_INDEX }`
- `enum RTbuffertype {`
  - `RT_BUFFER_INPUT = 0x1,`
  - `RT_BUFFER_OUTPUT = 0x2,`
  - `RT_BUFFER_INPUT_OUTPUT = RT_BUFFER_INPUT | RT_BUFFER_OUTPUT,`
  - `RT_BUFFER_PROGRESSIVE_STREAM = 0x10 }`
- `enum RTbufferflag {`
  - `RT_BUFFER_GPU_LOCAL = 0x4,`
  - `RT_BUFFER_COPY_ON_DIRTY = 0x8,`
  - `RT_BUFFER_LAYERED = 0x200000,`
  - `RT_BUFFER_CUBEMAP = 0x400000 }`
- `enum RTbuffermapflag {`
  - `RT_BUFFER_MAP_READ,`
  - `RT_BUFFER_MAP_WRITE,`
  - `RT_BUFFER_MAP_READ_WRITE,`
  - `RT_BUFFER_MAP_WRITE_DISCARD }`
- `enum RTexception {`
  - `RT_EXCEPTION_PROGRAM_ID_INVALID = 0x3EE,`
  - `RT_EXCEPTION_TEXTURE_ID_INVALID = 0x3EF,`
  - `RT_EXCEPTION_BUFFER_ID_INVALID = 0x3FA,`
  - `RT_EXCEPTION_INDEX_OUT_OF_BOUNDS = 0x3FB,`
  - `RT_EXCEPTION_STACK_OVERFLOW = 0x3FC,`
  - `RT_EXCEPTION_BUFFER_INDEX_OUT_OF_BOUNDS = 0x3FD,`
  - `RT_EXCEPTION_INVALID_RAY = 0x3FE,`
  - `RT_EXCEPTION_INTERNAL_ERROR = 0x3FF,`
  - `RT_EXCEPTION_USER = 0x400,`
  - `RT_EXCEPTION_ALL = 0x7FFFFFFF }`
- `enum RResult {`

```

RT_SUCCESS = 0,
RT_TIMEOUT_CALLBACK = 0x100,
RT_ERROR_INVALID_CONTEXT = 0x500,
RT_ERROR_INVALID_VALUE = 0x501,
RT_ERROR_MEMORY_ALLOCATION_FAILED = 0x502,
RT_ERROR_TYPE_MISMATCH = 0x503,
RT_ERROR_VARIABLE_NOT_FOUND = 0x504,
RT_ERROR_VARIABLE_REDECLARED = 0x505,
RT_ERROR_ILLEGAL_SYMBOL = 0x506,
RT_ERROR_INVALID_SOURCE = 0x507,
RT_ERROR_VERSION_MISMATCH = 0x508,
RT_ERROR_OBJECT_CREATION_FAILED = 0x600,
RT_ERROR_NO_DEVICE = 0x601,
RT_ERROR_INVALID_DEVICE = 0x602,
RT_ERROR_INVALID_IMAGE = 0x603,
RT_ERROR_FILE_NOT_FOUND = 0x604,
RT_ERROR_ALREADY_MAPPED = 0x605,
RT_ERROR_INVALID_DRIVER_VERSION = 0x606,
RT_ERROR_CONTEXT_CREATION_FAILED = 0x607,
RT_ERROR_RESOURCE_NOT_REGISTERED = 0x608,
RT_ERROR_RESOURCE_ALREADY_REGISTERED = 0x609,
RT_ERROR_LAUNCH_FAILED = 0x900,
RT_ERROR_NOT_SUPPORTED = 0xA00,
RT_ERROR_CONNECTION_FAILED = 0xB00,
RT_ERROR_AUTHENTICATION_FAILED = 0xB01,
RT_ERROR_CONNECTION_ALREADY_EXISTS = 0xB02,
RT_ERROR_NETWORK_LOAD_FAILED = 0xB03,
RT_ERROR_NETWORK_INIT_FAILED = 0xB04,
RT_ERROR_CLUSTER_NOT_RUNNING = 0xB06,
RT_ERROR_CLUSTER_ALREADY_RUNNING = 0xB07,
RT_ERROR_INSUFFICIENT_FREE_NODES = 0xB08,
RT_ERROR_UNKNOWN = ~0 }

• enum RTdeviceattribute {
    RT_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BLOCK,
    RT_DEVICE_ATTRIBUTE_CLOCK_RATE,
    RT_DEVICE_ATTRIBUTE_MULTIPROCESSOR_COUNT,
    RT_DEVICE_ATTRIBUTE_EXECUTION_TIMEOUT_ENABLED,
    RT_DEVICE_ATTRIBUTE_MAX_HARDWARE_TEXTURE_COUNT,
    RT_DEVICE_ATTRIBUTE_NAME,
    RT_DEVICE_ATTRIBUTE_COMPUTE_CAPABILITY,
    RT_DEVICE_ATTRIBUTE_TOTAL_MEMORY,
    RT_DEVICE_ATTRIBUTE_TCC_DRIVER,
    RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL }

• enum RTremotedeviceattribute {
    RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL,
    RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS,
    RT_REMOTEDEVICE_ATTRIBUTE_STATUS,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES,
    RT_REMOTEDEVICE_ATTRIBUTE_NAME,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS,
    RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY,
    RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS = 0x04000000 }

• enum RTremotedevicestatus {

```

- RT\_REMOTEDEVICE\_STATUS\_READY,
- RT\_REMOTEDEVICE\_STATUS\_CONNECTED,
- RT\_REMOTEDEVICE\_STATUS\_RESERVED,
- RT\_REMOTEDEVICE\_STATUS\_DISCONNECTED = ~0 }
- enum RTcontextattribute {
  - RT\_CONTEXT\_ATTRIBUTE\_MAX\_TEXTURE\_COUNT,
  - RT\_CONTEXT\_ATTRIBUTE\_CPU\_NUM\_THREADS,
  - RT\_CONTEXT\_ATTRIBUTE\_USED\_HOST\_MEMORY,
  - RT\_CONTEXT\_ATTRIBUTE\_GPU\_PAGING\_ACTIVE,
  - RT\_CONTEXT\_ATTRIBUTE\_GPU\_PAGING\_FORCED\_OFF,
  - RT\_CONTEXT\_ATTRIBUTE\_AVAILABLE\_DEVICE\_MEMORY = 0x10000000 }
- enum RTbufferattribute {
  - RT\_BUFFER\_ATTRIBUTE\_STREAM\_FORMAT,
  - RT\_BUFFER\_ATTRIBUTE\_STREAM\_BITRATE,
  - RT\_BUFFER\_ATTRIBUTE\_STREAM\_FPS,
  - RT\_BUFFER\_ATTRIBUTE\_STREAM\_GAMMA }
- enum RTbufferidnull { RT\_BUFFER\_ID\_NULL = 0 }
- enum RTprogramidnull { RT\_PROGRAM\_ID\_NULL = 0 }
- enum RTtextureidnull { RT\_TEXTURE\_ID\_NULL = 0 }

### 3.7.2 Enumeration Type Documentation

#### 3.7.2.1 enum RTbufferattribute

Buffer attributes

Enumerator

- RT\_BUFFER\_ATTRIBUTE\_STREAM\_FORMAT** Format string
- RT\_BUFFER\_ATTRIBUTE\_STREAM\_BITRATE** sizeof(int)
- RT\_BUFFER\_ATTRIBUTE\_STREAM\_FPS** sizeof(int)
- RT\_BUFFER\_ATTRIBUTE\_STREAM\_GAMMA** sizeof(float)

#### 3.7.2.2 enum RTbufferflag

Buffer flags

Enumerator

- RT\_BUFFER\_GPU\_LOCAL** An [RT\\_BUFFER\\_INPUT\\_OUTPUT](#) has separate copies on each device that are not synchronized
- RT\_BUFFER\_COPY\_ON\_DIRTY** A CUDA Interop buffer will only be synchronized across devices when dirtied by [rtBufferMap](#) or [rtBufferMarkDirty](#)
- RT\_BUFFER\_LAYERED** Depth specifies the number of layers, not the depth of a 3D array
- RT\_BUFFER\_CUBEMAP** Enables creation of cubemaps. If this flag is set, Width must be equal to Height, and Depth must be six. If the [RT\\_BUFFER\\_LAYERED](#) flag is also set, then Depth must be a multiple of six

#### 3.7.2.3 enum RTbufferidnull

Sentinel values

Enumerator

- RT\_BUFFER\_ID\_NULL** sentinel for describing a non-existent buffer id

## 3.7.2.4 enum RTbuffermapflag

Buffer mapping flags

Enumerator

**RT\_BUFFER\_MAP\_READ** Map buffer memory only for reading  
**RT\_BUFFER\_MAP\_WRITE** Map buffer memory only for writing  
**RT\_BUFFER\_MAP\_READ\_WRITE** Map buffer memory for both reading and writing  
**RT\_BUFFER\_MAP\_WRITE\_DISCARD** Map buffer memory for overwriting entirely

## 3.7.2.5 enum RTbuffertype

Buffer type

Enumerator

**RT\_BUFFER\_INPUT** Input buffer for the GPU  
**RT\_BUFFER\_OUTPUT** Output buffer for the GPU  
**RT\_BUFFER\_INPUT\_OUTPUT** Output/Input buffer for the GPU  
**RT\_BUFFER\_PROGRESSIVE\_STREAM** Progressive stream buffer

## 3.7.2.6 enum RTcontextattribute

Context attributes

Enumerator

**RT\_CONTEXT\_ATTRIBUTE\_MAX\_TEXTURE\_COUNT** sizeof(int)  
**RT\_CONTEXT\_ATTRIBUTE\_CPU\_NUM\_THREADS** sizeof(int)  
**RT\_CONTEXT\_ATTRIBUTE\_USED\_HOST\_MEMORY** sizeof(RTsize)  
**RT\_CONTEXT\_ATTRIBUTE\_GPU\_PAGING\_ACTIVE** sizeof(int)  
**RT\_CONTEXT\_ATTRIBUTE\_GPU\_PAGING\_FORCED\_OFF** sizeof(int)  
**RT\_CONTEXT\_ATTRIBUTE\_AVAILABLE\_DEVICE\_MEMORY** sizeof(RTsize)

## 3.7.2.7 enum RTdeviceattribute

Device attributes

Enumerator

**RT\_DEVICE\_ATTRIBUTE\_MAX\_THREADS\_PER\_BLOCK** Max Threads per Block  
**RT\_DEVICE\_ATTRIBUTE\_CLOCK\_RATE** Clock rate  
**RT\_DEVICE\_ATTRIBUTE\_MULTIPROCESSOR\_COUNT** Multiprocessor count  
**RT\_DEVICE\_ATTRIBUTE\_EXECUTION\_TIMEOUT\_ENABLED** Execution timeout enabled  
**RT\_DEVICE\_ATTRIBUTE\_MAX\_HARDWARE\_TEXTURE\_COUNT** Hardware Texture count  
**RT\_DEVICE\_ATTRIBUTE\_NAME** Attribute Name  
**RT\_DEVICE\_ATTRIBUTE\_COMPUTE\_CAPABILITY** Compute Capabilities  
**RT\_DEVICE\_ATTRIBUTE\_TOTAL\_MEMORY** Total Memory  
**RT\_DEVICE\_ATTRIBUTE\_TCC\_DRIVER** sizeof(int)  
**RT\_DEVICE\_ATTRIBUTE\_CUDA\_DEVICE\_ORDINAL** sizeof(int)

## 3.7.2.8 enum RTexception

Exceptions

Enumerator

**RT\_EXCEPTION\_PROGRAM\_ID\_INVALID** Program ID not valid  
**RT\_EXCEPTION\_TEXTURE\_ID\_INVALID** Texture ID not valid  
**RT\_EXCEPTION\_BUFFER\_ID\_INVALID** Buffer ID not valid  
**RT\_EXCEPTION\_INDEX\_OUT\_OF\_BOUNDS** Index out of bounds  
**RT\_EXCEPTION\_STACK\_OVERFLOW** Stack overflow  
**RT\_EXCEPTION\_BUFFER\_INDEX\_OUT\_OF\_BOUNDS** Buffer index out of bounds  
**RT\_EXCEPTION\_INVALID\_RAY** Invalid ray  
**RT\_EXCEPTION\_INTERNAL\_ERROR** Internal error  
**RT\_EXCEPTION\_USER** User exception  
**RT\_EXCEPTION\_ALL** All exceptions

## 3.7.2.9 enum RTfiltermode

Filter mode

Enumerator

**RT\_FILTER\_NEAREST** Nearest  
**RT\_FILTER\_LINEAR** Linear  
**RT\_FILTER\_NONE** No filter

## 3.7.2.10 enum RTformat

OptiX formats

Enumerator

**RT\_FORMAT\_UNKNOWN** Format unknown  
**RT\_FORMAT\_FLOAT** Float  
**RT\_FORMAT\_FLOAT2** sizeof(float)\*2  
**RT\_FORMAT\_FLOAT3** sizeof(float)\*3  
**RT\_FORMAT\_FLOAT4** sizeof(float)\*4  
**RT\_FORMAT\_BYTE** BYTE  
**RT\_FORMAT\_BYTE2** sizeof(CHAR)\*2  
**RT\_FORMAT\_BYTE3** sizeof(CHAR)\*3  
**RT\_FORMAT\_BYTE4** sizeof(CHAR)\*4  
**RT\_FORMAT\_UNSIGNED\_BYTE** UCHAR  
**RT\_FORMAT\_UNSIGNED\_BYTE2** sizeof(UCHAR)\*2  
**RT\_FORMAT\_UNSIGNED\_BYTE3** sizeof(UCHAR)\*3  
**RT\_FORMAT\_UNSIGNED\_BYTE4** sizeof(UCHAR)\*4  
**RT\_FORMAT\_SHORT** SHORT  
**RT\_FORMAT\_SHORT2** sizeof(SHORT)\*2  
**RT\_FORMAT\_SHORT3** sizeof(SHORT)\*3  
**RT\_FORMAT\_SHORT4** sizeof(SHORT)\*4  
**RT\_FORMAT\_UNSIGNED\_SHORT** USHORT

***RT\_FORMAT\_UNSIGNED\_SHORT2*** sizeof(USHORT)\*2  
***RT\_FORMAT\_UNSIGNED\_SHORT3*** sizeof(USHORT)\*3  
***RT\_FORMAT\_UNSIGNED\_SHORT4*** sizeof(USHORT)\*4  
***RT\_FORMAT\_INT*** INT  
***RT\_FORMAT\_INT2*** sizeof(INT)\*2  
***RT\_FORMAT\_INT3*** sizeof(INT)\*3  
***RT\_FORMAT\_INT4*** sizeof(INT)\*4  
***RT\_FORMAT\_UNSIGNED\_INT*** sizeof(UINT)  
***RT\_FORMAT\_UNSIGNED\_INT2*** sizeof(UINT)\*2  
***RT\_FORMAT\_UNSIGNED\_INT3*** sizeof(UINT)\*3  
***RT\_FORMAT\_UNSIGNED\_INT4*** sizeof(UINT)\*4  
***RT\_FORMAT\_USER*** User Format  
***RT\_FORMAT\_BUFFER\_ID*** Buffer Id  
***RT\_FORMAT\_PROGRAM\_ID*** Program Id  
***RT\_FORMAT\_HALF*** half float  
***RT\_FORMAT\_HALF2*** sizeof(half float)\*2  
***RT\_FORMAT\_HALF3*** sizeof(half float)\*3  
***RT\_FORMAT\_HALF4*** sizeof(half float)\*4

### 3.7.2.11 enum RTgltarget

GL Target

Enumerator

***RT\_TARGET\_GL\_TEXTURE\_2D*** GL texture 2D  
***RT\_TARGET\_GL\_TEXTURE\_RECTANGLE*** GL texture rectangle  
***RT\_TARGET\_GL\_TEXTURE\_3D*** GL texture 3D  
***RT\_TARGET\_GL\_RENDER\_BUFFER*** GL render buffer  
***RT\_TARGET\_GL\_TEXTURE\_1D*** GL texture 1D  
***RT\_TARGET\_GL\_TEXTURE\_1D\_ARRAY*** GL array of 1D textures  
***RT\_TARGET\_GL\_TEXTURE\_2D\_ARRAY*** GL array of 2D textures  
***RT\_TARGET\_GL\_TEXTURE\_CUBE\_MAP*** GL cube map texture  
***RT\_TARGET\_GL\_TEXTURE\_CUBE\_MAP\_ARRAY*** GL array of cube maps

### 3.7.2.12 enum RTOBJECTTYPE

OptiX Object Types

Enumerator

***RT\_OBJECTTYPE\_UNKNOWN*** Object Type Unknown  
***RT\_OBJECTTYPE\_GROUP*** Group Type  
***RT\_OBJECTTYPE\_GEOMETRY\_GROUP*** Geometry Group Type  
***RT\_OBJECTTYPE\_TRANSFORM*** Transform Type  
***RT\_OBJECTTYPE\_SELECTOR*** Selector Type  
***RT\_OBJECTTYPE\_GEOMETRY\_INSTANCE*** Geometry Instance Type  
***RT\_OBJECTTYPE\_BUFFER*** Buffer Type  
***RT\_OBJECTTYPE\_TEXTURE\_SAMPLER*** Texture Sampler Type

**RT\_OBJECTTYPE\_OBJECT** Object Type  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT2x2** Matrix Float 2x2  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT2x3** Matrix Float 2x3  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT2x4** Matrix Float 2x4  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT3x2** Matrix Float 3x2  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT3x3** Matrix Float 3x3  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT3x4** Matrix Float 3x4  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT4x2** Matrix Float 4x2  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT4x3** Matrix Float 4x3  
**RT\_OBJECTTYPE\_MATRIX\_FLOAT4x4** Matrix Float 4x4  
**RT\_OBJECTTYPE\_FLOAT** Float Type  
**RT\_OBJECTTYPE\_FLOAT2** Float2 Type  
**RT\_OBJECTTYPE\_FLOAT3** Float3 Type  
**RT\_OBJECTTYPE\_FLOAT4** Float4 Type  
**RT\_OBJECTTYPE\_INT** Integer Type  
**RT\_OBJECTTYPE\_INT2** Integer2 Type  
**RT\_OBJECTTYPE\_INT3** Integer3 Type  
**RT\_OBJECTTYPE\_INT4** Integer4 Type  
**RT\_OBJECTTYPE\_UNSIGNED\_INT** Unsigned Integer Type  
**RT\_OBJECTTYPE\_UNSIGNED\_INT2** Unsigned Integer2 Type  
**RT\_OBJECTTYPE\_UNSIGNED\_INT3** Unsigned Integer3 Type  
**RT\_OBJECTTYPE\_UNSIGNED\_INT4** Unsigned Integer4 Type  
**RT\_OBJECTTYPE\_USER** User Object Type  
**RT\_OBJECTTYPE\_PROGRAM** Object Type Program - Added in OptiX 3.0

### 3.7.2.13 enum RTprogramidnull

Enumerator

**RT\_PROGRAM\_ID\_NULL** sentinel for describing a non-existent program id

### 3.7.2.14 enum RTremotedeviceattribute

RemoteDevice attributes

Enumerator

**RT\_REMOTEDevice\_ATTRIBUTE\_CLUSTER\_URL** URL for the Cluster Manager  
**RT\_REMOTEDevice\_ATTRIBUTE\_HEAD\_NODE\_URL** URL for the Head Node  
**RT\_REMOTEDevice\_ATTRIBUTE\_NUM\_CONFIGURATIONS** Number of available configurations  
**RT\_REMOTEDevice\_ATTRIBUTE\_STATUS** Status  
**RT\_REMOTEDevice\_ATTRIBUTE\_NUM\_TOTAL\_NODES** Number of total nodes  
**RT\_REMOTEDevice\_ATTRIBUTE\_NUM\_FREE\_NODES** Number of free nodes  
**RT\_REMOTEDevice\_ATTRIBUTE\_NUM\_RESERVED\_NODES** Number of reserved nodes  
**RT\_REMOTEDevice\_ATTRIBUTE\_NAME** Name  
**RT\_REMOTEDevice\_ATTRIBUTE\_NUM\_GPUS** Number of GPUs  
**RT\_REMOTEDevice\_ATTRIBUTE\_GPU\_TOTAL\_MEMORY** Total Memory (per GPU, in bytes)  
**RT\_REMOTEDevice\_ATTRIBUTE\_CONFIGURATIONS** List of descriptions for the available configurations

## 3.7.2.15 enum RTremotedevicestatus

Enumerator

**RT\_REMOTEDEVICE\_STATUS\_READY** RemoteDevice Status Ready  
**RT\_REMOTEDEVICE\_STATUS\_CONNECTED** RemoteDevice Status Connected  
**RT\_REMOTEDEVICE\_STATUS\_RESERVED** RemoteDevice Status Reserved  
**RT\_REMOTEDEVICE\_STATUS\_DISCONNECTED** RemoteDevice Status Disconnected

## 3.7.2.16 enum RTresult

Result

Enumerator

**RT\_SUCCESS** Success  
**RT\_TIMEOUT\_CALLBACK** Timeout callback  
**RT\_ERROR\_INVALID\_CONTEXT** Invalid Context  
**RT\_ERROR\_INVALID\_VALUE** Invalid Value  
**RT\_ERROR\_MEMORY\_ALLOCATION\_FAILED** Timeout callback  
**RT\_ERROR\_TYPE\_MISMATCH** Type Mismatch  
**RT\_ERROR\_VARIABLE\_NOT\_FOUND** Variable not found  
**RT\_ERROR\_VARIABLE\_REDECLARED** Variable redeclared  
**RT\_ERROR\_ILLEGAL\_SYMBOL** Illegal symbol  
**RT\_ERROR\_INVALID\_SOURCE** Invalid source  
**RT\_ERROR\_VERSION\_MISMATCH** Version mismatch  
**RT\_ERROR\_OBJECT\_CREATION\_FAILED** Object creation failed  
**RT\_ERROR\_NO\_DEVICE** No device  
**RT\_ERROR\_INVALID\_DEVICE** Invalid device  
**RT\_ERROR\_INVALID\_IMAGE** Invalid image  
**RT\_ERROR\_FILE\_NOT\_FOUND** File not found  
**RT\_ERROR\_ALREADY\_MAPPED** Already mapped  
**RT\_ERROR\_INVALID\_DRIVER\_VERSION** Invalid driver version  
**RT\_ERROR\_CONTEXT\_CREATION\_FAILED** Context creation failed  
**RT\_ERROR\_RESOURCE\_NOT\_REGISTERED** Resource not registered  
**RT\_ERROR\_RESOURCE\_ALREADY\_REGISTERED** Resource already registered  
**RT\_ERROR\_LAUNCH\_FAILED** Launch failed  
**RT\_ERROR\_NOT\_SUPPORTED** Not supported  
**RT\_ERROR\_CONNECTION\_FAILED** Connection failed  
**RT\_ERROR\_AUTHENTICATION\_FAILED** Authentication failed  
**RT\_ERROR\_CONNECTION\_ALREADY\_EXISTS** Connection already exists  
**RT\_ERROR\_NETWORK\_LOAD\_FAILED** Network component failed to load  
**RT\_ERROR\_NETWORK\_INIT\_FAILED** Network initialization failed  
**RT\_ERROR\_CLUSTER\_NOT\_RUNNING** No cluster is running  
**RT\_ERROR\_CLUSTER\_ALREADY\_RUNNING** Cluster is already running  
**RT\_ERROR\_INSUFFICIENT\_FREE\_NODES** Not enough free nodes  
**RT\_ERROR\_UNKNOWN** Error unknown



## 3.7.2.17 enum RTtextureidnull

Enumerator

**RT\_TEXTURE\_ID\_NULL** sentinel for describing a non-existent texture id

## 3.7.2.18 enum RTtextureindexmode

Texture index mode

Enumerator

**RT\_TEXTURE\_INDEX\_NORMALIZED\_COORDINATES** Texture Index normalized coordinates

**RT\_TEXTURE\_INDEX\_ARRAY\_INDEX** Texture Index Array

## 3.7.2.19 enum RTtexturereadmode

Texture read mode

Enumerator

**RT\_TEXTURE\_READ\_ELEMENT\_TYPE** Read element type

**RT\_TEXTURE\_READ\_NORMALIZED\_FLOAT** Read normalized float

**RT\_TEXTURE\_READ\_ELEMENT\_TYPE\_SRGB** Read element type and apply sRGB to linear conversion during texture read for 8-bit integer buffer formats

**RT\_TEXTURE\_READ\_NORMALIZED\_FLOAT\_SRGB** Read normalized float and apply sRGB to linear conversion during texture read for 8-bit integer buffer formats

## 3.7.2.20 enum RTwrapmode

Wrap mode

Enumerator

**RT\_WRAP\_REPEAT** Wrap repeat

**RT\_WRAP\_CLAMP\_TO\_EDGE** Clamp to edge

**RT\_WRAP\_MIRROR** Mirror

**RT\_WRAP\_CLAMP\_TO\_BORDER** Clamp to border

## 3.8 optix\_defines.h File Reference

## 3.8.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Definitions

Classes

- struct [rti\\_internal\\_typeinfo::rti\\_typeinfo](#)
- struct [rti\\_internal\\_typeinfo::rti\\_typeenum< T >](#)

## Macros

- #define **OPTIX\_ASM\_PTR** "r"
- #define **OPTIX\_ASM\_SIZE\_T** "r"
- #define **OPTIX\_ASM\_PTR\_SIZE\_STR** "32"
- #define **OPTIX\_BITNESS\_SUFFIX** ""

## Typedefs

- typedef size\_t **optix::optix\_size\_t**

## Enumerations

- enum [rtSemanticTypes](#) {  
[\\_OPTIX\\_SEMANTIC\\_TYPE\\_LaunchIndex](#) = 0x100,  
[\\_OPTIX\\_SEMANTIC\\_TYPE\\_CurrentRay](#) = 0x200,  
[\\_OPTIX\\_SEMANTIC\\_TYPE\\_IntersectionDistance](#) = 0x300 }
- enum [RTtransformkind](#) {  
[RT\\_WORLD\\_TO\\_OBJECT](#) = 0xf00,  
[RT\\_OBJECT\\_TO\\_WORLD](#) }
- enum [RTtransformflags](#) { [RT\\_INTERNAL\\_INVERSE\\_TRANSPOSE](#) = 0x1000 }
- enum [rtiTypeKind](#) { [\\_OPTIX\\_VARIABLE](#) = 0x796152 }
- enum [rtiTypeEnum](#) {  
[\\_OPTIX\\_TYPE\\_ENUM\\_UNKNOWN](#) = 0x1337,  
[\\_OPTIX\\_TYPE\\_ENUM\\_PROGRAM\\_ID](#),  
[\\_OPTIX\\_TYPE\\_ENUM\\_PROGRAM\\_AS\\_ID](#) }
- enum [rtiTexLookupKind](#) {  
[TEX\\_LOOKUP\\_1D](#) = 1,  
[TEX\\_LOOKUP\\_2D](#) = 2,  
[TEX\\_LOOKUP\\_3D](#) = 3,  
[TEX\\_LOOKUP\\_A1](#) = 4,  
[TEX\\_LOOKUP\\_A2](#) = 5,  
[TEX\\_LOOKUP\\_CUBE](#) = 6,  
[TEX\\_LOOKUP\\_ACUBE](#) = 7 }

## 3.8.2 Enumeration Type Documentation

3.8.2.1 enum [rtSemanticTypes](#)

## Enumerator

- [\\_OPTIX\\_SEMANTIC\\_TYPE\\_LaunchIndex](#) Type [uint3](#)
- [\\_OPTIX\\_SEMANTIC\\_TYPE\\_CurrentRay](#) Type [Ray](#)
- [\\_OPTIX\\_SEMANTIC\\_TYPE\\_IntersectionDistance](#) Type [float](#)

3.8.2.2 enum [RTtransformflags](#)

## Transform flags

## Enumerator

- [RT\\_INTERNAL\\_INVERSE\\_TRANSPOSE](#) Inverse transpose flag

## 3.8.2.3 enum RTtransformkind

Transform type

Enumerator

**RT\_WORLD\_TO\_OBJECT** World to Object transformation

**RT\_OBJECT\_TO\_WORLD** Object to World transformation

## 3.9 optix\_device.h File Reference

## 3.9.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Host/Device side

Classes

- struct [optix::VectorTypes< T, Dim >](#)
- struct [optix::VectorTypes< int, 1 >](#)
- struct [optix::VectorTypes< int, 2 >](#)
- struct [optix::VectorTypes< int, 3 >](#)
- struct [optix::VectorTypes< int, 4 >](#)
- struct [optix::VectorTypes< unsigned int, 1 >](#)
- struct [optix::VectorTypes< unsigned int, 2 >](#)
- struct [optix::VectorTypes< unsigned int, 3 >](#)
- struct [optix::VectorTypes< unsigned int, 4 >](#)
- struct [optix::VectorTypes< float, 1 >](#)
- struct [optix::VectorTypes< float, 2 >](#)
- struct [optix::VectorTypes< float, 3 >](#)
- struct [optix::VectorTypes< float, 4 >](#)
- struct [rtObject](#)
- struct [rtCallableProgramSizeofWrapper< T >](#)
- struct [rtCallableProgramSizeofWrapper< void >](#)
- struct [optix::bufferId< T, Dim >](#)
- struct [optix::buffer< T, Dim >](#)
- struct [optix::buffer< T, Dim >::type< T2 >](#)
- struct [optix::bufferId< T, Dim >](#)
- class [rti\\_internal\\_callableprogram::CPArgVoid](#)
- struct [rti\\_internal\\_callableprogram::is\\_CPArgVoid< T1 >](#)
- struct [rti\\_internal\\_callableprogram::is\\_CPArgVoid< CPArgVoid >](#)
- struct [rti\\_internal\\_callableprogram::check\\_is\\_CPArgVoid< Condition, Dummy >](#)
- struct [rti\\_internal\\_callableprogram::check\\_is\\_CPArgVoid< false, IntentionalError >](#)
- class [rti\\_internal\\_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >](#)
- class [optix::callableProgramId< T >](#)
- class [optix::boundCallableProgramId< T >](#)
- struct [rti\\_internal\\_typeinfo::rti\\_typeenum< optix::callableProgramId< T > >](#)
- struct [rti\\_internal\\_typeinfo::rti\\_typeenum< optix::boundCallableProgramId< T > >](#)

## Macros

- #define [rtDeclareVariable](#)(type, name, semantic, annotation)
- #define [rtDeclareAnnotation](#)(variable, annotation)
- #define [rtCallableProgram](#)(return\_type, function\_name, parameter\_list)
- #define [rtBuffer](#) \_\_device\_\_ [optix::buffer](#)
- #define [rtBufferId](#) [optix::bufferId](#)
- #define [rtTextureSampler](#) texture
- #define [\\_OPTIX\\_TEX\\_FUNC\\_DECLARE](#)\_(FUNC, SIGNATURE, PARAMS)
- #define [RT\\_PROGRAM](#) \_\_global\_\_
- #define [RT\\_CALLABLE\\_PROGRAM](#) \_\_device\_\_ \_\_noinline\_\_
- #define [RT\\_INTERNAL\\_CALLABLE\\_PROGRAM\\_DEFS](#)(...)
- #define [RT\\_INTERNAL\\_BOUND\\_CALLABLE\\_PROGRAM\\_DEFS](#)(...)
- #define [rtCallableProgramId](#) [optix::callableProgramId](#)
- #define [rtCallableProgramX](#) [optix::boundCallableProgramId](#)

## Typedefs

- typedef unsigned int [optix::rtPICKLEDLocalPointer](#)
- typedef int [optix::rtTextureId](#)

## Functions

- static \_\_device\_\_ [rtPICKLEDLocalPointer](#) [optix::rtPICKLELocalPointer](#) (void \*p)
- static \_\_device\_\_ void \* [optix::rtUNPICKLELocalPointer](#) ([rtPICKLEDLocalPointer](#) p)
- \_\_device\_\_ int4 [optix::float4AsInt4](#) (float4 f4)
- \_\_device\_\_ uint4 [optix::float4AsUInt4](#) (float4 f4)
- template<class T >
  - static \_\_device\_\_ void [rtTrace](#) ([rtObject](#) topNode, [optix::Ray](#) ray, T &prd)
- static \_\_device\_\_ bool [rtPotentialIntersection](#) (float tmin)
- static \_\_device\_\_ bool [rtReportIntersection](#) (unsigned int material)
- static \_\_device\_\_ void [rtIgnoreIntersection](#) ()
- static \_\_device\_\_ void [rtTerminateRay](#) ()
- static \_\_device\_\_ void [rtIntersectChild](#) (unsigned int index)
- static \_\_device\_\_ float3 [rtTransformPoint](#) ([RTtransformkind](#) kind, const float3 &p)
- static \_\_device\_\_ float3 [rtTransformVector](#) ([RTtransformkind](#) kind, const float3 &v)
- static \_\_device\_\_ float3 [rtTransformNormal](#) ([RTtransformkind](#) kind, const float3 &n)
- static \_\_device\_\_ void [rtGetTransform](#) ([RTtransformkind](#) kind, float matrix[16])
- static \_\_device\_\_ void [rtThrow](#) (unsigned int code)
- static \_\_device\_\_ unsigned int [rtGetExceptionCode](#) ()
- static \_\_device\_\_ void [rtPrintExceptionDetails](#) ()
- static \_\_device\_\_ void [rtPrintf](#) (const char \*fmt)
- template<typename T1 >
  - static \_\_device\_\_ void [rtPrintf](#) (const char \*fmt, T1 arg1)
- template<typename T1, typename T2 >
  - static \_\_device\_\_ void [rtPrintf](#) (const char \*fmt, T1 arg1, T2 arg2)
- template<typename T1, typename T2, typename T3 >
  - static \_\_device\_\_ void [rtPrintf](#) (const char \*fmt, T1 arg1, T2 arg2, T3 arg3)
- template<typename T1, typename T2, typename T3, typename T4 >
  - static \_\_device\_\_ void [rtPrintf](#) (const char \*fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4)
- template<typename T1, typename T2, typename T3, typename T4, typename T5 >
  - static \_\_device\_\_ void [rtPrintf](#) (const char \*fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5)

- `template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >`  
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6)`
- `template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >`  
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7)`
- `template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 >`  
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8)`
- `template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 , typename T9 >`  
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9)`
- `template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 , typename T9 , typename T10 >`  
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10)`
- `template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 , typename T9 , typename T10 , typename T11 >`  
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11)`
- `template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 , typename T9 , typename T10 , typename T11 , typename T12 >`  
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11, T12 arg12)`
  
- `rtTextureId optix::id`
- `rtTextureId float optix::x`
- `* optix::retVal = tmp`
- `rtTextureId float float optix::y`
- `rtTextureId float float float optix::z`
- `rtTextureId float float int optix::comp`
- `rtTextureId float float optix::dPdx`
- `rtTextureId float float float optix::dPdy`
- `rtTextureId float int optix::layer`
- `rtTextureId float float optix::level`
- `__device__ uint3 optix::rtTexSize (rtTextureId id)`
- `template<typename T >`  
`__device__ T optix::rtTex1D (rtTextureId id, float x)`
- `template<>`  
`__device__ float4 optix::rtTex1D (rtTextureId id, float x)`
- `template<>`  
`__device__ int4 optix::rtTex1D (rtTextureId id, float x)`
- `template<>`  
`__device__ uint4 optix::rtTex1D (rtTextureId id, float x)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1D,(rtTextureId id, float x),(id, x)) template< typename T >`  
`inline __device__ void rtTex1D(T *retVal)`
- `template<typename T >`  
`__device__ T optix::rtTex1DFetch (rtTextureId id, int x)`
- `template<>`  
`__device__ float4 optix::rtTex1DFetch (rtTextureId id, int x)`
- `template<>`  
`__device__ int4 optix::rtTex1DFetch (rtTextureId id, int x)`
- `template<>`  
`__device__ uint4 optix::rtTex1DFetch (rtTextureId id, int x)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1DFetch,(rtTextureId id, int x),(id, x)) template< typename T >`  
`inline __device__ void rtTex1DFetch(T *retVal)`

- `template<typename T >`  
`__device__ T optix::rtTex2D (rtTextureId id, float x, float y)`
- `template<>`  
`__device__ float4 optix::rtTex2D (rtTextureId id, float x, float y)`
- `template<>`  
`__device__ int4 optix::rtTex2D (rtTextureId id, float x, float y)`
- `template<>`  
`__device__ uint4 optix::rtTex2D (rtTextureId id, float x, float y)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2D,(rtTextureId id, float x, float y),(id, x, y))` `template< type-`  
`name T > inline __device__ void rtTex2D(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex2DFetch (rtTextureId id, int x, int y)`
- `template<>`  
`__device__ float4 optix::rtTex2DFetch (rtTextureId id, int x, int y)`
- `template<>`  
`__device__ int4 optix::rtTex2DFetch (rtTextureId id, int x, int y)`
- `template<>`  
`__device__ uint4 optix::rtTex2DFetch (rtTextureId id, int x, int y)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DFetch,(rtTextureId id, int x, int y),(id, x, y))` `template< type-`  
`name T > inline __device__ void rtTex2DFetch(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex3D (rtTextureId id, float x, float y, float z)`
- `template<>`  
`__device__ float4 optix::rtTex3D (rtTextureId id, float x, float y, float z)`
- `template<>`  
`__device__ int4 optix::rtTex3D (rtTextureId id, float x, float y, float z)`
- `template<>`  
`__device__ uint4 optix::rtTex3D (rtTextureId id, float x, float y, float z)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex3D,(rtTextureId id, float x, float y, float z),(id, x, y, z))`  
`template< typename T > inline __device__ void rtTex3D(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex3DFetch (rtTextureId id, int x, int y, int z)`
- `template<>`  
`__device__ float4 optix::rtTex3DFetch (rtTextureId id, int x, int y, int z)`
- `template<>`  
`__device__ int4 optix::rtTex3DFetch (rtTextureId id, int x, int y, int z)`
- `template<>`  
`__device__ uint4 optix::rtTex3DFetch (rtTextureId id, int x, int y, int z)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex3DFetch,(rtTextureId id, int x, int y, int z),(id, x, y, z))`  
`template< typename T > inline __device__ void rtTex3DFetch(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex2DGather (rtTextureId id, float x, float y, int comp=0)`
- `template<>`  
`__device__ float4 optix::rtTex2DGather (rtTextureId id, float x, float y, int comp)`
- `template<>`  
`__device__ int4 optix::rtTex2DGather (rtTextureId id, float x, float y, int comp)`
- `template<>`  
`__device__ uint4 optix::rtTex2DGather (rtTextureId id, float x, float y, int comp)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DGather,(rtTextureId id, float x, float y, int comp),(id, x, y,`  
`comp))` `template< typename T > inline __device__ void rtTex2DGather(T *retVal`
- `template<>`  
`__device__ float4 optix::rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)`
- `template<>`  
`__device__ int4 optix::rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)`
- `template<>`  
`__device__ uint4 optix::rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)`

- **optix::OPTIX\_TEX\_FUNC\_DECLARE\_** (rtTex1DGrad,(rtTextureId id, float x, float dPdx, float dPdy),(id, x, dPdx, dPdy)) template< typename T > inline \_\_device\_\_ void rtTex1DGrad(T \*retVal
- template<typename T >  
\_\_device\_\_ T **optix::rtTex2DGrad** (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)
- template<>  
\_\_device\_\_ float4 **optix::rtTex2DGrad** (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)
- template<>  
\_\_device\_\_ int4 **optix::rtTex2DGrad** (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)
- template<>  
\_\_device\_\_ uint4 **optix::rtTex2DGrad** (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)
- **optix::OPTIX\_TEX\_FUNC\_DECLARE\_** (rtTex2DGrad,(rtTextureId id, float x, float y, float2 dPdx, float2 dPdy),(id, x, y, dPdx, dPdy)) template< typename T > inline \_\_device\_\_ void rtTex2DGrad(T \*retVal
- template<typename T >  
\_\_device\_\_ T **optix::rtTex3DGrad** (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)
- template<>  
\_\_device\_\_ float4 **optix::rtTex3DGrad** (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)
- template<>  
\_\_device\_\_ int4 **optix::rtTex3DGrad** (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)
- template<>  
\_\_device\_\_ uint4 **optix::rtTex3DGrad** (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)
- **optix::OPTIX\_TEX\_FUNC\_DECLARE\_** (rtTex3DGrad,(rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy),(id, x, y, z, dPdx, dPdy)) template< typename T > inline \_\_device\_\_ void rtTex3DGrad(T \*retVal
- template<typename T >  
\_\_device\_\_ T **optix::rtTex1DLayeredGrad** (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>  
\_\_device\_\_ float4 **optix::rtTex1DLayeredGrad** (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>  
\_\_device\_\_ int4 **optix::rtTex1DLayeredGrad** (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>  
\_\_device\_\_ uint4 **optix::rtTex1DLayeredGrad** (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- **optix::OPTIX\_TEX\_FUNC\_DECLARE\_** (rtTex1DLayeredGrad,(rtTextureId id, float x, int layer, float dPdx, float dPdy),(id, x, layer, dPdx, dPdy)) template< typename T > inline \_\_device\_\_ void rtTex1DLayeredGrad(T \*retVal
- template<typename T >  
\_\_device\_\_ T **optix::rtTex2DLayeredGrad** (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>  
\_\_device\_\_ float4 **optix::rtTex2DLayeredGrad** (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>  
\_\_device\_\_ int4 **optix::rtTex2DLayeredGrad** (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>  
\_\_device\_\_ uint4 **optix::rtTex2DLayeredGrad** (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- **optix::OPTIX\_TEX\_FUNC\_DECLARE\_** (rtTex2DLayeredGrad,(rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy),(id, x, y, layer, dPdx, dPdy)) template< typename T > inline \_\_device\_\_ void rtTex2DLayeredGrad(T \*retVal
- template<typename T >  
\_\_device\_\_ T **optix::rtTex1DLod** (rtTextureId id, float x, float level)
- template<>  
\_\_device\_\_ float4 **optix::rtTex1DLod** (rtTextureId id, float x, float level)
- template<>  
\_\_device\_\_ int4 **optix::rtTex1DLod** (rtTextureId id, float x, float level)
- template<>  
\_\_device\_\_ uint4 **optix::rtTex1DLod** (rtTextureId id, float x, float level)
- **optix::OPTIX\_TEX\_FUNC\_DECLARE\_** (rtTex1DLod,(rtTextureId id, float x, float level),(id, x, level)) template< typename T > inline \_\_device\_\_ void rtTex1DLod(T \*retVal

- `template<typename T >`  
`__device__ T optix::rtTex2DLod (rtTextureId id, float x, float y, float level)`
- `template<>`  
`__device__ float4 optix::rtTex2DLod (rtTextureId id, float x, float y, float level)`
- `template<>`  
`__device__ int4 optix::rtTex2DLod (rtTextureId id, float x, float y, float level)`
- `template<>`  
`__device__ uint4 optix::rtTex2DLod (rtTextureId id, float x, float y, float level)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DLod,(rtTextureId id, float x, float y, float level),(id, x, y, level))`** `template< typename T > inline __device__ void rtTex2DLod(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex3DLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`  
`__device__ float4 optix::rtTex3DLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`  
`__device__ int4 optix::rtTex3DLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`  
`__device__ uint4 optix::rtTex3DLod (rtTextureId id, float x, float y, float z, float level)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex3DLod,(rtTextureId id, float x, float y, float z, float level),(id, x, y, z, level))`** `template< typename T > inline __device__ void rtTex3DLod(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- `template<>`  
`__device__ float4 optix::rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- `template<>`  
`__device__ int4 optix::rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- `template<>`  
`__device__ uint4 optix::rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1DLayeredLod,(rtTextureId id, float x, int layer, float level),(id, x, layer, level))`** `template< typename T > inline __device__ void rtTex1DLayeredLod(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- `template<>`  
`__device__ float4 optix::rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- `template<>`  
`__device__ int4 optix::rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- `template<>`  
`__device__ uint4 optix::rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DLayeredLod,(rtTextureId id, float x, float y, int layer, float level),(id, x, y, layer, level))`** `template< typename T > inline __device__ void rtTex2DLayeredLod(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex1DLayered (rtTextureId id, float x, int layer)`
- `template<>`  
`__device__ float4 optix::rtTex1DLayered (rtTextureId id, float x, int layer)`
- `template<>`  
`__device__ int4 optix::rtTex1DLayered (rtTextureId id, float x, int layer)`
- `template<>`  
`__device__ uint4 optix::rtTex1DLayered (rtTextureId id, float x, int layer)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1DLayered,(rtTextureId id, float x, int layer),(id, x, layer))`** `template< typename T > inline __device__ void rtTex1DLayered(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- `template<>`  
`__device__ float4 optix::rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- `template<>`  
`__device__ int4 optix::rtTex2DLayered (rtTextureId id, float x, float y, int layer)`



- `template<>`  
`__device__ uint4 optix::rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_`** (rtTex2DLayered,(rtTextureId id, float x, float y, int layer),(id, x, y, layer)) `template< typename T > inline __device__ void rtTex2DLayered(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTexCubemap (rtTextureId id, float x, float y, float z)`
- `template<>`  
`__device__ float4 optix::rtTexCubemap (rtTextureId id, float x, float y, float z)`
- `template<>`  
`__device__ int4 optix::rtTexCubemap (rtTextureId id, float x, float y, float z)`
- `template<>`  
`__device__ uint4 optix::rtTexCubemap (rtTextureId id, float x, float y, float z)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_`** (rtTexCubemap,(rtTextureId id, float x, float y, float z),(id, x, y, z)) `template< typename T > inline __device__ void rtTexCubemap(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `template<>`  
`__device__ float4 optix::rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `template<>`  
`__device__ int4 optix::rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `template<>`  
`__device__ uint4 optix::rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_`** (rtTexCubemapLayered,(rtTextureId id, float x, float y, float z, int layer),(id, x, y, z, layer)) `template< typename T > inline __device__ void rtTexCubemapLayered(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`  
`__device__ float4 optix::rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`  
`__device__ int4 optix::rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`  
`__device__ uint4 optix::rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_`** (rtTexCubemapLod,(rtTextureId id, float x, float y, float z, float level),(id, x, y, z, level)) `template< typename T > inline __device__ void rtTexCubemapLod(T *retVal`
- `template<typename T >`  
`__device__ T optix::rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`  
`__device__ float4 optix::rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`  
`__device__ int4 optix::rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`  
`__device__ uint4 optix::rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- **`optix::OPTIX_TEX_FUNC_DECLARE_`** (rtTexCubemapLayeredLod,(rtTextureId id, float x, float y, float z, int layer, float level),(id, x, y, z, layer, level)) `template< typename T > inline __device__ void rtTexCubemapLayeredLod(T *retVal`

### 3.9.2 Macro Definition Documentation

#### 3.9.2.1 #define RT\_INTERNAL\_BOUND\_CALLABLE\_PROGRAM\_DEFS( ... )

**Value:**

```
public rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
{
    \
}
```

## 3.9.2.2 #define RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS( ... )

**Value:**

```
public rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
{
public:
    /* Default constructor */
    __device__ __forceinline__ callableProgramId() {}
    /* Constructor that initializes the id with null.*/
    __device__ __forceinline__ callableProgramId(RTprogramidnull nullid)
        : rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
          (nullid) {}
    /* Constructor that initializes the id.*/
    __device__ __forceinline__ explicit callableProgramId(int id)
        : rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
          (id) {}
    /* assignment that initializes the id with null. */
    __device__ __forceinline__ callableProgramId& operator= (RTprogramidnull nullid)
    { this->m_id = nullid; return *this; }
    /* Return the id */
    __device__ __forceinline__ int getId() const { return this->m_id; }
    /* Return whether the id is valid */
    __device__ __forceinline__ operator bool() const
    { return this->m_id != RT_PROGRAM_ID_NULL; }
}
```

## 3.9.2.3 #define RT\_PROGRAM \_\_global\_\_

Define an OptiX program.

### Description

**RT\_PROGRAM** defines a program **program\_name** with the specified arguments and return value. This function can be bound to a specific program object using [rtProgramCreateFromPTXString](#) or [rtProgramCreateFromPTXFile](#), which will subsequently get bound to different programmable binding points.

All programs should have a "void" return type. Bounding box programs will have an argument for the primitive index and the bounding box reference return value (type **nvrt::AAbb&**). Intersection programs will have a single int primitiveIndex argument. All other programs take zero arguments.

### History

**RT\_PROGRAM** was introduced in OptiX 1.0.

**See also** [RT\\_PROGRAM](#) [rtProgramCreateFromPTXFile](#) [rtProgramCreateFromPTXString](#)

## 3.9.2.4 #define rtBuffer \_\_device\_\_ optix::buffer

Declare a reference to a buffer object.

### Description

```
rtBuffer<Type, Dim> name;
```

**rtBuffer** declares a buffer of type *Type* and dimensionality *Dim*. *Dim* must be between 1 and 4 inclusive and defaults to 1 if not specified. The resulting object provides access to buffer data through the [] indexing operator, where the index is either unsigned int, uint2, uint3, or uint4 for 1, 2, 3 or 4-dimensional buffers (respectively). This operator can be used to read from or write to the resulting buffer at the specified index.

The named buffer obeys the runtime name lookup semantics as described in [rtDeclareVariable](#). A compile error will result if the named buffer is not bound to a buffer object, or is bound to a buffer object of the incorrect type or dimension. The behavior of writing to a read-only buffer is undefined. Reading from a write-only buffer is well defined only if a value has been written previously by the same thread.

This declaration must appear at the file scope (not within a function), and will be visible to all **RT\_PROGRAM** instances within the same compilation unit.

An annotation may be associated with the buffer variable by using the [rtDeclareAnnotation](#) macro.

### History

[rtBuffer](#) was introduced in OptiX 1.0.

**See also** [rtDeclareAnnotation](#), [rtDeclareVariable](#), [rtBufferCreate](#), [rtTextureSampler](#), [rtVariableSetObject](#) [rtBufferId](#)

### 3.9.2.5 #define rtBufferId optix::bufferId

A class that wraps buffer access functionality when using a buffer id.

#### Description

The [rtBufferId](#) provides an interface similar to [rtBuffer](#) when using a buffer id obtained through [rtBufferGetId](#). Unlike [rtBuffer](#), this class can be passed to functions or stored in other data structures such as the ray payload. It should be noted, however, doing so can limit the extent that OptiX can optimize the generated code.

There is also a version of [rtBufferId](#) that can be used by the host code, so that types can exist in both host and device code. See the documentation for [rtBufferId](#) found in the optix C++ API header.

#### History

[rtBufferId](#) was introduced in OptiX 3.5.

#### See also

[rtBuffer](#) [rtBufferGetId](#)

### 3.9.2.6 #define rtCallableProgram( return\_type, function\_name, parameter\_list )

#### Value:

```
namespace rti_internal_typeinfo {
    __device__ ::rti_internal_typeinfo::rti_typeinfo function_name = {
        ::rti_internal_typeinfo::_OPTIX_VARIABLE, rtCallableProgramSizeofWrapper<return_type>::value }
    ; \
}
namespace rti_internal_typename {
    __device__ char function_name[] = #return_type;
}
namespace rti_internal_semantic {
    __device__ char function_name[] = ""; /* used to be rt_call, but not needed anymore */ \
}
namespace rti_internal_annotation {
    __device__ char function_name[] = #parameter_list;
}
__noinline__ __device__ return_type function_name parameter_list { typedef return_type localtype; return
    localtype(); }
```

Callable Program Declaration.

#### Description

[rtCallableProgram](#) declares callable program *name*, which will appear to be a callable function with the specified return type and list of arguments. This callable program must be matched against a variable declared on the API object using [rtVariableSetObject](#).

Unless compatibility with SM\_10 is needed, new code should #define RT\_USE\_TEMPLATED\_RTCALLABLEPROGRAM and rely on the new templated version of [rtCallableProgram](#).

Example(s):

```
rtCallableProgram(float3, modColor, (float3, float));
```

#### Parameters

in	<i>return_type</i>	Return type of the callable program
in	<i>function_name</i>	Name of the callable program
in	<i>parameter_list</i>	Parameter_List of the callable program

#### History

[rtCallableProgram](#) was introduced in OptiX 3.0.

**See also** [rtDeclareVariable](#) [rtCallableProgramId](#) [rtCallableProgramX](#)

### 3.9.2.7 #define rtCallableProgramId optix::callableProgramId

Callable Program ID Declaration.

#### Description

[rtCallableProgramId](#) declares callable program *name*, which will appear to be a callable function with the specified return type and list of arguments. This callable program must be matched against a variable declared on the API object of type int.

Example(s):

```
rtDeclareVariable(rtCallableProgramId<float3(float3, float)>, modColor)
;
rtBuffer<rtCallableProgramId<float3(float3, float)>, 1> modColors;
```

#### History

[rtCallableProgramId](#) was introduced in OptiX 3.6.

**See also** [rtCallableProgram](#) [rtCallableProgramX](#) [rtDeclareVariable](#)

### 3.9.2.8 #define rtCallableProgramX optix::boundCallableProgramId

Callable Program X Declaration.

#### Description

[rtCallableProgramX](#) declares callable program *name*, which will appear to be a callable function with the specified return type and list of arguments. This callable program must be matched against a variable declared on the API object using [rtVariableSetObject](#).

Unless compatibility with SM\_10 is needed, new code should `#define RT_USE_TEMPLATED_RTCALLABLEPROGRAM` and rely on the new templated version of [rtCallableProgram](#) instead of directly using [rtCallableProgramX](#).

Example(s):

```
rtDeclareVariable(rtCallableProgramX<float3(float3, float)>, modColor);
// With RT_USE_TEMPLATED_RTCALLABLEPROGRAM defined
rtDeclareVariable(rtCallableProgram<float3(float3, float)>, modColor);
```

#### History

[rtCallableProgramX](#) was introduced in OptiX 3.6.

**See also** [rtCallableProgram](#) [rtCallableProgramId](#) [rtDeclareVariable](#)

### 3.9.2.9 #define rtDeclareAnnotation( variable, annotation )

**Value:**

```
namespace rti_internal_annotation { \
    __device__ char variable[] = #annotation; \
}
```

Annotation declaration.

#### Description

[rtDeclareAnnotation](#) sets the annotation *annotation* of the given variable *name*. Typically annotations are declared using an argument to [rtDeclareVariable](#), but variables of type [rtBuffer](#) and [rtTextureSampler](#) are declared using templates, so separate annotation attachment is required.

OptiX does not attempt to interpret the annotation in any way. It is considered metadata for the application to query and interpret in its own way.

#### Valid annotations

The macro [rtDeclareAnnotation](#) uses the C pre-processor's "stringification" feature to turn the literal text of the annotation argument into a string constant. The pre-processor will backslash-escape quotes and backslashes

within the text of the annotation. Leading and trailing whitespace will be ignored, and sequences of whitespace in the middle of the text is converted to a single space character in the result. The only restriction the C-PP places on the text is that it may not contain a comma character unless it is either quoted or contained within parens: "," or "(,)."

Example(s):

```
rtDeclareAnnotation( tex, this is a test );
annotation = "this is a test"

rtDeclareAnnotation( tex, "this is a test" );
annotation = "\"this is a test\""

rtDeclareAnnotation( tex, float3 a = {1, 2, 3} );
--> Compile Error, no unquoted commas may be present in the annotation

rtDeclareAnnotation( tex, "float3 a = {1, 2, 3}" );
annotation = "\"float3 a = {1, 2, 3}\""

rtDeclareAnnotation( tex, string UIWidget = "slider";
                    float UIMin = 0.0;
                    float UIMax = 1.0; );
annotation = "string UIWidget = \"slider\"; float UIMin = 0.0; float UIMax = 1.0;"
```

#### Parameters

in	<i>variable</i>	Variable to annotate
in	<i>annotation</i>	Annotation metadata

#### History

[rtDeclareAnnotation](#) was introduced in OptiX 1.0.

See also [rtDeclareVariable](#), [rtVariableGetAnnotation](#)

#### 3.9.2.10 #define rtDeclareVariable( type, name, semantic, annotation )

**Value:**

```
namespace rti_internal_typeinfo { \
    __device__ ::rti_internal_typeinfo::rti_typeinfo name = { ::rti_internal_typeinfo::OPTIX_VARIABLE, \
        sizeof(type)}; \
} \
namespace rti_internal_typename { \
    __device__ char name[] = #type; \
} \
namespace rti_internal_typeenum { \
    __device__ int name = \
        ::rti_internal_typeinfo::rti_typeenum<type>::m_typeenum \
        ; \
} \
namespace rti_internal_semantic { \
    __device__ char name[] = #semantic; \
} \
namespace rti_internal_annotation { \
    __device__ char name[] = #annotation; \
} \
__device__ type name
```

Variable declaration.

#### Description

[rtDeclareVariable](#) declares variable *name* of the specified *type*. By default, the variable name will be matched against a variable declared on the API object using the lookup hierarchy for the current program. Using the semantic-Name, this variable can be bound to internal state, to the payload associated with a ray, or to attributes that are communicated between intersection and material programs. An additional optional annotation can be used to associate application-specific metadata with the variable as well.

*type* may be a primitive type or a user-defined struct (See [rtVariableSetUserData](#)). Except for the ray payload and attributes, the declared variable will be read-only. The variable will be visible to all of the cuda functions defined in the current file. The binding of variables to values on API objects is allowed to vary from one instance to another.

#### Valid semanticNames

- **rtLaunchIndex** - The launch invocation index. Type must be one of *unsigned int*, *uint2*, *uint3*, *int*, *int2*, *int3* and is read-only.

- **rtLaunchDim** - The size of each dimension of the launch. The values range from 1 to the launch size in that dimension. Type must be one of *unsigned int*, *uint2*, *uint3*, *int*, *int2*, *int3* and is read-only.
- **rtCurrentRay** - The currently active ray, valid only when a call to [rtTrace](#) is active. Type must be *optix::Ray* and is read-only.
- **rtIntersectionDistance** - The current closest hit distance, valid only when a call to [rtTrace](#) is active. Type must be *float* and is read-only.
- **rtRayPayload** - The struct passed into the most recent [rtTrace](#) call and is read-write.
- **attribute name** - A named attribute passed from the intersection program to a closest-hit or any-hit program. The types must match in both sets of programs. This variable is read-only in the closest-hit or any-hit program and is written in the intersection program.

#### Parameters

in	<i>type</i>	Type of the variable
in	<i>name</i>	Name of the variable
in	<i>semantic</i>	Semantic name
in	<i>annotation</i>	Annotation for this variable

#### History

- [rtDeclareVariable](#) was introduced in OptiX 1.0.
- [rtLaunchDim](#) was introduced in OptiX 2.0.

**See also** [rtDeclareAnnotation](#), [rtVariableGetAnnotation](#), [rtContextDeclareVariable](#), [rtProgramDeclareVariable](#), [rtSelectorDeclareVariable](#), [rtGeometryInstanceDeclareVariable](#), [rtGeometryDeclareVariable](#), [rtMaterialDeclareVariable](#)

#### 3.9.2.11 #define rtTextureSampler texture

Declares a reference to a texture sampler object.

#### Description

[rtTextureSampler](#) declares a texture of type *Type* and dimensionality *Dim*. *Dim* must be between 1 and 3 inclusive and defaults to 1 if not specified. The resulting object provides access to texture data through the `tex1D`, `tex2D` and `tex3D` functions. These functions can be used only to read the data.

Texture filtering and wrapping modes, specified in *ReadMode* will be dependent on the state of the texture sampler object created with [rtTextureSamplerCreate](#).

An annotation may be associated with the texture sampler variable by using the [rtDeclareAnnotation](#) macro.

#### History

[rtTextureSampler](#) was introduced in OptiX 1.0.

**See also** [rtDeclareAnnotation](#), [rtTextureSamplerCreate](#)

### 3.9.3 Function Documentation

#### 3.9.3.1 static \_\_device\_\_ unsigned int rtGetExceptionCode ( ) [inline], [static]

Retrieves the type of a caught exception.

#### Description

[rtGetExceptionCode](#) can be called from an exception program to query which type of exception was caught. The returned code is equivalent to one of the [RTException](#) constants passed to [rtContextSetExceptionEnabled](#), [RT\\_EXCEPTION\\_ALL](#) excluded. For user-defined exceptions, the code is equivalent to the argument passed to [rtThrow](#).

## Return values

<i>unsigned</i>	int Returned exception code
-----------------	-----------------------------

## History

[rtGetExceptionCode](#) was introduced in OptiX 1.1.

**See also** [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtThrow](#), [rtPrintExceptionDetails](#)

**3.9.3.2** `static __device__ void rtGetTransform ( RTtransformkind kind, float matrix[16] ) [inline],[static]`

Get requested transform.

## Description

[rtGetTransform](#) returns the requested transform in the return parameter *matrix*. The type of transform to be retrieved is specified with the *kind* parameter. *kind* is an enumerated value that can be either [RT\\_OBJECT\\_TO\\_WORLD](#) or [RT\\_WORLD\\_TO\\_OBJECT](#) and must be a constant literal. During traversal, intersection and any-hit programs, the current ray will be located in object space. During ray generation, closest-hit and miss programs, the current ray will be located in world space.

There may be significant performance overhead associated with a call to [rtGetTransform](#) compared to a call to [rtTransformPoint](#), [rtTransformVector](#), or [rtTransformNormal](#).

## Parameters

<i>in</i>	<i>kind</i>	The type of transform to retrieve
<i>out</i>	<i>matrix</i>	Return parameter for the requested transform

## Return values

<i>void</i>	void return value
-------------	-------------------

## History

[rtGetTransform](#) was introduced in OptiX 1.0.

**See also** [rtTransformCreate](#), [rtTransformPoint](#), [rtTransformVector](#), [rtTransformNormal](#)

**3.9.3.3** `static __device__ void rtIgnoreIntersection ( ) [inline],[static]`

Cancels the potential intersection with current ray.

## Description

[rtIgnoreIntersection](#) causes the current potential intersection to be ignored. This intersection will not become the new closest hit associated with the ray. This function does not return, so values affecting the per-ray data should be applied before calling [rtIgnoreIntersection](#). [rtIgnoreIntersection](#) is valid only within an any-hit program.

[rtIgnoreIntersection](#) can be used to implement alpha-mapped transparency by ignoring intersections that hit the geometry but are labeled as transparent in a texture. Since any-hit programs are called frequently during intersection, care should be taken to make them as efficient as possible.

## Return values

<i>void</i>	void return value
-------------	-------------------

## History

[rtIgnoreIntersection](#) was introduced in OptiX 1.0.

**See also** [rtTerminateRay](#), [rtPotentialIntersection](#)

**3.9.3.4** `static __device__ void rtIntersectChild ( unsigned int index ) [inline],[static]`

Visit child of selector.

## Description

[rtIntersectChild](#) will perform intersection on the specified child for the current active ray. This is used in a selector visit program to traverse one of the selector's children. The *index* specifies which of the children to be visited. As the child is traversed, intersection programs will be called and any-hit programs will be called for positive intersections. When this process is complete, [rtIntersectChild](#) will return unless one of the any-hit programs calls [rtTerminateRay](#), in which case this function will never return. Multiple children can be visited during a single selector visit call by calling this function multiple times.

*index* matches the index used in [rtSelectorSetChild](#) on the host. [rtIntersectChild](#) is valid only within a selector visit program.

#### Parameters

<i>in</i>	<i>index</i>	Specifies the child to perform intersection on
-----------	--------------	--

#### Return values

<i>void</i>	void return value
-------------	-------------------

#### History

[rtIntersectChild](#) was introduced in OptiX 1.0.

**See also** [rtSelectorSetVisitProgram](#), [rtSelectorCreate](#), [rtTerminateRay](#)

**3.9.3.5** `static __device__ bool rtPotentialIntersection ( float tmin ) [inline],[static]`

Determine whether a computed intersection is potentially valid.

#### Description

Reporting an intersection from a geometry program is a two-stage process. If the geometry program computes that the ray intersects the geometry, it will first call [rtPotentialIntersection](#). [rtPotentialIntersection](#) will determine whether the reported hit distance is within the valid interval associated with the ray, and return true if the intersection is valid. Subsequently, the geometry program will compute the attributes (normal, texture coordinates, etc.) associated with the intersection before calling [rtReportIntersection](#). When [rtReportIntersection](#) is called, the any-hit program associated with the material is called. If the any-hit program does not ignore the intersection then the *t* value will stand as the new closest intersection.

If [rtPotentialIntersection](#) returns true, then [rtReportIntersection](#) should **always** be called after computing the attributes. Furthermore, attributes variables should only be written after a successful return from [rtPotentialIntersection](#).

[rtPotentialIntersection](#) is passed the material index associated with the reported intersection. Objects with a single material should pass an index of zero.

[rtReportIntersection](#) and [rtPotentialIntersection](#) are valid only within a geometry intersection program.

#### Parameters

<i>in</i>	<i>tmin</i>	t value of the ray to be checked
-----------	-------------	----------------------------------

#### Return values

<i>bool</i>	Returns whether the intersection is valid or not
-------------	--

#### History

[rtPotentialIntersection](#) was introduced in OptiX 1.0.

**See also** [rtGeometrySetIntersectionProgram](#), [rtReportIntersection](#), [rtIgnoreIntersection](#)

**3.9.3.6** `static __device__ void rtPrintExceptionDetails ( ) [inline],[static]`

Print information on a caught exception.

#### Description

[rtGetExceptionCode](#) can be called from an exception program to provide information on the caught exception to the user. The function uses [rtPrintf](#) to output details depending on the type of the exception. It is necessary to have



printing enabled using [rtContextSetPrintEnabled](#) for this function to have any effect.

## Return values

<i>void</i>	void return type
-------------	------------------

## History

[rtPrintExceptionDetails](#) was introduced in OptiX 1.1.

**See also** [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtContextSetPrintEnabled](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintf](#)

**3.9.3.7** `static __device__ void rtPrintf ( const char * fmt ) [inline],[static]`

Prints text to the standard output.

## Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

## History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

**3.9.3.8** `template<typename T1> static __device__ void rtPrintf ( const char * fmt, T1 arg1 ) [inline],[static]`

Prints text to the standard output.

## Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

## History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

**3.9.3.9** `template<typename T1 , typename T2> static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2 ) [inline],[static]`

Prints text to the standard output.

## Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

## History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.10 template<typename T1 , typename T2 , typename T3 > static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2
arg2, T3 arg3 ) [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.11 template<typename T1 , typename T2 , typename T3 , typename T4 > static __device__ void rtPrintf ( const char *
fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4 ) [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.12 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 > static __device__ void rtPrintf (
const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5 ) [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.13 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 > static __device__
        void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6 ) [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.14 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >
        static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7 )
        [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.15 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
        typename T8 > static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6
        arg6, T7 arg7, T8 arg8 ) [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.16 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
            typename T8 , typename T9 > static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4,
            T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9 ) [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.17 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
            typename T8 , typename T9 , typename T10 > static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3
            arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10 ) [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.18 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
            typename T8 , typename T9 , typename T10 , typename T11 > static __device__ void rtPrintf ( const char * fmt, T1
            arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11 ) [inline],
            [static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.19 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
typename T8 , typename T9 , typename T10 , typename T11 , typename T12 > static __device__ void rtPrintf ( const
char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11,
T12 arg12 ) [inline],[static]
```

Prints text to the standard output.

#### Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

#### History

[rtPrintf](#) was introduced in OptiX 1.0.

**See also** [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.9.3.20 static __device__ bool rtReportIntersection ( unsigned int material ) [inline],[static]
```

Report an intersection with the current object and the specified material.

#### Description

[rtReportIntersection](#) reports an intersection of the current ray with the current object, and specifies the material associated with the intersection. [rtReportIntersection](#) should only be used in conjunction with [rtPotentialIntersection](#) as described in [rtPotentialIntersection](#).

#### Parameters

in	<i>material</i>	Material associated with the intersection
----	-----------------	---

#### Return values

<i>bool</i>	return value, this is set to <i>false</i> if the intersection is, for some reason, ignored
-------------	--

#### History

[rtReportIntersection](#) was introduced in OptiX 1.0.

**See also** [rtPotentialIntersection](#), [rtIgnoreIntersection](#)

```
3.9.3.21 static __device__ void rtTerminateRay ( ) [inline],[static]
```

Terminate traversal associated with the current ray.

#### Description

[rtTerminateRay](#) causes the traversal associated with the current ray to immediately terminate. After termination, the closest-hit program associated with the ray will be called. This function does not return, so values affecting the per-ray data should be applied before calling [rtTerminateRay](#). [rtTerminateRay](#) is valid only within an any-hit program. The value of [rtIntersectionDistance](#) is undefined when [rtTerminateRay](#) is used.

#### Return values

<i>void</i>	void return value
-------------	-------------------

#### History

[rtTerminateRay](#) was introduced in OptiX 1.0.

**See also** [rtIgnoreIntersection](#), [rtPotentialIntersection](#)

3.9.3.22 `static __device__ void rtThrow ( unsigned int code ) [inline], [static]`

Throw a user exception.

#### Description

[rtThrow](#) is used to trigger user defined exceptions which behave like built-in exceptions. That is, upon invocation, ray processing for the current launch index is immediately aborted and the corresponding exception program is executed. [rtThrow](#) does not return.

The *code* passed as argument must be within the range reserved for user exceptions, which starts at [RT\\_EXCEPTION\\_USER](#) (*0x400*) and ends at *0xFFFF*. The code can be queried within the exception program using [rtGetExceptionCode](#).

[rtThrow](#) may be called from within any program type except exception programs. Calls to [rtThrow](#) will be silently ignored unless user exceptions are enabled using [rtContextSetExceptionEnabled](#).

#### History

[rtThrow](#) was introduced in OptiX 1.1.

**See also** [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtGetExceptionCode](#), [rtPrintExceptionDetails](#)

3.9.3.23 `template<class T> static __device__ void rtTrace ( rtObject topNode, optix::Ray ray, T & prd ) [inline], [static]`

Traces a ray.

#### Description

[rtTrace](#) traces *ray* against object *topNode*. A reference to *prd*, the per-ray data, will be passed to all of the closest-hit and any-hit programs that are executed during this invocation of trace. *topNode* must refer to an OptiX object of type [RTgroup](#), [RTselector](#), [RTgeometrygroup](#) or [RTtransform](#).

#### Parameters

in	<i>topNode</i>	Top node object where to start the traversal
in	<i>ray</i>	<a href="#">Ray</a> to be traced
in	<i>prd</i>	Per-ray custom data

#### Return values

<i>void</i>	void return value
-------------	-------------------

#### History

[rtTrace](#) was introduced in OptiX 1.0.

**See also** [rtObject Ray](#)

3.9.3.24 `static __device__ float3 rtTransformNormal ( RTtransformkind kind, const float3 & n ) [inline], [static]`

Apply the current transformation to a normal.

#### Description

[rtTransformNormal](#) transforms *n* as a normal using the current active transformation stack (the inverse transpose). During traversal, intersection and any-hit programs, the current ray will be located in object space. During ray generation, closest-hit and miss programs, the current ray will be located in world space. This function can be used to transform values between object and world space.

*kind* is an enumerated value that can be either [RT\\_OBJECT\\_TO\\_WORLD](#) or [RT\\_WORLD\\_TO\\_OBJECT](#) and must be a constant literal. For ray generation and miss programs, the transform will always be the identity transform. For traversal, intersection, any-hit and closest-hit programs, the transform will be dependent on the set of active transform nodes for the current state.

## Parameters

in	<i>kind</i>	Type of the transform
in	<i>n</i>	Normal to transform

## Return values

<i>float3</i>	Transformed normal
---------------	--------------------

## History

[rtTransformNormal](#) was introduced in OptiX 1.0.

**See also** [rtTransformCreate](#), [rtTransformPoint](#), [rtTransformVector](#)

```
3.9.3.25 static __device__ float3 rtTransformPoint ( RTtransformkind kind, const float3 & p ) [inline], [static]
```

Apply the current transformation to a point.

## Description

[rtTransformPoint](#) transforms *p* as a point using the current active transformation stack. During traversal, intersection and any-hit programs, the current ray will be located in object space. During ray generation, closest-hit and miss programs, the current ray will be located in world space. This function can be used to transform the ray origin and other points between object and world space.

*kind* is an enumerated value that can be either [RT\\_OBJECT\\_TO\\_WORLD](#) or [RT\\_WORLD\\_TO\\_OBJECT](#) and must be a constant literal. For ray generation and miss programs, the transform will always be the identity transform. For traversal, intersection, any-hit and closest-hit programs, the transform will be dependent on the set of active transform nodes for the current state.

## Parameters

in	<i>kind</i>	Type of the transform
in	<i>p</i>	Point to transform

## Return values

<i>float3</i>	Transformed point
---------------	-------------------

## History

[rtTransformPoint](#) was introduced in OptiX 1.0.

**See also** [rtTransformCreate](#), [rtTransformVector](#), [rtTransformNormal](#)

```
3.9.3.26 static __device__ float3 rtTransformVector ( RTtransformkind kind, const float3 & v ) [inline],
[static]
```

Apply the current transformation to a vector.

## Description

[rtTransformVector](#) transforms *v* as a vector using the current active transformation stack. During traversal, intersection and any-hit programs, the current ray will be located in object space. During ray generation, closest-hit and miss programs, the current ray will be located in world space. This function can be used to transform the ray direction and other vectors between object and world space.

*kind* is an enumerated value that can be either [RT\\_OBJECT\\_TO\\_WORLD](#) or [RT\\_WORLD\\_TO\\_OBJECT](#) and must be a constant literal. For ray generation and miss programs, the transform will always be the identity transform. For traversal, intersection, any-hit and closest-hit programs, the transform will be dependent on the set of active transform nodes for the current state.

## Parameters



<code>in</code>	<i>kind</i>	Type of the transform
<code>in</code>	<i>v</i>	Vector to transform

Return values

<i>float3</i>	Transformed vector
---------------	--------------------

## History

[rtTransformVector](#) was introduced in OptiX 1.0.

**See also** [rtTransformCreate](#), [rtTransformPoint](#), [rtTransformNormal](#)

## 3.10 `optix_gl_interop.h` File Reference

### 3.10.1 Detailed Description

OptiX public API declarations GLInterop.

#### Author

NVIDIA Corporation OptiX public API declarations for GL interoperability

#### Typedefs

- typedef void \* **HGPUNV**

#### Functions

- **RTresult** RTAPI [rtBufferCreateFromGLBO](#) (**RTcontext** context, unsigned int *bufferdesc*, unsigned int *gld*, **RTbuffer** \**buffer*)
- **RTresult** RTAPI [rtTextureSamplerCreateFromGLImage](#) (**RTcontext** context, unsigned int *gld*, **RTgltarget** target, **RTtexturesampler** \**textureSampler*)
- **RTresult** RTAPI [rtBufferGetGLBOld](#) (**RTbuffer** *buffer*, unsigned int \**gld*)
- **RTresult** RTAPI [rtTextureSamplerGetGLImageId](#) (**RTtexturesampler** *textureSampler*, unsigned int \**gld*)
- **RTresult** RTAPI [rtBufferGLRegister](#) (**RTbuffer** *buffer*)
- **RTresult** RTAPI [rtBufferGLUnregister](#) (**RTbuffer** *buffer*)
- **RTresult** RTAPI [rtTextureSamplerGLRegister](#) (**RTtexturesampler** *textureSampler*)
- **RTresult** RTAPI [rtTextureSamplerGLUnregister](#) (**RTtexturesampler** *textureSampler*)
- **RTresult** RTAPI [rtDeviceGetWGLDevice](#) (int \**device*, HGPUNV *gpu*)

### 3.10.2 Function Documentation

#### 3.10.2.1 **RTresult** RTAPI [rtBufferCreateFromGLBO](#) ( **RTcontext** *context*, unsigned int *bufferdesc*, unsigned int *gld*, **RTbuffer** \* *buffer* )

Creates a new buffer object from an OpenGL buffer object.

#### Description

[rtBufferCreateFromGLBO](#) allocates and returns a handle to a new buffer object in \**buffer* associated with *context*. Supported OpenGL buffer types are:

- Pixel Buffer Objects
- Vertex Buffer Objects

These buffers can be used to share data with OpenGL; changes of the content in *buffer*, either done by OpenGL or OptiX, will be reflected automatically in both APIs. If the size, or format, of an OpenGL buffer is changed, appropriate OptiX calls have to be used to update *buffer* accordingly. OptiX keeps only a reference to OpenGL data, when *buffer* is destroyed, the state of the *gl\_id* object is unaltered.

The *type* of this buffer is specified by one of the following values in *bufferdesc*:

- [RT\\_BUFFER\\_INPUT](#)
- [RT\\_BUFFER\\_OUTPUT](#)
- [RT\\_BUFFER\\_INPUT\\_OUTPUT](#)

The type values are used to specify the direction of data flow from the host to the OptiX devices. [RT\\_BUFFER\\_INPUT](#) specifies that the host may only write to the buffer and the device may only read from the buffer. [RT\\_BUFFER\\_OUTPUT](#) specifies the opposite, read only access on the host and write only access on the device. Devices and the host may read and write from buffers of type [RT\\_BUFFER\\_INPUT\\_OUTPUT](#). Reading or writing to a buffer of the incorrect type (e.g., the host writing to a buffer of type [RT\\_BUFFER\\_OUTPUT](#)) is undefined.

Flags can be used to optimize data transfers between the host and it's devices. Currently no *flags* are supported for interop buffers.

#### Parameters

in	<i>context</i>	The context to create the buffer in
in	<i>bufferdesc</i>	Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer
in	<i>glId</i>	The OpenGL image object resource handle for use in OptiX
out	<i>buffer</i>	The return handle for the buffer object

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtBufferCreateFromGLBO](#) was introduced in OptiX 1.0.

See also [rtBufferCreate](#), [rtBufferDestroy](#)

#### 3.10.2.2 `RTresult RTAPI rtBufferGetGLBOid ( RTbuffer buffer, unsigned int * glId )`

Gets the OpenGL Buffer Object ID associated with this buffer.

#### Description

[rtBufferGetGLBOid](#) stores the OpenGL buffer object id in *\*gl\_id* if *buffer* was created with [rtBufferCreateFromGLBO](#). If *buffer* was not created from an OpenGL Buffer Object *\*gl\_id* will be 0 after the call and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned.

#### Parameters

in	<i>buffer</i>	The buffer to be queried for its OpenGL buffer object id
in	<i>glId</i>	The return handle for the id

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)

- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtBufferGetGLBOld](#) was introduced in OptiX 1.0.

**See also** [rtBufferCreateFromGLBO](#)

#### 3.10.2.3 `RTresult RTAPI rtBufferGLRegister ( RTbuffer buffer )`

Declares an OpenGL buffer as immutable and accessible by OptiX.

### Description

Once registered, properties like the size of the original GL buffer cannot be modified anymore. Calls to the corresponding GL functions will return with an error code. However, the buffer data of the GL buffer can still be read and written by the appropriate GL commands. Returns `RT_ERROR_RESOURCE_ALREADY_REGISTERED` if *buffer* is already registered. A buffer object must be registered in order to be used by OptiX. If a buffer object is not registered [RT\\_ERROR\\_INVALID\\_VALUE](#) will be returned. An OptiX buffer in a registered state can be unregistered via [rtBufferGLRegister](#).

### Parameters

<code>in</code>	<i>buffer</i>	The handle for the buffer object
-----------------	---------------	----------------------------------

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#)

### History

[rtBufferGLRegister](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromGLBO](#), [rtBufferGLUnregister](#)

#### 3.10.2.4 `RTresult RTAPI rtBufferGLUnregister ( RTbuffer buffer )`

Declares an OpenGL buffer as mutable and inaccessible by OptiX.

### Description

Once unregistered, properties like the size of the original GL buffer can be changed. As long as a buffer object is unregistered, OptiX will not be able to access the data and calls will fail with [RT\\_ERROR\\_INVALID\\_VALUE](#). Returns `RT_ERROR_RESOURCE_NOT_REGISTERED` if *buffer* is already unregistered. An OptiX buffer in an unregistered state can be registered to OptiX again via [rtBufferGLRegister](#).

### Parameters

<code>in</code>	<i>buffer</i>	The handle for the buffer object
-----------------	---------------	----------------------------------

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#)

### History

[rtBufferGLUnregister](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromGLBO](#), [rtBufferGLRegister](#)

#### 3.10.2.5 `RTresult RTAPI rtDeviceGetWGLDevice ( int * device, HGPUNV gpu )`

returns the OptiX device number associated with the specified GPU

### Description

[rtDeviceGetWGLDevice](#) returns in *device* the OptiX device ID of the GPU represented by *gpu*. *gpu* is returned from `WGL_NV_gpu_affinity`, an OpenGL extension. This enables OptiX to create a context on the same GPU that OpenGL commands will be sent to, improving OpenGL interoperation efficiency.

### Parameters

out	<i>device</i>	A handle to the memory location where the OptiX device ordinal associated with <i>gpu</i> will be stored
in	<i>gpu</i>	A handle to a GPU as returned from the <code>WGL_NV_gpu_affinity</code> OpenGL extension

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtDeviceGetWGLDevice](#) was introduced in OptiX 1.0.

**See also** [rtDeviceGetDeviceCount](#), `WGL_NV_gpu_affinity`

#### 3.10.2.6 `RTresult RTAPI rtTextureSamplerCreateFromGLImage ( RTcontext context, unsigned int glId, RTgltarget target, RTtexturesampler * textureSampler )`

Creates a new texture sampler object from an OpenGL image.

### Description

[rtTextureSamplerCreateFromGLImage](#) allocates and returns a handle to a new texture sampler object in \* *textureSampler* associated with *context*. If the allocated size of the GL texture is 0, [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#) will be returned. Supported OpenGL image types are:

### Renderbuffers

- `GL_TEXTURE_2D`
- `GL_TEXTURE_2D_RECT`
- `GL_TEXTURE_3D`

These types are reflected by *target*:

- [RT\\_TARGET\\_GL\\_RENDER\\_BUFFER](#)
- [RT\\_TARGET\\_GL\\_TEXTURE\\_1D](#)
- [RT\\_TARGET\\_GL\\_TEXTURE\\_2D](#)

- [RT\\_TARGET\\_GL\\_TEXTURE\\_RECTANGLE](#)
- [RT\\_TARGET\\_GL\\_TEXTURE\\_3D](#)
- [RT\\_TARGET\\_GL\\_TEXTURE\\_1D\\_ARRAY](#)
- [RT\\_TARGET\\_GL\\_TEXTURE\\_2D\\_ARRAY](#)
- [RT\\_TARGET\\_GL\\_TEXTURE\\_CUBE\\_MAP](#)
- [RT\\_TARGET\\_GL\\_TEXTURE\\_CUBE\\_MAP\\_ARRAY](#)

Supported attachment points for renderbuffers are:

- `GL_COLOR_ATTACHMENT<NUM>`

These texture samplers can be used to share data with OpenGL; changes of the content and size of *texturesampler* done by OpenGL will be reflected automatically in OptiX. Currently texture sampler data are read only in OptiX programs. OptiX keeps only a reference to OpenGL data, when *texturesampler* is destroyed, the state of the *gl\_id* image is unaltered.

The array size and number of mipmap levels can't be changed for texture samplers that encapsulate a GL image. Furthermore no buffer objects can be queried.

Currently OptiX supports only a limited number of internal OpenGL texture formats. Texture formats with an internal type of float, e.g. *GL\_RGBA32F*, and many integer formats are supported. Depth formats as well as multisample buffers are also currently not supported. Please refer to the InteropTypes section for a complete list of supported texture formats.

#### Parameters

in	<i>context</i>	The context to create the buffer in
in	<i>glId</i>	The OpenGL image object resource handle for use in OptiX
in	<i>target</i>	The OpenGL target
out	<i>textureSampler</i>	The return handle for the texture sampler object

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtTextureSamplerCreateFromGLImage](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreate](#), [rtTextureSamplerDestroy](#)

#### 3.10.2.7 RTresult RTAPI rtTextureSamplerGetGLImageId ( RTtexturesampler *textureSampler*, unsigned int \* *glId* )

Gets the OpenGL image object id associated with this texture sampler.

#### Description

[rtTextureSamplerGetGLImageId](#) stores the OpenGL image object id in *\*gl\_id* if *textureSampler* was created with [rtTextureSamplerCreateFromGLImage](#). If *textureSampler* was not created from an OpenGL image object *gl\_id* will be 0 after the call and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned.

## Parameters

in	<i>textureSampler</i>	The texture sampler to be queried for its OpenGL buffer object id
in	<i>gld</i>	The return handle for the id

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTextureSamplerGetGLImageId](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromGLImage](#)

3.10.2.8 `RTresult RTAPI rtTextureSamplerGLRegister ( RTtexturesampler textureSampler )`

Declares an OpenGL texture as immutable and accessible by OptiX.

## Description

Registers an OpenGL texture as accessible by OptiX. Once registered, properties like the size of the original GL texture cannot be modified anymore. Calls to the corresponding GL functions will return with an error code. However, the pixel data of the GL texture can still be read and written by the appropriate GL commands. Returns [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#) if *textureSampler* is already registered. A texture sampler must be registered in order to be used by OptiX. Otherwise, [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned. An OptiX texture sampler in a registered state can be unregistered via [rtTextureSamplerGLUnregister](#).

## Parameters

in	<i>textureSampler</i>	The handle for the texture object
----	-----------------------	-----------------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_RESOURCE\\_ALREADY\\_REGISTERED](#)

## History

[rtTextureSamplerGLRegister](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromGLImage](#), [rtTextureSamplerGLUnregister](#)

3.10.2.9 `RTresult RTAPI rtTextureSamplerGLUnregister ( RTtexturesampler textureSampler )`

Declares an OpenGL texture as mutable and inaccessible by OptiX.

## Description

Once unregistered, properties like the size of the original GL texture can be changed. As long as a texture is unregistered, OptiX will not be able to access the pixel data and calls will fail with [RT\\_ERROR\\_INVALID\\_VALUE](#). Returns [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#) if *textureSampler* is already unregistered. An OptiX texture sampler in an unregistered state can be registered to OptiX again via [rtTextureSamplerGLRegister](#).

## Parameters

<i>in</i>	<i>textureSampler</i>	The handle for the texture object
-----------	-----------------------	-----------------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_RESOURCE\\_NOT\\_REGISTERED](#)

## History

[rtTextureSamplerGLUnregister](#) was introduced in OptiX 2.0.

**See also** [rtTextureSamplerCreateFromGLImage](#), [rtTextureSamplerGLRegister](#)

## 3.11 optix\_host.h File Reference

## 3.11.1 Detailed Description

OptiX public API.

## Author

NVIDIA Corporation OptiX public API Reference - Host side

## Macros

- `#define RTAPI __declspec(dllimport)`

## Typedefs

- `typedef unsigned int RTsize`
- `typedef struct RTacceleration_api * RTacceleration`
- `typedef struct RTbuffer_api * RTbuffer`
- `typedef struct RTcontext_api * RTcontext`
- `typedef struct RTgeometry_api * RTgeometry`
- `typedef struct`  
`RTgeometryinstance_api * RTgeometryinstance`
- `typedef struct`  
`RTgeometrygroup_api * RTgeometrygroup`
- `typedef struct RTgroup_api * RTgroup`
- `typedef struct RTmaterial_api * RTmaterial`
- `typedef struct RTprogram_api * RTprogram`
- `typedef struct RTselector_api * RTselector`
- `typedef struct`  
`RTtexturesampler_api * RTtexturesampler`
- `typedef struct RTtransform_api * RTtransform`
- `typedef struct RTvariable_api * RTvariable`
- `typedef void * RTobject`
- `typedef struct RTremotedevice_api * RTremotedevice`
- `typedef int(* RTtimeoutcallback )(void)`

## Functions

- [RTresult RTAPI rtGetVersion](#) (unsigned int \*version)
- [RTresult RTAPI rtDeviceGetDeviceCount](#) (unsigned int \*count)
- [RTresult RTAPI rtDeviceGetAttribute](#) (int ordinal, [RTdeviceattribute](#) attrib, RTsize size, void \*p)
- [RTresult RTAPI rtVariableSetObject](#) ([RTvariable](#) v, [RTObject](#) object)
- [RTresult RTAPI rtVariableSetUserData](#) ([RTvariable](#) v, RTsize size, const void \*ptr)
- [RTresult RTAPI rtVariableGetObject](#) ([RTvariable](#) v, [RTObject](#) \*object)
- [RTresult RTAPI rtVariableGetUserData](#) ([RTvariable](#) v, RTsize size, void \*ptr)
- [RTresult RTAPI rtVariableGetName](#) ([RTvariable](#) v, const char \*\*name\_return)
- [RTresult RTAPI rtVariableGetAnnotation](#) ([RTvariable](#) v, const char \*\*annotation\_return)
- [RTresult RTAPI rtVariableGetType](#) ([RTvariable](#) v, [RTObjecttype](#) \*type\_return)
- [RTresult RTAPI rtVariableGetContext](#) ([RTvariable](#) v, [RTcontext](#) \*context)
- [RTresult RTAPI rtVariableGetSize](#) ([RTvariable](#) v, RTsize \*size)
- [RTresult RTAPI rtContextCreate](#) ([RTcontext](#) \*context)
- [RTresult RTAPI rtContextDestroy](#) ([RTcontext](#) context)
- [RTresult RTAPI rtContextValidate](#) ([RTcontext](#) context)
- void RTAPI [rtContextGetErrorString](#) ([RTcontext](#) context, [RTresult](#) code, const char \*\*return\_string)
- [RTresult RTAPI rtContextSetAttribute](#) ([RTcontext](#) context, [RTcontextattribute](#) attrib, RTsize size, void \*p)
- [RTresult RTAPI rtContextGetAttribute](#) ([RTcontext](#) context, [RTcontextattribute](#) attrib, RTsize size, void \*p)
- [RTresult RTAPI rtContextSetDevices](#) ([RTcontext](#) context, unsigned int count, const int \*devices)
- [RTresult RTAPI rtContextGetDevices](#) ([RTcontext](#) context, int \*devices)
- [RTresult RTAPI rtContextGetDeviceCount](#) ([RTcontext](#) context, unsigned int \*count)
- [RTresult RTAPI rtContextSetRemoteDevice](#) ([RTcontext](#) context, [RTremotedevice](#) remote\_dev)
- [RTresult RTAPI rtContextSetStackSize](#) ([RTcontext](#) context, RTsize stack\_size\_bytes)
- [RTresult RTAPI rtContextGetStackSize](#) ([RTcontext](#) context, RTsize \*stack\_size\_bytes)
- [RTresult RTAPI rtContextSetTimeoutCallback](#) ([RTcontext](#) context, [RTtimeoutcallback](#) callback, double min\_ - polling\_seconds)
- [RTresult RTAPI rtContextSetEntryPointCount](#) ([RTcontext](#) context, unsigned int num\_entry\_points)
- [RTresult RTAPI rtContextGetEntryPointCount](#) ([RTcontext](#) context, unsigned int \*num\_entry\_points)
- [RTresult RTAPI rtContextSetRayGenerationProgram](#) ([RTcontext](#) context, unsigned int entry\_point\_index, [RTprogram](#) program)
- [RTresult RTAPI rtContextGetRayGenerationProgram](#) ([RTcontext](#) context, unsigned int entry\_point\_index, [RTprogram](#) \*program)
- [RTresult RTAPI rtContextSetExceptionProgram](#) ([RTcontext](#) context, unsigned int entry\_point\_index, [RTprogram](#) program)
- [RTresult RTAPI rtContextGetExceptionProgram](#) ([RTcontext](#) context, unsigned int entry\_point\_index, [RTprogram](#) \*program)
- [RTresult RTAPI rtContextSetExceptionEnabled](#) ([RTcontext](#) context, [RTexception](#) exception, int enabled)
- [RTresult RTAPI rtContextGetExceptionEnabled](#) ([RTcontext](#) context, [RTexception](#) exception, int \*enabled)
- [RTresult RTAPI rtContextSetRayTypeCount](#) ([RTcontext](#) context, unsigned int num\_ray\_types)
- [RTresult RTAPI rtContextGetRayTypeCount](#) ([RTcontext](#) context, unsigned int \*num\_ray\_types)
- [RTresult RTAPI rtContextSetMissProgram](#) ([RTcontext](#) context, unsigned int ray\_type\_index, [RTprogram](#) program)
- [RTresult RTAPI rtContextGetMissProgram](#) ([RTcontext](#) context, unsigned int ray\_type\_index, [RTprogram](#) \*program)
- [RTresult RTAPI rtContextGetTextureSamplerFromId](#) ([RTcontext](#) context, int sampler\_id, [RTtexturesampler](#) \*sampler)
- [RTresult RTAPI rtContextCompile](#) ([RTcontext](#) context)
- [RTresult RTAPI rtContextLaunch1D](#) ([RTcontext](#) context, unsigned int entry\_point\_index, RTsize image\_ - width)
- [RTresult RTAPI rtContextLaunch2D](#) ([RTcontext](#) context, unsigned int entry\_point\_index, RTsize image\_width, RTsize image\_height)
- [RTresult RTAPI rtContextLaunch3D](#) ([RTcontext](#) context, unsigned int entry\_point\_index, RTsize image\_width, RTsize image\_height, RTsize image\_depth)



- [RTresult](#) RTAPI [rtContextGetRunningState](#) ([RTcontext](#) context, int \*running)
- [RTresult](#) RTAPI [rtContextLaunchProgressive2D](#) ([RTcontext](#) context, unsigned int entry\_index, RTsize width, RTsize height, unsigned int max\_subframes)
- [RTresult](#) RTAPI [rtContextStopProgressive](#) ([RTcontext](#) context)
- [RTresult](#) RTAPI [rtContextSetPrintEnabled](#) ([RTcontext](#) context, int enabled)
- [RTresult](#) RTAPI [rtContextGetPrintEnabled](#) ([RTcontext](#) context, int \*enabled)
- [RTresult](#) RTAPI [rtContextSetPrintBufferSize](#) ([RTcontext](#) context, RTsize buffer\_size\_bytes)
- [RTresult](#) RTAPI [rtContextGetPrintBufferSize](#) ([RTcontext](#) context, RTsize \*buffer\_size\_bytes)
- [RTresult](#) RTAPI [rtContextSetPrintLaunchIndex](#) ([RTcontext](#) context, int x, int y, int z)
- [RTresult](#) RTAPI [rtContextGetPrintLaunchIndex](#) ([RTcontext](#) context, int \*x, int \*y, int \*z)
- [RTresult](#) RTAPI [rtContextDeclareVariable](#) ([RTcontext](#) context, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtContextQueryVariable](#) ([RTcontext](#) context, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtContextRemoveVariable](#) ([RTcontext](#) context, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtContextGetVariableCount](#) ([RTcontext](#) context, unsigned int \*count)
- [RTresult](#) RTAPI [rtContextGetVariable](#) ([RTcontext](#) context, unsigned int index, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtProgramCreateFromPTXString](#) ([RTcontext](#) context, const char \*ptx, const char \*program\_name, [RTprogram](#) \*program)
- [RTresult](#) RTAPI [rtProgramCreateFromPTXFile](#) ([RTcontext](#) context, const char \*filename, const char \*program\_name, [RTprogram](#) \*program)
- [RTresult](#) RTAPI [rtProgramDestroy](#) ([RTprogram](#) program)
- [RTresult](#) RTAPI [rtProgramValidate](#) ([RTprogram](#) program)
- [RTresult](#) RTAPI [rtProgramGetContext](#) ([RTprogram](#) program, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtProgramDeclareVariable](#) ([RTprogram](#) program, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtProgramQueryVariable](#) ([RTprogram](#) program, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtProgramRemoveVariable](#) ([RTprogram](#) program, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtProgramGetVariableCount](#) ([RTprogram](#) program, unsigned int \*count)
- [RTresult](#) RTAPI [rtProgramGetVariable](#) ([RTprogram](#) program, unsigned int index, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtProgramGetId](#) ([RTprogram](#) program, int \*program\_id)
- [RTresult](#) RTAPI [rtContextGetProgramFromId](#) ([RTcontext](#) context, int program\_id, [RTprogram](#) \*program)
- [RTresult](#) RTAPI [rtGroupCreate](#) ([RTcontext](#) context, [RTgroup](#) \*group)
- [RTresult](#) RTAPI [rtGroupDestroy](#) ([RTgroup](#) group)
- [RTresult](#) RTAPI [rtGroupValidate](#) ([RTgroup](#) group)
- [RTresult](#) RTAPI [rtGroupGetContext](#) ([RTgroup](#) group, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtGroupSetAcceleration](#) ([RTgroup](#) group, [RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtGroupGetAcceleration](#) ([RTgroup](#) group, [RTacceleration](#) \*acceleration)
- [RTresult](#) RTAPI [rtGroupSetChildCount](#) ([RTgroup](#) group, unsigned int count)
- [RTresult](#) RTAPI [rtGroupGetChildCount](#) ([RTgroup](#) group, unsigned int \*count)
- [RTresult](#) RTAPI [rtGroupSetChild](#) ([RTgroup](#) group, unsigned int index, [RTobject](#) child)
- [RTresult](#) RTAPI [rtGroupGetChild](#) ([RTgroup](#) group, unsigned int index, [RTobject](#) \*child)
- [RTresult](#) RTAPI [rtGroupGetChildType](#) ([RTgroup](#) group, unsigned int index, [RTobjecttype](#) \*type)
- [RTresult](#) RTAPI [rtSelectorCreate](#) ([RTcontext](#) context, [RTselector](#) \*selector)
- [RTresult](#) RTAPI [rtSelectorDestroy](#) ([RTselector](#) selector)
- [RTresult](#) RTAPI [rtSelectorValidate](#) ([RTselector](#) selector)
- [RTresult](#) RTAPI [rtSelectorGetContext](#) ([RTselector](#) selector, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtSelectorSetVisitProgram](#) ([RTselector](#) selector, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtSelectorGetVisitProgram](#) ([RTselector](#) selector, [RTprogram](#) \*program)
- [RTresult](#) RTAPI [rtSelectorSetChildCount](#) ([RTselector](#) selector, unsigned int count)
- [RTresult](#) RTAPI [rtSelectorGetChildCount](#) ([RTselector](#) selector, unsigned int \*count)
- [RTresult](#) RTAPI [rtSelectorSetChild](#) ([RTselector](#) selector, unsigned int index, [RTobject](#) child)
- [RTresult](#) RTAPI [rtSelectorGetChild](#) ([RTselector](#) selector, unsigned int index, [RTobject](#) \*child)
- [RTresult](#) RTAPI [rtSelectorGetChildType](#) ([RTselector](#) selector, unsigned int index, [RTobjecttype](#) \*type)
- [RTresult](#) RTAPI [rtSelectorDeclareVariable](#) ([RTselector](#) selector, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtSelectorQueryVariable](#) ([RTselector](#) selector, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtSelectorRemoveVariable](#) ([RTselector](#) selector, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtSelectorGetVariableCount](#) ([RTselector](#) selector, unsigned int \*count)

- [RTresult](#) RTAPI [rtSelectorGetVariable](#) ([RTselector](#) selector, unsigned int index, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtTransformCreate](#) ([RTcontext](#) context, [RTtransform](#) \*transform)
- [RTresult](#) RTAPI [rtTransformDestroy](#) ([RTtransform](#) transform)
- [RTresult](#) RTAPI [rtTransformValidate](#) ([RTtransform](#) transform)
- [RTresult](#) RTAPI [rtTransformGetContext](#) ([RTtransform](#) transform, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtTransformSetMatrix](#) ([RTtransform](#) transform, int transpose, const float \*matrix, const float \*inverse\_matrix)
- [RTresult](#) RTAPI [rtTransformGetMatrix](#) ([RTtransform](#) transform, int transpose, float \*matrix, float \*inverse\_matrix)
- [RTresult](#) RTAPI [rtTransformSetChild](#) ([RTtransform](#) transform, [RTobject](#) child)
- [RTresult](#) RTAPI [rtTransformGetChild](#) ([RTtransform](#) transform, [RTobject](#) \*child)
- [RTresult](#) RTAPI [rtTransformGetChildType](#) ([RTtransform](#) transform, [RTobjecttype](#) \*type)
- [RTresult](#) RTAPI [rtGeometryGroupCreate](#) ([RTcontext](#) context, [RTgeometrygroup](#) \*geometrygroup)
- [RTresult](#) RTAPI [rtGeometryGroupDestroy](#) ([RTgeometrygroup](#) geometrygroup)
- [RTresult](#) RTAPI [rtGeometryGroupValidate](#) ([RTgeometrygroup](#) geometrygroup)
- [RTresult](#) RTAPI [rtGeometryGroupGetContext](#) ([RTgeometrygroup](#) geometrygroup, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtGeometryGroupSetAcceleration](#) ([RTgeometrygroup](#) geometrygroup, [RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtGeometryGroupGetAcceleration](#) ([RTgeometrygroup](#) geometrygroup, [RTacceleration](#) \*acceleration)
- [RTresult](#) RTAPI [rtGeometryGroupSetChildCount](#) ([RTgeometrygroup](#) geometrygroup, unsigned int count)
- [RTresult](#) RTAPI [rtGeometryGroupGetChildCount](#) ([RTgeometrygroup](#) geometrygroup, unsigned int \*count)
- [RTresult](#) RTAPI [rtGeometryGroupSetChild](#) ([RTgeometrygroup](#) geometrygroup, unsigned int index, [RTgeometryinstance](#) geometryinstance)
- [RTresult](#) RTAPI [rtGeometryGroupGetChild](#) ([RTgeometrygroup](#) geometrygroup, unsigned int index, [RTgeometryinstance](#) \*geometryinstance)
- [RTresult](#) RTAPI [rtAccelerationCreate](#) ([RTcontext](#) context, [RTacceleration](#) \*acceleration)
- [RTresult](#) RTAPI [rtAccelerationDestroy](#) ([RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtAccelerationValidate](#) ([RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtAccelerationGetContext](#) ([RTacceleration](#) acceleration, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtAccelerationSetBuilder](#) ([RTacceleration](#) acceleration, const char \*builder)
- [RTresult](#) RTAPI [rtAccelerationGetBuilder](#) ([RTacceleration](#) acceleration, const char \*\*return\_string)
- [RTresult](#) RTAPI [rtAccelerationSetTraverser](#) ([RTacceleration](#) acceleration, const char \*traverser)
- [RTresult](#) RTAPI [rtAccelerationGetTraverser](#) ([RTacceleration](#) acceleration, const char \*\*return\_string)
- [RTresult](#) RTAPI [rtAccelerationSetProperty](#) ([RTacceleration](#) acceleration, const char \*name, const char \*value)
- [RTresult](#) RTAPI [rtAccelerationGetProperty](#) ([RTacceleration](#) acceleration, const char \*name, const char \*\*return\_string)
- [RTresult](#) RTAPI [rtAccelerationGetDataSize](#) ([RTacceleration](#) acceleration, [RTsize](#) \*size)
- [RTresult](#) RTAPI [rtAccelerationGetData](#) ([RTacceleration](#) acceleration, void \*data)
- [RTresult](#) RTAPI [rtAccelerationSetData](#) ([RTacceleration](#) acceleration, const void \*data, [RTsize](#) size)
- [RTresult](#) RTAPI [rtAccelerationMarkDirty](#) ([RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtAccelerationIsDirty](#) ([RTacceleration](#) acceleration, int \*dirty)
- [RTresult](#) RTAPI [rtGeometryInstanceCreate](#) ([RTcontext](#) context, [RTgeometryinstance](#) \*geometryinstance)
- [RTresult](#) RTAPI [rtGeometryInstanceDestroy](#) ([RTgeometryinstance](#) geometryinstance)
- [RTresult](#) RTAPI [rtGeometryInstanceValidate](#) ([RTgeometryinstance](#) geometryinstance)
- [RTresult](#) RTAPI [rtGeometryInstanceGetContext](#) ([RTgeometryinstance](#) geometryinstance, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtGeometryInstanceSetGeometry](#) ([RTgeometryinstance](#) geometryinstance, [RTgeometry](#) geometry)
- [RTresult](#) RTAPI [rtGeometryInstanceGetGeometry](#) ([RTgeometryinstance](#) geometryinstance, [RTgeometry](#) \*geometry)
- [RTresult](#) RTAPI [rtGeometryInstanceSetMaterialCount](#) ([RTgeometryinstance](#) geometryinstance, unsigned int count)
- [RTresult](#) RTAPI [rtGeometryInstanceGetMaterialCount](#) ([RTgeometryinstance](#) geometryinstance, unsigned int \*count)

- [RTresult](#) RTAPI [rtGeometryInstanceSetMaterial](#) ([RTgeometryinstance](#) geometryinstance, unsigned int index, [RTmaterial](#) material)
- [RTresult](#) RTAPI [rtGeometryInstanceGetMaterial](#) ([RTgeometryinstance](#) geometryinstance, unsigned int index, [RTmaterial](#) \*material)
- [RTresult](#) RTAPI [rtGeometryInstanceDeclareVariable](#) ([RTgeometryinstance](#) geometryinstance, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtGeometryInstanceQueryVariable](#) ([RTgeometryinstance](#) geometryinstance, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtGeometryInstanceRemoveVariable](#) ([RTgeometryinstance](#) geometryinstance, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtGeometryInstanceGetVariableCount](#) ([RTgeometryinstance](#) geometryinstance, unsigned int \*count)
- [RTresult](#) RTAPI [rtGeometryInstanceGetVariable](#) ([RTgeometryinstance](#) geometryinstance, unsigned int index, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtGeometryCreate](#) ([RTcontext](#) context, [RTgeometry](#) \*geometry)
- [RTresult](#) RTAPI [rtGeometryDestroy](#) ([RTgeometry](#) geometry)
- [RTresult](#) RTAPI [rtGeometryValidate](#) ([RTgeometry](#) geometry)
- [RTresult](#) RTAPI [rtGeometryGetContext](#) ([RTgeometry](#) geometry, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtGeometrySetPrimitiveCount](#) ([RTgeometry](#) geometry, unsigned int num\_primitives)
- [RTresult](#) RTAPI [rtGeometryGetPrimitiveCount](#) ([RTgeometry](#) geometry, unsigned int \*num\_primitives)
- [RTresult](#) RTAPI [rtGeometrySetPrimitiveIndexOffset](#) ([RTgeometry](#) geometry, unsigned int index\_offset)
- [RTresult](#) RTAPI [rtGeometryGetPrimitiveIndexOffset](#) ([RTgeometry](#) geometry, unsigned int \*index\_offset)
- [RTresult](#) RTAPI [rtGeometrySetBoundingBoxProgram](#) ([RTgeometry](#) geometry, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtGeometryGetBoundingBoxProgram](#) ([RTgeometry](#) geometry, [RTprogram](#) \*program)
- [RTresult](#) RTAPI [rtGeometrySetIntersectionProgram](#) ([RTgeometry](#) geometry, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtGeometryGetIntersectionProgram](#) ([RTgeometry](#) geometry, [RTprogram](#) \*program)
- [RTresult](#) RTAPI [rtGeometryMarkDirty](#) ([RTgeometry](#) geometry)
- [RTresult](#) RTAPI [rtGeometryIsDirty](#) ([RTgeometry](#) geometry, int \*dirty)
- [RTresult](#) RTAPI [rtGeometryDeclareVariable](#) ([RTgeometry](#) geometry, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtGeometryQueryVariable](#) ([RTgeometry](#) geometry, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtGeometryRemoveVariable](#) ([RTgeometry](#) geometry, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtGeometryGetVariableCount](#) ([RTgeometry](#) geometry, unsigned int \*count)
- [RTresult](#) RTAPI [rtGeometryGetVariable](#) ([RTgeometry](#) geometry, unsigned int index, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtMaterialCreate](#) ([RTcontext](#) context, [RTmaterial](#) \*material)
- [RTresult](#) RTAPI [rtMaterialDestroy](#) ([RTmaterial](#) material)
- [RTresult](#) RTAPI [rtMaterialValidate](#) ([RTmaterial](#) material)
- [RTresult](#) RTAPI [rtMaterialGetContext](#) ([RTmaterial](#) material, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtMaterialSetClosestHitProgram](#) ([RTmaterial](#) material, unsigned int ray\_type\_index, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtMaterialGetClosestHitProgram](#) ([RTmaterial](#) material, unsigned int ray\_type\_index, [RTprogram](#) \*program)
- [RTresult](#) RTAPI [rtMaterialSetAnyHitProgram](#) ([RTmaterial](#) material, unsigned int ray\_type\_index, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtMaterialGetAnyHitProgram](#) ([RTmaterial](#) material, unsigned int ray\_type\_index, [RTprogram](#) \*program)
- [RTresult](#) RTAPI [rtMaterialDeclareVariable](#) ([RTmaterial](#) material, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtMaterialQueryVariable](#) ([RTmaterial](#) material, const char \*name, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtMaterialRemoveVariable](#) ([RTmaterial](#) material, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtMaterialGetVariableCount](#) ([RTmaterial](#) material, unsigned int \*count)
- [RTresult](#) RTAPI [rtMaterialGetVariable](#) ([RTmaterial](#) material, unsigned int index, [RTvariable](#) \*v)
- [RTresult](#) RTAPI [rtTextureSamplerCreate](#) ([RTcontext](#) context, [RTtexturesampler](#) \*texturesampler)
- [RTresult](#) RTAPI [rtTextureSamplerDestroy](#) ([RTtexturesampler](#) texturesampler)
- [RTresult](#) RTAPI [rtTextureSamplerValidate](#) ([RTtexturesampler](#) texturesampler)
- [RTresult](#) RTAPI [rtTextureSamplerGetContext](#) ([RTtexturesampler](#) texturesampler, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtTextureSamplerSetMipLevelCount](#) ([RTtexturesampler](#) texturesampler, unsigned int num\_mip\_levels)

- [RTresult](#) RTAPI [rtTextureSamplerGetMipLevelCount](#) ([RTtexturesampler](#) texturesampler, unsigned int \*num\_mip\_levels)
- [RTresult](#) RTAPI [rtTextureSamplerSetArraySize](#) ([RTtexturesampler](#) texturesampler, unsigned int num\_textures\_in\_array)
- [RTresult](#) RTAPI [rtTextureSamplerGetArraySize](#) ([RTtexturesampler](#) texturesampler, unsigned int \*num\_textures\_in\_array)
- [RTresult](#) RTAPI [rtTextureSamplerSetWrapMode](#) ([RTtexturesampler](#) texturesampler, unsigned int dimension, [RTwrapmode](#) wrapmode)
- [RTresult](#) RTAPI [rtTextureSamplerGetWrapMode](#) ([RTtexturesampler](#) texturesampler, unsigned int dimension, [RTwrapmode](#) \*wrapmode)
- [RTresult](#) RTAPI [rtTextureSamplerSetFilteringModes](#) ([RTtexturesampler](#) texturesampler, [RTfiltermode](#) minification, [RTfiltermode](#) magnification, [RTfiltermode](#) mipmapping)
- [RTresult](#) RTAPI [rtTextureSamplerGetFilteringModes](#) ([RTtexturesampler](#) texturesampler, [RTfiltermode](#) \*minification, [RTfiltermode](#) \*magnification, [RTfiltermode](#) \*mipmapping)
- [RTresult](#) RTAPI [rtTextureSamplerSetMaxAnisotropy](#) ([RTtexturesampler](#) texturesampler, float value)
- [RTresult](#) RTAPI [rtTextureSamplerGetMaxAnisotropy](#) ([RTtexturesampler](#) texturesampler, float \*value)
- [RTresult](#) RTAPI [rtTextureSamplerSetMipLevelClamp](#) ([RTtexturesampler](#) texturesampler, float minLevel, float maxLevel)
- [RTresult](#) RTAPI [rtTextureSamplerGetMipLevelClamp](#) ([RTtexturesampler](#) texturesampler, float \*minLevel, float \*maxLevel)
- [RTresult](#) RTAPI [rtTextureSamplerSetMipLevelBias](#) ([RTtexturesampler](#) texturesampler, float value)
- [RTresult](#) RTAPI [rtTextureSamplerGetMipLevelBias](#) ([RTtexturesampler](#) texturesampler, float \*value)
- [RTresult](#) RTAPI [rtTextureSamplerSetReadMode](#) ([RTtexturesampler](#) texturesampler, [RTtexturereadmode](#) readmode)
- [RTresult](#) RTAPI [rtTextureSamplerGetReadMode](#) ([RTtexturesampler](#) texturesampler, [RTtexturereadmode](#) \*readmode)
- [RTresult](#) RTAPI [rtTextureSamplerSetIndexingMode](#) ([RTtexturesampler](#) texturesampler, [RTtextureindexmode](#) indexmode)
- [RTresult](#) RTAPI [rtTextureSamplerGetIndexingMode](#) ([RTtexturesampler](#) texturesampler, [RTtextureindexmode](#) \*indexmode)
- [RTresult](#) RTAPI [rtTextureSamplerSetBuffer](#) ([RTtexturesampler](#) texturesampler, unsigned int deprecated0, unsigned int deprecated1, [RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtTextureSamplerGetBuffer](#) ([RTtexturesampler](#) texturesampler, unsigned int deprecated0, unsigned int deprecated1, [RTbuffer](#) \*buffer)
- [RTresult](#) RTAPI [rtTextureSamplerGetId](#) ([RTtexturesampler](#) texturesampler, int \*texture\_id)
- [RTresult](#) RTAPI [rtBufferCreate](#) ([RTcontext](#) context, unsigned int bufferdesc, [RTbuffer](#) \*buffer)
- [RTresult](#) RTAPI [rtBufferDestroy](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferValidate](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferGetContext](#) ([RTbuffer](#) buffer, [RTcontext](#) \*context)
- [RTresult](#) RTAPI [rtBufferSetFormat](#) ([RTbuffer](#) buffer, [RTformat](#) format)
- [RTresult](#) RTAPI [rtBufferGetFormat](#) ([RTbuffer](#) buffer, [RTformat](#) \*format)
- [RTresult](#) RTAPI [rtBufferSetElementSize](#) ([RTbuffer](#) buffer, [RTsize](#) size\_of\_element)
- [RTresult](#) RTAPI [rtBufferGetElementSize](#) ([RTbuffer](#) buffer, [RTsize](#) \*size\_of\_element)
- [RTresult](#) RTAPI [rtBufferSetSize1D](#) ([RTbuffer](#) buffer, [RTsize](#) width)
- [RTresult](#) RTAPI [rtBufferGetSize1D](#) ([RTbuffer](#) buffer, [RTsize](#) \*width)
- [RTresult](#) RTAPI [rtBufferSetSize2D](#) ([RTbuffer](#) buffer, [RTsize](#) width, [RTsize](#) height)
- [RTresult](#) RTAPI [rtBufferGetSize2D](#) ([RTbuffer](#) buffer, [RTsize](#) \*width, [RTsize](#) \*height)
- [RTresult](#) RTAPI [rtBufferSetSize3D](#) ([RTbuffer](#) buffer, [RTsize](#) width, [RTsize](#) height, [RTsize](#) depth)
- [RTresult](#) RTAPI [rtBufferSetMipLevelCount](#) ([RTbuffer](#) buffer, unsigned int levels)
- [RTresult](#) RTAPI [rtBufferGetSize3D](#) ([RTbuffer](#) buffer, [RTsize](#) \*width, [RTsize](#) \*height, [RTsize](#) \*depth)
- [RTresult](#) RTAPI [rtBufferGetMipLevelSize1D](#) ([RTbuffer](#) buffer, unsigned int level, [RTsize](#) \*width)
- [RTresult](#) RTAPI [rtBufferGetMipLevelSize2D](#) ([RTbuffer](#) buffer, unsigned int level, [RTsize](#) \*width, [RTsize](#) \*height)
- [RTresult](#) RTAPI [rtBufferGetMipLevelSize3D](#) ([RTbuffer](#) buffer, unsigned int level, [RTsize](#) \*width, [RTsize](#) \*height, [RTsize](#) \*depth)
- [RTresult](#) RTAPI [rtBufferSetSizev](#) ([RTbuffer](#) buffer, unsigned int dimensionality, const [RTsize](#) \*dims)

- [RTresult](#) RTAPI [rtBufferGetSizev](#) ([RTbuffer](#) buffer, unsigned int dimensionality, [RTsize](#) \*dims)
- [RTresult](#) RTAPI [rtBufferGetDimensionality](#) ([RTbuffer](#) buffer, unsigned int \*dimensionality)
- [RTresult](#) RTAPI [rtBufferGetMipLevelCount](#) ([RTbuffer](#) buffer, unsigned int \*level)
- [RTresult](#) RTAPI [rtBufferMap](#) ([RTbuffer](#) buffer, void \*\*user\_pointer)
- [RTresult](#) RTAPI [rtBufferUnmap](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferMapEx](#) ([RTbuffer](#) buffer, unsigned int map\_flags, unsigned int level, void \*user\_ - owned, void \*\*optix\_owned)
- [RTresult](#) RTAPI [rtBufferUnmapEx](#) ([RTbuffer](#) buffer, unsigned int level)
- [RTresult](#) RTAPI [rtBufferGetId](#) ([RTbuffer](#) buffer, int \*buffer\_id)
- [RTresult](#) RTAPI [rtContextGetBufferFromId](#) ([RTcontext](#) context, int buffer\_id, [RTbuffer](#) \*buffer)
- [RTresult](#) RTAPI [rtBufferGetProgressiveUpdateReady](#) ([RTbuffer](#) buffer, int \*ready, unsigned int \*subframe\_ - count, unsigned int \*max\_subframes)
- [RTresult](#) RTAPI [rtBufferBindProgressiveStream](#) ([RTbuffer](#) stream, [RTbuffer](#) source)
- [RTresult](#) RTAPI [rtBufferSetAttribute](#) ([RTbuffer](#) buffer, [RTbufferattribute](#) attrib, [RTsize](#) size, void \*p)
- [RTresult](#) RTAPI [rtBufferGetAttribute](#) ([RTbuffer](#) buffer, [RTbufferattribute](#) attrib, [RTsize](#) size, void \*p)
- [RTresult](#) RTAPI [rtRemoteDeviceCreate](#) (const char \*url, const char \*username, const char \*password, [RTremotedevice](#) \*remote\_dev)
- [RTresult](#) RTAPI [rtRemoteDeviceDestroy](#) ([RTremotedevice](#) remote\_dev)
- [RTresult](#) RTAPI [rtRemoteDeviceGetAttribute](#) ([RTremotedevice](#) remote\_dev, [RTremotedeviceattribute](#) attrib, [RTsize](#) size, void \*p)
- [RTresult](#) RTAPI [rtRemoteDeviceReserve](#) ([RTremotedevice](#) remote\_dev, unsigned int num\_nodes, unsigned int configuration)
- [RTresult](#) RTAPI [rtRemoteDeviceRelease](#) ([RTremotedevice](#) remote\_dev)
  
- [RTresult](#) RTAPI [rtVariableSet1f](#) ([RTvariable](#) v, float f1)
- [RTresult](#) RTAPI [rtVariableSet2f](#) ([RTvariable](#) v, float f1, float f2)
- [RTresult](#) RTAPI [rtVariableSet3f](#) ([RTvariable](#) v, float f1, float f2, float f3)
- [RTresult](#) RTAPI [rtVariableSet4f](#) ([RTvariable](#) v, float f1, float f2, float f3, float f4)
- [RTresult](#) RTAPI [rtVariableSet1fv](#) ([RTvariable](#) v, const float \*f)
- [RTresult](#) RTAPI [rtVariableSet2fv](#) ([RTvariable](#) v, const float \*f)
- [RTresult](#) RTAPI [rtVariableSet3fv](#) ([RTvariable](#) v, const float \*f)
- [RTresult](#) RTAPI [rtVariableSet4fv](#) ([RTvariable](#) v, const float \*f)
- [RTresult](#) RTAPI [rtVariableSet1i](#) ([RTvariable](#) v, int i1)
- [RTresult](#) RTAPI [rtVariableSet2i](#) ([RTvariable](#) v, int i1, int i2)
- [RTresult](#) RTAPI [rtVariableSet3i](#) ([RTvariable](#) v, int i1, int i2, int i3)
- [RTresult](#) RTAPI [rtVariableSet4i](#) ([RTvariable](#) v, int i1, int i2, int i3, int i4)
- [RTresult](#) RTAPI [rtVariableSet1iv](#) ([RTvariable](#) v, const int \*i)
- [RTresult](#) RTAPI [rtVariableSet2iv](#) ([RTvariable](#) v, const int \*i)
- [RTresult](#) RTAPI [rtVariableSet3iv](#) ([RTvariable](#) v, const int \*i)
- [RTresult](#) RTAPI [rtVariableSet4iv](#) ([RTvariable](#) v, const int \*i)
- [RTresult](#) RTAPI [rtVariableSet1ui](#) ([RTvariable](#) v, unsigned int u1)
- [RTresult](#) RTAPI [rtVariableSet2ui](#) ([RTvariable](#) v, unsigned int u1, unsigned int u2)
- [RTresult](#) RTAPI [rtVariableSet3ui](#) ([RTvariable](#) v, unsigned int u1, unsigned int u2, unsigned int u3)
- [RTresult](#) RTAPI [rtVariableSet4ui](#) ([RTvariable](#) v, unsigned int u1, unsigned int u2, unsigned int u3, unsigned int u4)
- [RTresult](#) RTAPI [rtVariableSet1uiv](#) ([RTvariable](#) v, const unsigned int \*u)
- [RTresult](#) RTAPI [rtVariableSet2uiv](#) ([RTvariable](#) v, const unsigned int \*u)
- [RTresult](#) RTAPI [rtVariableSet3uiv](#) ([RTvariable](#) v, const unsigned int \*u)
- [RTresult](#) RTAPI [rtVariableSet4uiv](#) ([RTvariable](#) v, const unsigned int \*u)
- [RTresult](#) RTAPI [rtVariableSetMatrix2x2fv](#) ([RTvariable](#) v, int transpose, const float \*m)
- [RTresult](#) RTAPI [rtVariableSetMatrix2x3fv](#) ([RTvariable](#) v, int transpose, const float \*m)
- [RTresult](#) RTAPI [rtVariableSetMatrix2x4fv](#) ([RTvariable](#) v, int transpose, const float \*m)
- [RTresult](#) RTAPI [rtVariableSetMatrix3x2fv](#) ([RTvariable](#) v, int transpose, const float \*m)
- [RTresult](#) RTAPI [rtVariableSetMatrix3x3fv](#) ([RTvariable](#) v, int transpose, const float \*m)
- [RTresult](#) RTAPI [rtVariableSetMatrix3x4fv](#) ([RTvariable](#) v, int transpose, const float \*m)



- **RTresult** RTAPI **rtVariableSetMatrix4x2fv** (**RTvariable** v, int transpose, const float \*m)
- **RTresult** RTAPI **rtVariableSetMatrix4x3fv** (**RTvariable** v, int transpose, const float \*m)
- **RTresult** RTAPI **rtVariableSetMatrix4x4fv** (**RTvariable** v, int transpose, const float \*m)
  
- **RTresult** RTAPI **rtVariableGet1f** (**RTvariable** v, float \*f1)
- **RTresult** RTAPI **rtVariableGet2f** (**RTvariable** v, float \*f1, float \*f2)
- **RTresult** RTAPI **rtVariableGet3f** (**RTvariable** v, float \*f1, float \*f2, float \*f3)
- **RTresult** RTAPI **rtVariableGet4f** (**RTvariable** v, float \*f1, float \*f2, float \*f3, float \*f4)
- **RTresult** RTAPI **rtVariableGet1fv** (**RTvariable** v, float \*f)
- **RTresult** RTAPI **rtVariableGet2fv** (**RTvariable** v, float \*f)
- **RTresult** RTAPI **rtVariableGet3fv** (**RTvariable** v, float \*f)
- **RTresult** RTAPI **rtVariableGet4fv** (**RTvariable** v, float \*f)
- **RTresult** RTAPI **rtVariableGet1i** (**RTvariable** v, int \*i1)
- **RTresult** RTAPI **rtVariableGet2i** (**RTvariable** v, int \*i1, int \*i2)
- **RTresult** RTAPI **rtVariableGet3i** (**RTvariable** v, int \*i1, int \*i2, int \*i3)
- **RTresult** RTAPI **rtVariableGet4i** (**RTvariable** v, int \*i1, int \*i2, int \*i3, int \*i4)
- **RTresult** RTAPI **rtVariableGet1iv** (**RTvariable** v, int \*i)
- **RTresult** RTAPI **rtVariableGet2iv** (**RTvariable** v, int \*i)
- **RTresult** RTAPI **rtVariableGet3iv** (**RTvariable** v, int \*i)
- **RTresult** RTAPI **rtVariableGet4iv** (**RTvariable** v, int \*i)
- **RTresult** RTAPI **rtVariableGet1ui** (**RTvariable** v, unsigned int \*u1)
- **RTresult** RTAPI **rtVariableGet2ui** (**RTvariable** v, unsigned int \*u1, unsigned int \*u2)
- **RTresult** RTAPI **rtVariableGet3ui** (**RTvariable** v, unsigned int \*u1, unsigned int \*u2, unsigned int \*u3)
- **RTresult** RTAPI **rtVariableGet4ui** (**RTvariable** v, unsigned int \*u1, unsigned int \*u2, unsigned int \*u3, unsigned int \*u4)
- **RTresult** RTAPI **rtVariableGet1uiv** (**RTvariable** v, unsigned int \*u)
- **RTresult** RTAPI **rtVariableGet2uiv** (**RTvariable** v, unsigned int \*u)
- **RTresult** RTAPI **rtVariableGet3uiv** (**RTvariable** v, unsigned int \*u)
- **RTresult** RTAPI **rtVariableGet4uiv** (**RTvariable** v, unsigned int \*u)
- **RTresult** RTAPI **rtVariableGetMatrix2x2fv** (**RTvariable** v, int transpose, float \*m)
- **RTresult** RTAPI **rtVariableGetMatrix2x3fv** (**RTvariable** v, int transpose, float \*m)
- **RTresult** RTAPI **rtVariableGetMatrix2x4fv** (**RTvariable** v, int transpose, float \*m)
- **RTresult** RTAPI **rtVariableGetMatrix3x2fv** (**RTvariable** v, int transpose, float \*m)
- **RTresult** RTAPI **rtVariableGetMatrix3x3fv** (**RTvariable** v, int transpose, float \*m)
- **RTresult** RTAPI **rtVariableGetMatrix3x4fv** (**RTvariable** v, int transpose, float \*m)
- **RTresult** RTAPI **rtVariableGetMatrix4x2fv** (**RTvariable** v, int transpose, float \*m)
- **RTresult** RTAPI **rtVariableGetMatrix4x3fv** (**RTvariable** v, int transpose, float \*m)
- **RTresult** RTAPI **rtVariableGetMatrix4x4fv** (**RTvariable** v, int transpose, float \*m)

### 3.11.2 Typedef Documentation

#### 3.11.2.1 typedef struct RTacceleration\_api\* RTacceleration

Opaque type to handle Acceleration Structures - Note that the \*\_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.2 typedef struct RTbuffer\_api\* RTbuffer

Opaque type to handle Buffers - Note that the \*\_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.3 typedef struct RTcontext\_api\* RTcontext

Opaque type to handle Contexts - Note that the \*\_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.4 `typedef struct RTgeometry_api* RTgeometry`

Opaque type to handle Geometry - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.5 `typedef struct RTgeometrygroup_api* RTgeometrygroup`

Opaque type to handle Geometry Group - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.6 `typedef struct RTgeometryinstance_api* RTgeometryinstance`

Opaque type to handle Geometry Instance - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.7 `typedef struct RTgroup_api* RTgroup`

Opaque type to handle Group - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.8 `typedef struct RTmaterial_api* RTmaterial`

Opaque type to handle Material - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.9 `typedef void* RTOBJECT`

Opaque type to handle Object - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.10 `typedef struct RTprogram_api* RTprogram`

Opaque type to handle Program - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.11 `typedef struct RTremotedevice_api* RTremotedevice`

Opaque type to handle RemoteDevice - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.12 `typedef struct RTselector_api* RTselector`

Opaque type to handle Selector - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.13 `typedef struct RTtexturesampler_api* RTtexturesampler`

Opaque type to handle Texture Sampler - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

#### 3.11.2.14 `typedef int(* RTtimeoutcallback)(void)`

Callback signature for use with `rtContextSetTimeoutCallback`. Return 1 to ask for abort, 0 to continue.

#### 3.11.2.15 `typedef struct RTtransform_api* RTtransform`

Opaque type to handle Transform - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.11.2.16 `typedef struct RTvariable_api* RTvariable`

Opaque type to handle Variable - Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

## 3.11.3 Function Documentation

3.11.3.1 `RTresult RTAPI rtAccelerationCreate ( RTcontext context, RTacceleration * acceleration )`

Creates a new acceleration structure.

**Description**

`rtAccelerationCreate` creates a new ray tracing acceleration structure within a context. An acceleration structure is used by attaching it to a group or geometry group by calling `rtGroupSetAcceleration` or `rtGeometryGroupSetAcceleration`. Note that an acceleration structure can be shared by attaching it to multiple groups or geometry groups if the underlying geometric structures are the same, see `rtGroupSetAcceleration` and `rtGeometryGroupSetAcceleration` for more details. A newly created acceleration structure is initially in dirty state. Sets `*acceleration` to the handle of a newly created acceleration structure within `context`. Returns `RT_ERROR_INVALID_VALUE` if `acceleration` is `NULL`.

**Parameters**

in	<code>context</code>	Specifies a context within which to create a new acceleration structure
out	<code>acceleration</code>	Returns the newly created acceleration structure

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

**History**

`rtAccelerationCreate` was introduced in OptiX 1.0.

**See also** `rtAccelerationDestroy`, `rtContextCreate`, `rtAccelerationMarkDirty`, `rtAccelerationIsDirty`, `rtGroupSetAcceleration`, `rtGeometryGroupSetAcceleration`

3.11.3.2 `RTresult RTAPI rtAccelerationDestroy ( RTacceleration acceleration )`

Destroys an acceleration structure object.

**Description**

`rtAccelerationDestroy` removes `acceleration` from its context and deletes it. `acceleration` should be a value returned by `rtAccelerationCreate`. After the call, `acceleration` is no longer a valid handle.

**Parameters**

in	<code>acceleration</code>	Handle of the acceleration structure to destroy
----	---------------------------	---

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

**History**



[rtAccelerationDestroy](#) was introduced in OptiX 1.0.

See also [rtAccelerationCreate](#)

#### 3.11.3.3 RTresult RTAPI `rtAccelerationGetBuilder` ( `RTacceleration acceleration`, `const char ** return_string` )

Query the current builder from an acceleration structure.

##### Description

[rtAccelerationGetBuilder](#) returns the name of the builder currently used in the acceleration structure *acceleration*. If no builder has been set for *acceleration*, an empty string is returned. *return\_string* will be set to point to the returned string. The memory *return\_string* points to will be valid until the next API call that returns a string.

##### Parameters

in	<i>acceleration</i>	The acceleration structure handle
out	<i>return_string</i>	Return string buffer

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

##### History

[rtAccelerationGetBuilder](#) was introduced in OptiX 1.0.

See also [rtAccelerationSetBuilder](#)

#### 3.11.3.4 RTresult RTAPI `rtAccelerationGetContext` ( `RTacceleration acceleration`, `RTcontext * context` )

Returns the context associated with an acceleration structure.

##### Description

[rtAccelerationGetContext](#) queries an acceleration structure for its associated context. The context handle is returned in *\*context*.

##### Parameters

in	<i>acceleration</i>	The acceleration structure handle
out	<i>context</i>	Returns the context associated with the acceleration structure

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

##### History

[rtAccelerationGetContext](#) was introduced in OptiX 1.0.

See also [rtAccelerationCreate](#)

#### 3.11.3.5 RTresult RTAPI `rtAccelerationGetData` ( `RTacceleration acceleration`, `void * data` )

Retrieves acceleration structure data.

##### Description

[rtAccelerationGetData](#) retrieves the full state of the *acceleration* object, and copies it to the memory region pointed to by *data*. Sufficient memory must be available to hold the entire state. Use [rtAccelerationGetDataSize](#) to query the required memory size.

The returned *data* from this call is valid input data for `rtAccelerationSetData`.

If *acceleration* is marked dirty, this call is invalid and will return `RT_ERROR_INVALID_VALUE`.

## Parameters

in	<i>acceleration</i>	The acceleration structure handle
out	<i>data</i>	Pointer to a memory region to be filled with the state of <i>acceleration</i>

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtAccelerationGetData](#) was introduced in OptiX 1.0.

See also [rtAccelerationSetData](#), [rtAccelerationGetDataSize](#)

### 3.11.3.6 RTresult RTAPI rtAccelerationGetDataSize ( RTacceleration *acceleration*, RTsize \* *size* )

Returns the size of the data to be retrieved from an acceleration structure.

## Description

[rtAccelerationGetDataSize](#) queries the size of the data that will be returned on a subsequent call to [rtAccelerationGetData](#). The size in bytes will be written to *\*size*. The returned value is guaranteed to be valid only if no other function using the handle *acceleration* is made before [rtAccelerationGetData](#).

If *acceleration* is marked dirty, this call is invalid and will return [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

in	<i>acceleration</i>	The acceleration structure handle
out	<i>size</i>	The returned size of the data in bytes

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtAccelerationGetDataSize](#) was introduced in OptiX 1.0.

See also [rtAccelerationGetData](#), [rtAccelerationSetData](#)

### 3.11.3.7 RTresult RTAPI rtAccelerationGetProperty ( RTacceleration *acceleration*, const char \* *name*, const char \*\* *return\_string* )

Queries an acceleration structure property.

## Description

[rtAccelerationGetProperty](#) returns the value of the acceleration structure property *name*. See [rtAccelerationSetProperty](#) for a list of supported properties. If the property name is not found, an empty string is returned. *return\_string* will be set to point to the returned string. The memory *return\_string* points to will be valid until the next API call that returns a string.

## Parameters

in	<i>acceleration</i>	The acceleration structure handle
in	<i>name</i>	The name of the property to be queried
out	<i>return_string</i>	Return string buffer

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtAccelerationGetProperty](#) was introduced in OptiX 1.0.

**See also** [rtAccelerationSetProperty](#), [rtAccelerationSetBuilder](#), [rtAccelerationSetTraverser](#)

### 3.11.3.8 RTresult RTAPI rtAccelerationGetTraverser ( RTacceleration *acceleration*, const char \*\* *return\_string* )

Query the current traverser from an acceleration structure.

**Description**

[rtAccelerationGetTraverser](#) returns the name of the traverser currently used in the acceleration structure *acceleration*. If no traverser has been set for *acceleration*, an empty string is returned. *return\_string* will be set to point to the returned string. The memory *return\_string* points to will be valid until the next API call that returns a string.

**Parameters**

in	<i>acceleration</i>	The acceleration structure handle
out	<i>return_string</i>	Return string buffer

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtAccelerationGetTraverser](#) was introduced in OptiX 1.0.

**See also** [rtAccelerationSetTraverser](#)

### 3.11.3.9 RTresult RTAPI rtAccelerationIsDirty ( RTacceleration *acceleration*, int \* *dirty* )

Returns the dirty flag of an acceleration structure.

**Description**

[rtAccelerationIsDirty](#) returns whether the acceleration structure is currently marked dirty. If the flag is set, a nonzero value will be returned in *\*dirty*. Otherwise, zero is returned.

Any acceleration structure which is marked dirty will be rebuilt on a call to one of the [rtContextLaunch](#) functions, and its dirty flag will be reset. The dirty flag will also be reset on a successful call to [rtAccelerationSetData](#).

An acceleration structure which is not marked dirty will never be rebuilt, even if associated groups, geometry, properties, or any other values have changed.

Initially after creation, acceleration structures are marked dirty.

**Parameters**

in	<i>acceleration</i>	The acceleration structure handle
out	<i>dirty</i>	Returned dirty flag

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtAccelerationIsDirty](#) was introduced in OptiX 1.0.

**See also** [rtAccelerationMarkDirty](#), [rtAccelerationSetData](#), [rtContextLaunch](#)

### 3.11.3.10 RTresult RTAPI rtAccelerationMarkDirty ( RTacceleration *acceleration* )

Marks an acceleration structure as dirty.

**Description**

[rtAccelerationMarkDirty](#) sets the dirty flag for *acceleration*.

Any acceleration structure which is marked dirty will be rebuilt on a call to one of the [rtContextLaunch](#) functions, and its dirty flag will be reset. The dirty flag will also be reset on a successful call to [rtAccelerationSetData](#).

An acceleration structure which is not marked dirty will never be rebuilt, even if associated groups, geometry, properties, or any other values have changed.

Initially after creation, acceleration structures are marked dirty.

**Parameters**

in	<i>acceleration</i>	The acceleration structure handle
----	---------------------	-----------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtAccelerationMarkDirty](#) was introduced in OptiX 1.0.

**See also** [rtAccelerationIsDirty](#), [rtAccelerationSetData](#), [rtContextLaunch](#)

### 3.11.3.11 RTresult RTAPI rtAccelerationSetBuilder ( RTacceleration *acceleration*, const char \* *builder* )

Specifies the builder to be used for an acceleration structure.

**Description**

[rtAccelerationSetBuilder](#) specifies the method used to construct the ray tracing acceleration structure represented by *acceleration*. A builder must be set for the acceleration structure to pass validation. The current builder can be changed at any time, including after a call to [rtContextLaunch](#). In this case, data previously computed for the acceleration structure is invalidated and the acceleration will be marked dirty.

An acceleration structure is only valid with a correct pair of builder and traverser. The traverser type is specified using [rtAccelerationSetTraverser](#). For a list of valid combinations of builders and traversers, see below. For a description of the individual traversers, see [rtAccelerationSetTraverser](#).

*builder* can take one of the following values:

- "NoAccel": Specifies that no acceleration structure is explicitly built. Traversal linearly loops through the list of primitives to intersect. This can be useful e.g. for higher level groups with only few children, where managing a more complex structure introduces unnecessary overhead. Valid traverser types: "NoAccel".
- "Bvh": A standard bounding volume hierarchy, useful for most types of graph levels and geometry. Medium build speed, good ray tracing performance. Valid traverser types: "Bvh", "BvhCompact".
- "Sbvh": A high quality BVH variant for maximum ray tracing performance. Slower build speed and slightly higher memory footprint than "Bvh". Valid traverser types: "Bvh", "BvhCompact".
- "MedianBvh": A medium quality bounding volume hierarchy with quick build performance. Useful for dynamic and semi-dynamic content. Valid traverser types: "Bvh", "BvhCompact".
- "Lbvh": A simple bounding volume hierarchy with very fast build performance. Useful for dynamic content. Valid traverser types: "Bvh", "BvhCompact".
- "Trbvh": High quality similar to Sbvh but with fast build performance similar to Lbvh. Valid traverser types: "-Bvh". The Trbvh builder uses about 2.5 times the size of the final BVH for scratch space. A CPU-based Trbvh builder that does not have the memory constraints is available. OptiX Commercial includes an optional automatic fallback to the CPU version when out of GPU memory. See details in section 3.5 of the Programming Guide.
- "TriangleKdTree": A high quality kd-tree builder, for triangle geometry only. This may provide better ray tracing performance than the BVH builders for some scenarios. Valid traverser types: "KdTree".

#### Parameters

in	<i>acceleration</i>	The acceleration structure handle
in	<i>builder</i>	String value specifying the builder type

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtAccelerationSetBuilder](#) was introduced in OptiX 1.0.

**See also** [rtAccelerationGetBuilder](#), [rtAccelerationSetTraverser](#), [rtAccelerationSetProperty](#)

#### 3.11.3.12 RTresult RTAPI rtAccelerationSetData ( RTacceleration *acceleration*, const void \* *data*, RTsize *size* )

Sets the state of an acceleration structure.

#### Description

[rtAccelerationSetData](#) sets the full state of the *acceleration* object, including builder and traverser type as well as properties, as defined by *data*. The memory pointed to by *data* must be unaltered values previously retrieved from a (potentially different) acceleration structure handle. This mechanism is useful for implementing caching mechanisms, especially when using high quality structures which are expensive to build.

Note that no check is performed on whether the contents of *data* match the actual underlying geometry on which the acceleration structure is used. If the children of associated groups or geometry groups differ in number of children, layout of bounding boxes, or geometry, then behavior after this call is undefined.

This call returns [RT\\_ERROR\\_VERSION\\_MISMATCH](#) if the specified data was retrieved from a different, incompatible version of OptiX. In this case, the state of *acceleration* is not changed.

If the call is successful, the dirty flag of *acceleration* will be cleared.

## Parameters

in	<i>acceleration</i>	The acceleration structure handle
in	<i>data</i>	Pointer to data containing the serialized state
in	<i>size</i>	The size in bytes of the buffer pointed to by <i>data</i>

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_VERSION\\_MISMATCH](#)

## History

[rtAccelerationSetData](#) was introduced in OptiX 1.0.

See also [rtAccelerationGetData](#), [rtAccelerationGetDataSize](#)

3.11.3.13 **RTresult** RTAPI `rtAccelerationSetProperty` ( **RTacceleration** *acceleration*, `const char *`*name*, `const char *`*value* )

Sets an acceleration structure property.

## Description

[rtAccelerationSetProperty](#) sets a named property value for an acceleration structure. Properties can be used to fine tune the way an acceleration structure is built, in order to achieve faster build times or better ray tracing performance. Properties are evaluated and applied by the acceleration structure during build time, and different builders recognize different properties. Setting a property will never fail as long as *acceleration* is a valid handle. Properties that are not recognized by an acceleration structure will be ignored.

The following is a list of the properties used by the individual builders:

- "NoAccel": No properties are available for this builder.
- "Bvh": **refit** is an integer value specifying whether the BVH should be refitted or rebuilt from scratch when a valid BVH over similar geometry is already existent. The value indicates how many frames are to pass before forcing a rebuild, the exception being a value of 1, which will always refit (never rebuild if possible). A value of 0 will never refit (always rebuild). Regardless of the refit value, if the number of primitives changes from the last frame, a rebuild is forced. Refitting is much faster than a full rebuild, and usually yields good ray tracing performance if deformations to the underlying geometry are not too large. The default is 0. **refit** is only supported on SM\_20 (Fermi) class GPUs and later. Older devices will simply ignore the refit property, effectively rebuilding any time the structure is marked dirty. **refine** can be used in combination with refit, and will apply tree rotations to the existing BVH to attempt to improve the quality for faster traversal. Like refit, tree rotations are much faster than a full rebuild. The value indicates how many rotation passes over the tree to perform per frame. With **refine** on, the quality of the tree degrades much less rapidly than with just refit, and can increase the number of frames between rebuilds before traversal performance suffers. In some cases, it can eliminate the need for rebuilds entirely. The default is 0. **refine** is only supported on SM\_20 (Fermi) class GPUs and later.
- "Sbvh": The SBVH can be used for any type of geometry, but especially efficient structures can be built for triangles. For this case, the following properties are used in order to provide the necessary geometry information to the acceleration object: **vertex\_buffer\_name** specifies the name of the vertex buffer variable for underlying geometry, containing float3 vertices, and defaults to "vertex\_buffer". **vertex\_buffer\_stride** is used to define the offset between two vertices in the buffer, given in bytes. The default stride is zero, which assumes that the vertices are tightly packed. **index\_buffer\_name** specifies the name of the index buffer variable for underlying geometry, if any, and defaults to "index\_buffer". The entries in this buffer are indices of type int, where each index refers to one entry in the vertex buffer. A sequence of three indices represent one triangle. **index\_buffer\_stride** can be used analog to **vertex\_buffer\_stride** to describe interleaved arrays.
- "MedianBvh": **refit** (see **refit** flag for "Bvh" above). **refine**, (see **refine** flag for "Bvh" above).

- "Lbvh": **refit** (see **refit** flag for "Bvh" above). **refine**, (see **refine** flag for "Bvh" above), with one important difference: for "Lbvh", **refine** can be used alone, and does not require **refit**. If used without **refit**, tree rotations will be applied after the Lbvh build. The default is 0.
- "Trbvh": Similar in quality to Sbvh but builds much faster. Builds on the GPU and is subject to GPU memory constraints, including requiring scratch space about 2.5 times as large as the final data structure. See section 3.5 of the programming guide for details. See Sbvh for a description of the relevant properties (**vertex\_buffer\_name**, **index\_buffer\_name**, **vertex\_buffer\_stride**, and **index\_buffer\_stride**).
- "TriangleKdTree": Since the kd-tree can build its acceleration structure over triangles only, the geometry data and its format must be made available to the acceleration object. See Sbvh for a description of the relevant properties (**vertex\_buffer\_name**, **index\_buffer\_name**, **vertex\_buffer\_stride**, and **index\_buffer\_stride**).

#### Parameters

in	<i>acceleration</i>	The acceleration structure handle
in	<i>name</i>	String value specifying the name of the property
in	<i>value</i>	String value specifying the value of the property

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtAccelerationSetProperty](#) was introduced in OptiX 1.0.

**See also** [rtAccelerationGetProperty](#), [rtAccelerationSetBuilder](#), [rtAccelerationSetTraverser](#)

#### 3.11.3.14 `RTresult RTAPI rtAccelerationSetTraverser ( RTacceleration acceleration, const char * traverser )`

Specifies the traverser to be used for an acceleration structure.

#### Description

[rtAccelerationSetTraverser](#) specifies the method used to traverse the ray tracing acceleration structure represented by *acceleration*. A traverser must be set for the acceleration structure to pass validation. The current active traverser can be changed at any time.

An acceleration structure is only valid with a correct pair of builder and traverser. The builder type is specified using [rtAccelerationSetBuilder](#). For a list of valid combinations of builders and traversers, see below. For a description of the individual builders, see [rtAccelerationSetBuilder](#).

*traverser* can take one of the following values:

- "NoAccel": Linearly loops through the list of primitives to intersect. This is highly inefficient in all but the most trivial scenarios (but there it can provide good performance due to very little overhead). Valid builder types: "NoAccel".
- "Bvh": Optimized traversal of generic bounding volume hierarchies. Valid builder types: "Trbvh", "Sbvh", "Bvh", "MedianBvh", "Lbvh".
- "BvhCompact": Optimized traversal of bounding volume hierarchies for large datasets when virtual memory is turned on. It compresses the BVH data in 4 times before uploading to the device. And decompress the BVH data in real-time during traversal of a bounding volume hierarchy. Valid builder types: "Sbvh", "Bvh", "MedianBvh", "Lbvh".
- "KdTree": Standard traversal for kd-trees. Valid builder types: "TriangleKdTree".



## Parameters

<code>in</code>	<code>acceleration</code>	The acceleration structure handle
<code>in</code>	<code>traverser</code>	String value specifying the traverser type

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtAccelerationSetTraverser](#) was introduced in OptiX 1.0.

**See also** [rtAccelerationGetTraverser](#), [rtAccelerationSetBuilder](#), [rtAccelerationSetProperty](#)

3.11.3.15 `RTresult RTAPI rtAccelerationValidate ( RTacceleration acceleration )`

Validates the state of an acceleration structure.

## Description

[rtAccelerationValidate](#) checks *acceleration* for completeness. If *acceleration* is not valid, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

<code>in</code>	<code>acceleration</code>	The acceleration structure handle
-----------------	---------------------------	-----------------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtAccelerationValidate](#) was introduced in OptiX 1.0.

**See also** [rtAccelerationCreate](#)

3.11.3.16 `RTresult RTAPI rtBufferBindProgressiveStream ( RTbuffer stream, RTbuffer source )`

Bind a stream buffer to an output buffer source.

## Description

Binds an output buffer to a progressive stream. The output buffer thereby becomes the data source for the stream. To form a valid output/stream pair, the stream buffer must be of format [RT\\_FORMAT\\_UNSIGNED\\_BYTE4](#), and the output buffer must be of format [RT\\_FORMAT\\_FLOAT3](#) or [RT\\_FORMAT\\_FLOAT4](#). The use of [RT\\_FORMAT\\_FLOAT4](#) is recommended for performance reasons, even if the fourth component is unused. The output buffer must be of type [RT\\_BUFFER\\_OUTPUT](#); it may not be of type [RT\\_BUFFER\\_INPUT\\_OUTPUT](#).

## Parameters

<code>in</code>	<code>stream</code>	The stream buffer for which the source is to be specified
<code>in</code>	<code>source</code>	The output buffer to function as the stream's source

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtBufferBindProgressiveStream](#) was introduced in OptiX 3.8.

**See also** [rtBufferCreate](#) [rtBufferSetAttribute](#) [rtBufferGetAttribute](#)

3.11.3.17 **RTresult** RTAPI [rtBufferCreate](#) ( **RTcontext** *context*, unsigned int *bufferdesc*, **RTbuffer** \* *buffer* )

Creates a new buffer object.

## Description

[rtBufferCreate](#) allocates and returns a new handle to a new buffer object in \**buffer* associated with *context*. The backing storage of the buffer is managed by OptiX. A buffer is specified by a bitwise *or* combination of a *type* and *flags* in *bufferdesc*. The supported types are:

- [RT\\_BUFFER\\_INPUT](#)
- [RT\\_BUFFER\\_OUTPUT](#)
- [RT\\_BUFFER\\_INPUT\\_OUTPUT](#)
- [RT\\_BUFFER\\_PROGRESSIVE\\_STREAM](#)

The type values are used to specify the direction of data flow from the host to the OptiX devices. [RT\\_BUFFER\\_INPUT](#) specifies that the host may only write to the buffer and the device may only read from the buffer. [RT\\_BUFFER\\_OUTPUT](#) specifies the opposite, read only access on the host and write only access on the device. Devices and the host may read and write from buffers of type [RT\\_BUFFER\\_INPUT\\_OUTPUT](#). Reading or writing to a buffer of the incorrect type (e.g., the host writing to a buffer of type [RT\\_BUFFER\\_OUTPUT](#)) is undefined. [RT\\_BUFFER\\_PROGRESSIVE\\_STREAM](#) is used to receive stream updates generated by progressive launches (see [rtContextLaunchProgressive2D](#)).

The supported flags are:

- [RT\\_BUFFER\\_GPU\\_LOCAL](#)
- [RT\\_BUFFER\\_COPY\\_ON\\_DIRTY](#)
- [RT\\_BUFFER\\_LAYERED](#)
- [RT\\_BUFFER\\_CUBEMAP](#)

If [RT\\_BUFFER\\_LAYERED](#) flag is set, buffer depth specifies the number of layers, not the depth of a 3D buffer. If [RT\\_BUFFER\\_CUBEMAP](#) flag is set, buffer depth specifies the number of cube faces, not the depth of a 3D buffer. See details in [rtBufferSetSize3D](#)

Flags can be used to optimize data transfers between the host and its devices. The flag [RT\\_BUFFER\\_GPU\\_LOCAL](#) can only be used in combination with [RT\\_BUFFER\\_INPUT\\_OUTPUT](#). [RT\\_BUFFER\\_INPUT\\_OUTPUT](#) and [RT\\_BUFFER\\_GPU\\_LOCAL](#) used together specify a buffer that allows the host to *only* write, and the device to read *and* write data. The written data will never be visible on the host side and will generally not be visible on other devices.

If [rtBufferSetDevicePointer](#) or [rtBufferGetDevicePointer](#) have been called for a single device for a given buffer, the user can change the buffer's content on that device. The new buffer contents must be synchronized to all devices. These synchronization copies occur at every [rtContextLaunch](#), unless the buffer is declared with [RT\\_BUFFER\\_COPY\\_ON\\_DIRTY](#). In this case, use [rtBufferMarkDirty](#) to notify OptiX that the buffer has been dirtied and must be synchronized.

Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *buffer* is *NULL*.

## Parameters

in	<i>context</i>	The context to create the buffer in
in	<i>bufferdesc</i>	Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer
out	<i>buffer</i>	The return handle for the buffer object

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtBufferCreate](#) was introduced in OptiX 1.0.

[RT\\_BUFFER\\_GPU\\_LOCAL](#) was introduced in OptiX 2.0.

**See also** [rtBufferCreateFromGLBO](#), [rtBufferDestroy](#), [rtBufferMarkDirty](#) [rtBufferBindProgressiveStream](#)

### 3.11.3.18 RTresult RTAPI rtBufferDestroy ( RTbuffer *buffer* )

Destroys a buffer object.

**Description**

[rtBufferDestroy](#) removes *buffer* from its context and deletes it. *buffer* should be a value returned by [rtBufferCreate](#). After the call, *buffer* is no longer a valid handle. Any API object that referenced *buffer* will have its reference invalidated.

**Parameters**

in	<i>buffer</i>	Handle of the buffer to destroy
----	---------------	---------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtBufferDestroy](#) was introduced in OptiX 1.0.

**See also** [rtBufferCreate](#), [rtBufferCreateFromGLBO](#)

### 3.11.3.19 RTresult RTAPI rtBufferGetAttribute ( RTbuffer *buffer*, RTbufferattribute *attrib*, RTsize *size*, void \* *p* )

Query a buffer attribute.

**Description**

[rtBufferGetAttribute](#) is used to query buffer attributes. For a list of available attributes, please refer to [rtBufferSetAttribute](#).

## Parameters

in	<i>buffer</i>	The buffer to query the attribute from
in	<i>attrib</i>	The attribute to query
in	<i>size</i>	The size of the attribute value, in bytes. For string attributes, this is the maximum buffer size the returned string will use (including a terminating null character).
out	<i>p</i>	Pointer to the attribute value to be filled in. Must point to valid memory of at least <i>size</i> bytes.

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtBufferGetAttribute](#) was introduced in OptiX 3.8.

See also [rtBufferSetAttribute](#)

3.11.3.20 `RTresult RTAPI rtBufferGetContext ( RTbuffer buffer, RTcontext * context )`

Returns the context object that created this buffer.

## Description

[rtBufferGetContext](#) returns a handle to the context that created *buffer* in *\*context*. If *\*context* is `NULL`, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

in	<i>buffer</i>	The buffer to be queried for its context
out	<i>context</i>	The return handle for the buffer's context

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtBufferGetContext](#) was introduced in OptiX 1.0.

See also [rtContextCreate](#)

3.11.3.21 `RTresult RTAPI rtBufferGetDimensionality ( RTbuffer buffer, unsigned int * dimensionality )`

Gets the dimensionality of this buffer object.

## Description

[rtBufferGetDimensionality](#) returns the dimensionality of *buffer* in *\*dimensionality*. The value returned will be one of 1, 2 or 3, corresponding to 1D, 2D and 3D buffers, respectively.

## Parameters

in	<i>buffer</i>	The buffer to be queried for its dimensionality
out	<i>dimensionality</i>	The return handle for the buffer's dimensionality

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtBufferGetDimensionality](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetSize{1-2-3}D](#)

3.11.3.22 `RtResult RTAPI rtBufferGetElementSize ( RTbuffer buffer, RTsize * size_of_element )`

Returns the size of a buffer's individual elements.

## Description

[rtBufferGetElementSize](#) queries the size of a buffer's elements. The target buffer is specified by *buffer*, which should be a value returned by [rtBufferCreate](#). The size, in bytes, of the buffer's individual elements is returned in *\*element\_size\_return*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if given a *NULL* pointer.

## Parameters

in	<i>buffer</i>	Specifies the buffer to be queried
out	<i>size_of_element</i>	Returns the size of the buffer's individual elements

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_UNKNOWN](#)

## History

[rtBufferGetElementSize](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetElementSize](#), [rtBufferCreate](#)

3.11.3.23 `RtResult RTAPI rtBufferGetFormat ( RTbuffer buffer, RTformat * format )`

Gets the format of this buffer.

## Description

[rtBufferGetFormat](#) returns, in *\*format*, the format of *buffer*. See [rtBufferSetFormat](#) for a listing of [RTbuffer](#) values.

## Parameters

in	<i>buffer</i>	The buffer to be queried for its format
out	<i>format</i>	The return handle for the buffer's format

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtBufferGetFormat](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetFormat](#), [rtBufferGetFormat](#)

### 3.11.3.24 `RTresult RTAPI rtBufferGetId ( RTbuffer buffer, int * buffer_id )`

Gets an id suitable for use with buffers of buffers.

**Description**

[rtBufferGetId](#) returns an ID for the provided buffer. The returned ID is used on the device to reference the buffer. It needs to be copied into a buffer of type [RT\\_FORMAT\\_BUFFER\\_ID](#) or used in a [rtBufferId](#) object.. If *\*buffer\_id* is `NULL` or the *buffer* is not a valid `RTbuffer`, returns [RT\\_ERROR\\_INVALID\\_VALUE](#). [RT\\_BUFFER\\_ID\\_NULL](#) can be used as a sentinel for a non-existent buffer, since this value will never be returned as a valid buffer id.

**Parameters**

in	<i>buffer</i>	The buffer to be queried for its id
out	<i>buffer_id</i>	The returned ID of the buffer

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtBufferGetId](#) was introduced in OptiX 3.5.

**See also** [rtContextGetBufferFromId](#)

### 3.11.3.25 `RTresult RTAPI rtBufferGetMipLevelCount ( RTbuffer buffer, unsigned int * level )`

Gets the number of mipmap levels of this buffer object.

**Description**

[rtBufferGetMipLevelCount](#) returns the number of mipmap levels. Default number of MIP levels is 1.

**Parameters**

in	<i>buffer</i>	The buffer to be queried for its number of mipmap levels
out	<i>level</i>	The return number of mipmap levels

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)

- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtBufferGetMipLevelCount](#) was introduced in OptiX 3.9.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.11.3.26 **RTresult** RTAPI [rtBufferGetMipLevelSize1D](#) ( *RTbuffer* *buffer*, unsigned int *level*, RTsize \* *width* )

Gets the width of buffer specific MIP level.

### Description

[rtBufferGetMipLevelSize1D](#) stores the width of *buffer* in \**width*.

#### Parameters

in	<i>buffer</i>	The buffer to be queried for its dimensions
in	<i>level</i>	The buffer MIP level index to be queried for its dimensions
out	<i>width</i>	The return handle for the buffer's width <b>Return values</b>

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtBufferGetMipLevelSize1D](#) was introduced in OptiX 3.9.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.11.3.27 **RTresult** RTAPI [rtBufferGetMipLevelSize2D](#) ( *RTbuffer* *buffer*, unsigned int *level*, RTsize \* *width*, RTsize \* *height* )

Gets the width, height of buffer specific MIP level.

### Description

[rtBufferGetMipLevelSize2D](#) stores the width, height of *buffer* in \**width* and \**height* respectively.

#### Parameters

in	<i>buffer</i>	The buffer to be queried for its dimensions
in	<i>level</i>	The buffer MIP level index to be queried for its dimensions
out	<i>width</i>	The return handle for the buffer's width
out	<i>height</i>	The return handle for the buffer's height <b>Return values</b>

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtBufferGetMipLevelSize2D](#) was introduced in OptiX 3.9.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

**3.11.3.28** **RTresult** RTAPI [rtBufferGetMipLevelSize3D](#) ( **RTbuffer** *buffer*, unsigned int *level*, RTsize \* *width*, RTsize \* *height*, RTsize \* *depth* )

Gets the width, height and depth of buffer specific MIP level.

## Description

[rtBufferGetMipLevelSize3D](#) stores the width, height and depth of *buffer* in \**width*, \**height* and \**depth*, respectively.

## Parameters

in	<i>buffer</i>	The buffer to be queried for its dimensions
in	<i>level</i>	The buffer MIP level index to be queried for its dimensions
out	<i>width</i>	The return handle for the buffer's width
out	<i>height</i>	The return handle for the buffer's height
out	<i>depth</i>	The return handle for the buffer's depth <b>Return values</b>

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtBufferGetMipLevelSize3D](#) was introduced in OptiX 3.9.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

**3.11.3.29** **RTresult** RTAPI [rtBufferGetProgressiveUpdateReady](#) ( **RTbuffer** *buffer*, int \* *ready*, unsigned int \* *subframe\_count*, unsigned int \* *max\_subframes* )

Check whether stream buffer content has been updated by a Progressive Launch.

## Description

Returns whether or not the result of a progressive launch in *buffer* has been updated since the last time this function was called. A client application should use this call in its main render/display loop to poll for frame refreshes after initiating a progressive launch. If *subframe\_count* and *max\_subframes* are non-null, they will be filled with the corresponding counters if and only if *ready* returns 1.

Note that this call does not stop a progressive render.

## Parameters

in	<i>buffer</i>	The stream buffer to be queried
out	<i>ready</i>	Ready flag. Will be set to 1 if an update is available, or 0 if no update is available.
out	<i>subframe_count</i>	The number of subframes accumulated in the latest result
out	<i>max_subframes</i>	The <i>max_subframes</i> parameter as specified in the call to <a href="#">rtContextLaunchProgressive2D</a>

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)



- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtBufferGetProgressiveUpdateReady](#) was introduced in OptiX 3.8.

**See also** [rtContextLaunchProgressive2D](#)

#### 3.11.3.30 RTresult RTAPI rtBufferGetSize1D ( RTbuffer *buffer*, RTsize \* *width* )

Get the width of this buffer.

### Description

[rtBufferGetSize1D](#) stores the width of *buffer* in \**width*.

#### Parameters

in	<i>buffer</i>	The buffer to be queried for its dimensions
out	<i>width</i>	The return handle for the buffer's width

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtBufferGetSize1D](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

#### 3.11.3.31 RTresult RTAPI rtBufferGetSize2D ( RTbuffer *buffer*, RTsize \* *width*, RTsize \* *height* )

Gets the width and height of this buffer.

### Description

[rtBufferGetSize2D](#) stores the width and height of *buffer* in \**width* and \**height*, respectively.

#### Parameters

in	<i>buffer</i>	The buffer to be queried for its dimensions
out	<i>width</i>	The return handle for the buffer's width
out	<i>height</i>	The return handle for the buffer's height

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtBufferGetSize2D](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

#### 3.11.3.32 `RTresult RTAPI rtBufferGetSize3D ( RTbuffer buffer, RTsize * width, RTsize * height, RTsize * depth )`

Gets the width, height and depth of this buffer.

##### Description

[rtBufferGetSize3D](#) stores the width, height and depth of *buffer* in \**width*, \**height* and \**depth*, respectively.

##### Parameters

in	<i>buffer</i>	The buffer to be queried for its dimensions
out	<i>width</i>	The return handle for the buffer's width
out	<i>height</i>	The return handle for the buffer's height
out	<i>depth</i>	The return handle for the buffer's depth

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

##### History

[rtBufferGetSize3D](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSizev](#)

#### 3.11.3.33 `RTresult RTAPI rtBufferGetSizev ( RTbuffer buffer, unsigned int dimensionality, RTsize * dims )`

Gets the dimensions of this buffer.

##### Description

[rtBufferGetSizev](#) stores the dimensions of *buffer* in \**dims*. The number of dimensions returned is specified by *dimensionality*. The storage at *dims* must be large enough to hold the number of requested buffer dimensions.

##### Parameters

in	<i>buffer</i>	The buffer to be queried for its dimensions
in	<i>dimensionality</i>	The number of requested dimensions
out	<i>dims</i>	The array of dimensions to store to

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

##### History

[rtBufferGetSizev](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#)

#### 3.11.3.34 `RTresult RTAPI rtBufferMap ( RTbuffer buffer, void ** user_pointer )`

Maps a buffer object to the host.

##### Description

[rtBufferMap](#) returns a pointer, accessible by the host, in `*user_pointer` that contains a mapped copy of the contents of `buffer`. The memory pointed to by `*user_pointer` can be written to or read from, depending on the type of `buffer`. For example, this code snippet demonstrates creating and filling an input buffer with floats.

```
RTbuffer buffer;
float* data;
rtBufferCreate(context, RT_BUFFER_INPUT, &buffer);
rtBufferSetFormat(buffer, RT_FORMAT_FLOAT);
rtBufferSetSize1D(buffer, 10);
rtBufferMap(buffer, (void*)&data);
for(int i = 0; i < 10; ++i)
    data[i] = 4.f * i;
rtBufferUnmap(buffer);
```

If `buffer` has already been mapped, returns [RT\\_ERROR\\_ALREADY\\_MAPPED](#).

Note that this call does not stop a progressive render if called on a stream buffer.

##### Parameters

in	<code>buffer</code>	The buffer to be mapped
out	<code>user_pointer</code>	Return handle to a user pointer where the buffer will be mapped to

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_ALREADY\\_MAPPED](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

##### History

[rtBufferMap](#) was introduced in OptiX 1.0.

**See also** [rtBufferUnmap](#), [rtBufferMapEx](#), [rtBufferUnmapEx](#)

#### 3.11.3.35 `RTresult RTAPI rtBufferMapEx ( RTbuffer buffer, unsigned int map_flags, unsigned int level, void * user_owned, void ** optix_owned )`

Maps mipmap level of buffer object to the host.

##### Description

[rtBufferMapEx](#) returns a pointer, accessible by the host, in `*optix_owned` if `user_owned` is null otherwise `user_owned` is used for the mapping the contents of `level`. [rtBufferMap](#) is equivalent to [rtBufferMapEx](#) with `level = 0` and [RT\\_BUFFER\\_MAP\\_READ\\_WRITE](#) flag and `user_owned = NULL`.

Note that this call does not stop a progressive render if called on a stream buffer.

**Parameters**

in	<i>buffer</i>	The buffer to be mapped
in	<i>map_flag</i>	Declare access pattern to the mapped buffer memory. <b>Not supported yet and must be <a href="#">RT_BUFFER_MAP_READ_WRITE</a>.</b>
in	<i>level</i>	The mipmap level to be mapped
in	<i>user_owned</i>	Preallocated memory for mapping, can be null, then it is allocated internally. <b>Not supported yet and must be NULL.</b>
out	<i>optix_owned</i>	Return handle to a user pointer where the buffer will be mapped to

The supported flags are:

- [RT\\_BUFFER\\_MAP\\_READ](#)
- [RT\\_BUFFER\\_MAP\\_WRITE](#)
- [RT\\_BUFFER\\_MAP\\_READ\\_WRITE](#)
- [RT\\_BUFFER\\_MAP\\_WRITE\\_DISCARD](#)

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_ALREADY\\_MAPPED](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtBufferMapEx](#) was introduced in OptiX 3.9.

**See also** [rtBufferMap](#), [rtBufferUnmap](#), [rtBufferUnmapEx](#)

3.11.3.36 **RTresult** RTAPI `rtBufferSetAttribute ( RTbuffer buffer, RTbufferattribute attrib, RTsize size, void * p )`

Set a buffer attribute.

**Description**

Sets a buffer attribute. Currently, all available attributes refer to stream buffers only, and attempting to set them on a non-stream buffer will generate an error.

Each attribute can have a different size. The sizes are given in the following list:

- [RT\\_BUFFER\\_ATTRIBUTE\\_STREAM\\_FORMAT](#) `strlen(input_string)`
- [RT\\_BUFFER\\_ATTRIBUTE\\_STREAM\\_BITRATE](#) `sizeof(int)`
- [RT\\_BUFFER\\_ATTRIBUTE\\_STREAM\\_FPS](#) `sizeof(int)`
- [RT\\_BUFFER\\_ATTRIBUTE\\_STREAM\\_GAMMA](#) `sizeof(float)`

[RT\\_BUFFER\\_ATTRIBUTE\\_STREAM\\_FORMAT](#) sets the encoding format used for streams sent over the network, specified as a string. The default is "auto". Various other common stream and image formats are available (e.g. "h264", "png"). This attribute has no effect if the progressive API is used locally.

[RT\\_BUFFER\\_ATTRIBUTE\\_STREAM\\_BITRATE](#) sets the target bitrate for streams sent over the network, if the stream format supports it. The data is specified as a 32-bit integer. The default is 5000000. This attribute has no effect if the progressive API is used locally or if the stream format does not support variable bitrates.

[RT\\_BUFFER\\_ATTRIBUTE\\_STREAM\\_FPS](#) sets the target update rate per second for streams sent over the network, if the stream format supports it. The data is specified as a 32-bit integer. The default is 30. This attribute has no effect if the progressive API is used locally or if the stream format does not support variable framerates.

[RT\\_BUFFER\\_ATTRIBUTE\\_STREAM\\_GAMMA](#) sets the gamma value for the built-in tonemapping operator. The data is specified as a 32-bit float, the default is 1.0. Tonemapping is executed before encoding the accumulated output into the stream, i.e. on the server side if remote rendering is used. See the section on Buffers below for more details.

#### Parameters

<i>in</i>	<i>buffer</i>	The buffer on which to set the attribute
<i>in</i>	<i>attrib</i>	The attribute to set
<i>in</i>	<i>size</i>	The size of the attribute value, in bytes
<i>in</i>	<i>p</i>	Pointer to the attribute value

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtBufferSetAttribute](#) was introduced in OptiX 3.8.

See also [rtBufferGetAttribute](#)

#### 3.11.3.37 `RTresult RTAPI rtBufferSetElementSize ( RTbuffer buffer, RTsize size_of_element )`

Modifies the size in bytes of a buffer's individual elements.

#### Description

[rtBufferSetElementSize](#) modifies the size in bytes of a buffer's user-formatted elements. The target buffer is specified by *buffer*, which should be a value returned by [rtBufferCreate](#) and should have format [RT\\_FORMAT\\_USER](#). The new size of the buffer's individual elements is specified by *element\_size* and should not be 0. If the buffer has format [RT\\_FORMAT\\_USER](#), and *element\_size* is not 0, then the buffer's individual element size is set to *element\_size* and all storage associated with the buffer is reset. Otherwise, this call has no effect and returns either [RT\\_ERROR\\_TYPE\\_MISMATCH](#) if the buffer does not have format [RT\\_FORMAT\\_USER](#) or [RT\\_ERROR\\_INVALID\\_VALUE](#) if the buffer has format [RT\\_FORMAT\\_USER](#) but *element\_size* is 0.

#### Parameters

<i>in</i>	<i>buffer</i>	Specifies the buffer to be modified
<i>in</i>	<i>size_of_element</i>	Specifies the new size in bytes of the buffer's individual elements

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

#### History

[rtBufferSetElementSize](#) was introduced in OptiX 1.0.

See also [rtBufferGetElementSize](#), [rtBufferCreate](#)

### 3.11.3.38 `RTresult RTAPI rtBufferSetFormat ( RTbuffer buffer, RTformat format )`

Sets the format of this buffer.

#### Description

`rtBufferSetFormat` changes the *format* of *buffer* to the specified value. The data elements of the buffer will have the specified type and can either be vector formats, or a user-defined type whose size is specified with `rtBufferSetElementSize`. Possible values for *format* are:

- `RT_FORMAT_HALF`
- `RT_FORMAT_HALF2`
- `RT_FORMAT_HALF3`
- `RT_FORMAT_HALF4`
- `RT_FORMAT_FLOAT`
- `RT_FORMAT_FLOAT2`
- `RT_FORMAT_FLOAT3`
- `RT_FORMAT_FLOAT4`
- `RT_FORMAT_BYTE`
- `RT_FORMAT_BYTE2`
- `RT_FORMAT_BYTE3`
- `RT_FORMAT_BYTE4`
- `RT_FORMAT_UNSIGNED_BYTE`
- `RT_FORMAT_UNSIGNED_BYTE2`
- `RT_FORMAT_UNSIGNED_BYTE3`
- `RT_FORMAT_UNSIGNED_BYTE4`
- `RT_FORMAT_SHORT`
- `RT_FORMAT_SHORT2`
- `RT_FORMAT_SHORT3`
- `RT_FORMAT_SHORT4`
- `RT_FORMAT_UNSIGNED_SHORT`
- `RT_FORMAT_UNSIGNED_SHORT2`
- `RT_FORMAT_UNSIGNED_SHORT3`
- `RT_FORMAT_UNSIGNED_SHORT4`
- `RT_FORMAT_INT`
- `RT_FORMAT_INT2`
- `RT_FORMAT_INT3`
- `RT_FORMAT_INT4`
- `RT_FORMAT_UNSIGNED_INT`
- `RT_FORMAT_UNSIGNED_INT2`
- `RT_FORMAT_UNSIGNED_INT3`
- `RT_FORMAT_UNSIGNED_INT4`
- `RT_FORMAT_USER`

## Parameters

<i>in</i>	<i>buffer</i>	The buffer to have its format set
<i>in</i>	<i>format</i>	The target format of the buffer

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtBufferSetFormat](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetFormat](#), [rtBufferGetFormat](#), [rtBufferGetFormat](#), [rtBufferGetElementSize](#), [rtBufferSetElementSize](#)

### 3.11.3.39 `RTresult RTAPI rtBufferSetMipLevelCount ( RTbuffer buffer, unsigned int levels )`

Sets the MIP level count of a buffer.

## Description

[rtBufferSetMipLevelCount](#) sets the number of MIP levels to *levels*. Default number of MIP levels is 1. Fails with [RT\\_ERROR\\_ALREADY\\_MAPPED](#) if called on a buffer that is mapped.

## Parameters

<i>in</i>	<i>buffer</i>	The buffer to be resized
<i>in</i>	<i>width</i>	The width of the resized buffer
<i>in</i>	<i>levels</i>	Number of mip levels

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_ALREADY\\_MAPPED](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtBufferSetMipLevelCount](#) was introduced in OptiX 3.9.

**See also** [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

### 3.11.3.40 `RTresult RTAPI rtBufferSetSize1D ( RTbuffer buffer, RTsize width )`

Sets the width and dimensionality of this buffer.

## Description

[rtBufferSetSize1D](#) sets the dimensionality of *buffer* to 1 and sets its width to *width*. Fails with [RT\\_ERROR\\_ALREADY\\_MAPPED](#) if called on a buffer that is mapped.

## Parameters

in	<i>buffer</i>	The buffer to be resized
in	<i>width</i>	The width of the resized buffer

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_ALREADY\\_MAPPED](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtBufferSetSize1D](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

#### 3.11.3.41 RTresult RTAPI rtBufferSetSize2D ( RTbuffer *buffer*, RTsize *width*, RTsize *height* )

Sets the width, height and dimensionality of this buffer.

## Description

[rtBufferSetSize2D](#) sets the dimensionality of *buffer* to 2 and sets its width and height to *width* and *height*, respectively. If *width* or *height* is zero, they both must be zero. Fails with [RT\\_ERROR\\_ALREADY\\_MAPPED](#) if called on a buffer that is mapped.

## Parameters

in	<i>buffer</i>	The buffer to be resized
in	<i>width</i>	The width of the resized buffer
in	<i>height</i>	The height of the resized buffer

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_ALREADY\\_MAPPED](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtBufferSetSize2D](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)



3.11.3.42 `RTresult RTAPI rtBufferSetSize3D ( RTbuffer buffer, RTsize width, RTsize height, RTsize depth )`

Sets the width, height, depth and dimensionality of a buffer.

**Description**

`rtBufferSetSize3D` sets the dimensionality of *buffer* to 3 and sets its width, height and depth to *width*, *height* and *depth*, respectively. If *width*, *height* or *depth* is zero, they all must be zero.

A 1D layered mipmapped buffer is allocated if *height* is 1 and the `RT_BUFFER_LAYERED` flag was set at buffer creating. The number of layers is determined by the *depth*. A 2D layered mipmapped buffer is allocated if the `RT_BUFFER_LAYERED` flag was set at buffer creating. The number of layers is determined by the *depth*. A cubemap mipmapped buffer is allocated if the `RT_BUFFER_CUBEMAP` flag was set at buffer creating. *width* must be equal to *height* and the number of cube faces is determined by the *depth*, it must be six or a multiple of six, if the `RT_BUFFER_LAYERED` flag was also set. Layered, mipmapped and cubemap buffers are supported only as texture buffers.

Fails with `RT_ERROR_ALREADY_MAPPED` if called on a buffer that is mapped.

**Parameters**

in	<i>buffer</i>	The buffer to be resized
in	<i>width</i>	The width of the resized buffer
in	<i>height</i>	The height of the resized buffer
in	<i>depth</i>	The depth of the resized buffer

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_ALREADY_MAPPED`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`

**History**

`rtBufferSetSize3D` was introduced in OptiX 1.0.

**See also** `rtBufferSetMipLevelCount`, `rtBufferSetSize1D`, `rtBufferSetSize2D`, `rtBufferSetSizev`, `rtBufferGetMipLevelSize1D`, `rtBufferGetMipLevelSize2D`, `rtBufferGetMipLevelSize3D`, `rtBufferGetMipLevelCount`, `rtBufferGetSize1D`, `rtBufferGetSize2D`, `rtBufferGetSize3D`, `rtBufferGetSizev`

3.11.3.43 `RTresult RTAPI rtBufferSetSizev ( RTbuffer buffer, unsigned int dimensionality, const RTsize * dims )`

Sets the dimensionality and dimensions of a buffer.

**Description**

`rtBufferSetSizev` sets the dimensionality of *buffer* to *dimensionality* and sets the dimensions of the buffer to the values stored at *\*dims*, which must contain a number of values equal to *dimensionality*. If any of values of *dims* is zero they must all be zero.

**Parameters**

in	<i>buffer</i>	The buffer to be resized
in	<i>dimensionality</i>	The dimensionality the buffer will be resized to
in	<i>dims</i>	The array of sizes for the dimension of the resize

**Return values**

Relevant return values:

- `RT_SUCCESS`

- [RT\\_ERROR\\_ALREADY\\_MAPPED](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtBufferSetSizev](#) was introduced in OptiX 1.0.

**See also** [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

#### 3.11.3.44 `RTresult RTAPI rtBufferUnmap ( RTbuffer buffer )`

Unmaps a buffer's storage from the host.

### Description

[rtBufferUnmap](#) unmaps a buffer from the host after a call to [rtBufferMap](#). `rtContextLaunch` cannot be called while buffers are still mapped to the host. A call to [rtBufferUnmap](#) that does not follow a matching [rtBufferMap](#) call will return [RT\\_ERROR\\_INVALID\\_VALUE](#).

Note that this call does not stop a progressive render if called with a stream buffer.

### Parameters

<code>in</code>	<code>buffer</code>	The buffer to unmap
-----------------	---------------------	---------------------

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtBufferUnmap](#) was introduced in OptiX 1.0.

**See also** [rtBufferMap](#), [rtBufferMapEx](#), [rtBufferUnmapEx](#)

#### 3.11.3.45 `RTresult RTAPI rtBufferUnmapEx ( RTbuffer buffer, unsigned int level )`

Unmaps mipmap level storage from the host.

### Description

[rtBufferUnmapEx](#) unmaps buffer level from the host after a call to [rtBufferMapEx](#). `rtContextLaunch` cannot be called while buffers are still mapped to the host. A call to [rtBufferUnmapEx](#) that does not follow a matching [rtBufferMapEx](#) call will return [RT\\_ERROR\\_INVALID\\_VALUE](#). [rtBufferUnmap](#) is equivalent to [rtBufferUnmapEx](#) with `level = 0`.

Note that this call does not stop a progressive render if called with a stream buffer.

### Parameters

<code>in</code>	<code>buffer</code>	The buffer to unmap
-----------------	---------------------	---------------------

<code>in</code>	<code>level</code>	The mipmap level to unmap
-----------------	--------------------	---------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtBufferUnmapEx](#) was introduced in OptiX 3.9.

**See also** [rtBufferMap](#), [rtBufferUnmap](#), [rtBufferMapEx](#)

3.11.3.46 **RTresult RTAPI rtBufferValidate ( RTbuffer *buffer* )**

Validates the state of a buffer.

**Description**

[rtBufferValidate](#) checks *buffer* for completeness. If *buffer* has not had its dimensionality, size or format set, this call will return [RT\\_ERROR\\_INVALID\\_CONTEXT](#).

**Parameters**

<code>in</code>	<code>buffer</code>	The buffer to validate
-----------------	---------------------	------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtBufferValidate](#) was introduced in OptiX 1.0.

**See also** [rtBufferCreate](#), [rtBufferCreateFromGLBO](#) [rtContextValidate](#)

3.11.3.47 **RTresult RTAPI rtContextCompile ( RTcontext *context* )**

Compiles a context object.

**Description**

[rtContextCompile](#) creates a final computation kernel from the given context's programs and scene hierarchy. This kernel will be executed upon subsequent invocations of [rtContextLaunch](#).

Calling [rtContextCompile](#) is not strictly necessary since any changes to the scene specification or programs will cause an internal compilation upon the next [rtContextLaunch](#) call. [rtContextCompile](#) allows the application to control when the compilation work occurs.

Conversely, if no changes to the scene specification or programs have occurred since the last compilation, [rtContextCompile](#) and [rtContextLaunch](#) will not perform a recompilation.

## Parameters

<code>in</code>	<code>context</code>	The context to be compiled
-----------------	----------------------	----------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_INVALID\\_SOURCE](#)

## History

[rtContextCompile](#) was introduced in OptiX 1.0.

**See also** [rtContextLaunch](#)

3.11.3.48 **RTresult** RTAPI `rtContextCreate` ( **RTcontext** \* *context* )

Creates a new context object.

## Description

[rtContextCreate](#) allocates and returns a handle to a new context object. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if passed a `NULL` pointer.

## Parameters

<code>out</code>	<code>context</code>	Handle to context for return value
------------------	----------------------	------------------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_NO\\_DEVICE](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextCreate](#) was introduced in OptiX 1.0.

**See also**

3.11.3.49 **RTresult** RTAPI `rtContextDeclareVariable` ( **RTcontext** *context*, `const char` \* *name*, **RTvariable** \* *v* )

Declares a new named variable associated with this context.

## Description

[rtContextDeclareVariable](#) - Declares a new variable named *name* and associated with this context. Only a single variable of a given name can exist for a given context and any attempt to create multiple variables with the same name will cause a failure with a return value of [RT\\_ERROR\\_VARIABLE\\_REDECLARED](#). Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if passed a `NULL` pointer. Return [RT\\_ERROR\\_ILLEGAL\\_SYMBOL](#) if *name* is not syntactically valid.

## Parameters

in	<i>context</i>	The context node to which the variable will be attached
in	<i>name</i>	The name that identifies the variable to be queried
out	<i>v</i>	Pointer to variable handle used to return the new object

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_VARIABLE\\_REDECLARED](#)

**History**

[rtContextDeclareVariable](#) was introduced in OptiX 1.0.

**See also** [rtGeometryDeclareVariable](#), [rtGeometryInstanceDeclareVariable](#), [rtMaterialDeclareVariable](#), [rtProgramDeclareVariable](#), [rtSelectorDeclareVariable](#), [rtContextGetVariable](#), [rtContextGetVariableCount](#), [rtContextQueryVariable](#), [rtContextRemoveVariable](#)

**3.11.3.50 RTresult RTAPI rtContextDestroy ( RTcontext context )**

Destroys a context and frees all associated resources.

**Description**

[rtContextDestroy](#) frees all resources, including OptiX objects, associated with this object. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if passed a `NULL` context. [RT\\_ERROR\\_LAUNCH\\_FAILED](#) may be returned if a previous call to [rtContextLaunch](#) failed.

**Parameters**

in	<i>context</i>	Handle of the context to destroy
----	----------------	----------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_LAUNCH\\_FAILED](#)

**History**

[rtContextDestroy](#) was introduced in OptiX 1.0.

**See also** [rtContextCreate](#)

**3.11.3.51 RTresult RTAPI rtContextGetAttribute ( RTcontext context, RTcontextattribute attrib, RTsize size, void \* p )**

Returns an attribute specific to an OptiX context.

**Description**

[rtContextGetAttribute](#) returns in *p* the value of the per context attribute specified by *attrib*.

Each attribute can have a different size. The sizes are given in the following list:

- [RT\\_CONTEXT\\_ATTRIBUTE\\_MAX\\_TEXTURE\\_COUNT](#) `sizeof(int)`
- [RT\\_CONTEXT\\_ATTRIBUTE\\_CPU\\_NUM\\_THREADS](#) `sizeof(int)`
- [RT\\_CONTEXT\\_ATTRIBUTE\\_USED\\_HOST\\_MEMORY](#) `sizeof(RTsize)`
- [RT\\_CONTEXT\\_ATTRIBUTE\\_GPU\\_PAGING\\_ACTIVE](#) `sizeof(int)`

- [RT\\_CONTEXT\\_ATTRIBUTE\\_GPU\\_PAGING\\_FORCED\\_OFF](#) sizeof(int)
- [RT\\_CONTEXT\\_ATTRIBUTE\\_AVAILABLE\\_DEVICE\\_MEMORY](#) sizeof(RTsize)

[RT\\_CONTEXT\\_ATTRIBUTE\\_MAX\\_TEXTURE\\_COUNT](#) queries the maximum number of textures handled by OptiX. For OptiX versions below 2.5 this value depends on the number of textures supported by CUDA.

[RT\\_CONTEXT\\_ATTRIBUTE\\_CPU\\_NUM\\_THREADS](#) queries the number of host CPU threads OptiX can use for various tasks.

[RT\\_CONTEXT\\_ATTRIBUTE\\_USED\\_HOST\\_MEMORY](#) queries the amount of host memory allocated by OptiX.

[RT\\_CONTEXT\\_ATTRIBUTE\\_GPU\\_PAGING\\_ACTIVE](#) queries if software paging of device memory has been turned on by the context. The returned value is a boolean, where 1 means that paging is currently active.

[RT\\_CONTEXT\\_ATTRIBUTE\\_GPU\\_PAGING\\_FORCED\\_OFF](#) queries if software paging has been prohibited by the user. The returned value is a boolean, where 0 means that OptiX is allowed to activate paging if necessary, 1 means that paging is always off.

[RT\\_CONTEXT\\_ATTRIBUTE\\_AVAILABLE\\_DEVICE\\_MEMORY](#) queries the amount of free device memory.

Some attributes are used to get per device information. In contrast to [rtDeviceGetAttribute](#), these attributes are determined by the context and are therefore queried through the context. This is done by summing the attribute with the OptiX ordinal number when querying the attribute. The following are per device attributes.

#### [RT\\_CONTEXT\\_ATTRIBUTE\\_AVAILABLE\\_DEVICE\\_MEMORY](#)

##### Parameters

in	<i>context</i>	The context object to be queried
in	<i>attrib</i>	Attribute to query
in	<i>size</i>	Size of the attribute being queried. Parameter <i>p</i> must have at least this much memory allocated
out	<i>p</i>	Return pointer where the value of the attribute will be copied into. This must point to at least <i>size</i> bytes of memory

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#) - Can be returned if *size* does not match the proper size of the attribute, if *p* is *NULL*, or if *attribute+ordinal* does not correspond to an OptiX device

##### History

[rtContextGetAttribute](#) was introduced in OptiX 2.0.

**See also** [rtContextGetDeviceCount](#), [rtContextSetAttribute](#), [rtDeviceGetAttribute](#)

#### 3.11.3.52 RTresult RTAPI rtContextGetBufferFromId ( RTcontext context, int buffer\_id, RTbuffer \* buffer )

Gets an RTbuffer corresponding to the buffer id.

##### Description

[rtContextGetBufferFromId](#) returns a handle to the buffer in *\*buffer* corresponding to the *buffer\_id* supplied. If *buffer\_id* does not map to a valid buffer handle, *\*buffer* is *NULL* or if *context* is invalid, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

##### Parameters

in	<i>context</i>	The context the buffer should be originated from
----	----------------	--

in	<i>buffer_id</i>	The ID of the buffer to query
out	<i>buffer</i>	The return handle for the buffer object corresponding to the <i>buffer_id</i>

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtContextGetBufferFromId](#) was introduced in OptiX 3.5.

**See also** [rtBufferGetId](#)

### 3.11.3.53 `RTresult RTAPI rtContextGetDeviceCount ( RTcontext context, unsigned int * count )`

Query the number of devices currently being used.

**Description**

[rtContextGetDeviceCount](#) - Query the number of devices currently being used.

**Parameters**

in	<i>context</i>	The context containing the devices
out	<i>count</i>	Return parameter for the device count

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtContextGetDeviceCount](#) was introduced in OptiX 2.0.

**See also** [rtContextSetDevices](#), [rtContextGetDevices](#)

### 3.11.3.54 `RTresult RTAPI rtContextGetDevices ( RTcontext context, int * devices )`

Retrieve a list of hardware devices being used by the kernel.

**Description**

[rtContextGetDevices](#) retrieves a list of hardware devices used during execution of the subsequent trace kernels.

**Parameters**

in	<i>context</i>	The context to which the hardware list is applied
out	<i>devices</i>	Return parameter for the list of devices. The memory must be able to hold entries numbering least the number of devices as returned by <a href="#">rtContextGetDeviceCount</a>

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextGetDevices](#) was introduced in OptiX 2.0.

**See also** [rtContextSetDevices](#), [rtContextGetDeviceCount](#)

3.11.3.55 `RTresult RTAPI rtContextGetEntryPointCount ( RTcontext context, unsigned int * num_entry_points )`

Query the number of entry points for this context.

## Description

[rtContextGetEntryPointCount](#) passes back the number of entry points associated with this context in *num\_entry\_points*. Returns `RT_ERROR_INVALID_VALUE` if passed a `NULL` pointer.

## Parameters

in	<i>context</i>	The context node to be queried
out	<i>num_entry_points</i>	Return parameter for passing back the entry point count

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextGetEntryPointCount](#) was introduced in OptiX 1.0.

**See also** [rtContextSetEntryPointCount](#)

3.11.3.56 `void RTAPI rtContextGetErrorString ( RTcontext context, RTresult code, const char ** return_string )`

Returns the error string associated with a given error.

## Description

[rtContextGetErrorString](#) return a descriptive string given an error code. If *context* is valid and additional information is available from the last OptiX failure, it will be appended to the generic error code description. *return\_string* will be set to point to this string. The memory *return\_string* points to will be valid until the next API call that returns a string.

## Parameters

in	<i>context</i>	The context object to be queried, or <code>NULL</code>
in	<i>code</i>	The error code to be converted to string
out	<i>return_string</i>	The return parameter for the error string

## Return values

[rtContextGetErrorString](#) does not return a value

## History

[rtContextGetErrorString](#) was introduced in OptiX 1.0.

**See also**

3.11.3.57 `RTresult RTAPI rtContextGetExceptionEnabled ( RTcontext context, RTexception exception, int * enabled )`

Query whether a specified exception is enabled.

## Description

[rtContextGetExceptionEnabled](#) passes back `1` in *\*enabled* if the given exception is enabled, `0` otherwise. *exception* specifies the type of exception to be queried. For a list of available types, see [rtContextSetExceptionEnabled](#). If *exception* is `RT_EXCEPTION_ALL`, *enabled* is set to `1` only if all possible exceptions are enabled.



**Parameters**

in	<i>context</i>	The context to be queried
in	<i>exception</i>	The exception of which to query the state
out	<i>enabled</i>	Return parameter to store whether the exception is enabled

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtContextGetExceptionEnabled](#) was introduced in OptiX 1.1.

**See also** [rtContextSetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintExceptionDetails](#)

**3.11.3.58** `RTresult RTAPI rtContextGetExceptionProgram ( RTcontext context, unsigned int entry_point_index, RTprogram * program )`

Queries the exception program associated with the given context and entry point.

**Description**

[rtContextGetExceptionProgram](#) passes back the exception program associated with the given context and entry point. This program is set via [rtContextSetExceptionProgram](#). Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if given an invalid entry point index or *NULL* pointer.

**Parameters**

in	<i>context</i>	The context node associated with the exception program
in	<i>entry_point_index</i>	The entry point index for the desired exception program
out	<i>program</i>	Return parameter to store the exception program

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtContextGetExceptionProgram](#) was introduced in OptiX 1.0.

**See also** [rtContextSetExceptionProgram](#), [rtContextSetEntryPointCount](#), [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintExceptionDetails](#)

**3.11.3.59** `RTresult RTAPI rtContextGetMissProgram ( RTcontext context, unsigned int ray_type_index, RTprogram * program )`

Queries the miss program associated with the given context and ray type.

**Description**

[rtContextGetMissProgram](#) passes back the miss program associated with the given context and ray type. This program is set via [rtContextSetMissProgram](#). Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if given an invalid ray type index or a *NULL* pointer.

## Parameters

in	<i>context</i>	The context node associated with the miss program
in	<i>ray_type_index</i>	The ray type index for the desired miss program
out	<i>program</i>	Return parameter to store the miss program

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextGetMissProgram](#) was introduced in OptiX 1.0.

**See also** [rtContextSetMissProgram](#), [rtContextGetRayTypeCount](#)

### 3.11.3.60 `RTresult RTAPI rtContextGetPrintBufferSize ( RTcontext context, RTsize * buffer_size_bytes )`

Get the current size of the print buffer.

## Description

[rtContextGetPrintBufferSize](#) is used to query the buffer size available to hold data generated by [rtPrintf](#). Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if passed a `NULL` pointer.

## Parameters

in	<i>context</i>	The context from which to query the print buffer size
out	<i>buffer_size_bytes</i>	The returned print buffer size in bytes

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextGetPrintBufferSize](#) was introduced in OptiX 1.0.

**See also** [rtPrintf](#), [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextGetPrintLaunchIndex](#)

### 3.11.3.61 `RTresult RTAPI rtContextGetPrintEnabled ( RTcontext context, int * enabled )`

Query whether text printing from programs is enabled.

## Description

[rtContextGetPrintEnabled](#) passes back `1` if text printing from programs through [rtPrintf](#) is currently enabled for this context; `0` otherwise. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if passed a `NULL` pointer.

## Parameters

in	<i>context</i>	The context to be queried
out	<i>enabled</i>	Return parameter to store whether printing is enabled

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)

- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtContextGetPrintEnabled](#) was introduced in OptiX 1.0.

**See also** [rtPrintf](#), [rtContextSetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextGetPrintLaunchIndex](#)

#### 3.11.3.62 RTresult RTAPI rtContextGetPrintLaunchIndex ( RTcontext context, int \* x, int \* y, int \* z )

Gets the active print launch index.

### Description

[rtContextGetPrintLaunchIndex](#) is used to query for which launch indices [rtPrintf](#) generates output. The initial value of (x,y,z) is (-1,-1,-1), which generates output for all indices.

#### Parameters

in	context	The context from which to query the print launch index
out	x	Returns the launch index in the x dimension to which the output of <a href="#">rtPrintf</a> invocations is limited. Will not be written to if a <i>NULL</i> pointer is passed
out	y	Returns the launch index in the y dimension to which the output of <a href="#">rtPrintf</a> invocations is limited. Will not be written to if a <i>NULL</i> pointer is passed
out	z	Returns the launch index in the z dimension to which the output of <a href="#">rtPrintf</a> invocations is limited. Will not be written to if a <i>NULL</i> pointer is passed

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtContextGetPrintLaunchIndex](#) was introduced in OptiX 1.0.

**See also** [rtPrintf](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#)

#### 3.11.3.63 RTresult RTAPI rtContextGetProgramFromId ( RTcontext context, int program\_id, RTprogram \* program )

Gets an RTprogram corresponding to the program id.

### Description

[rtContextGetProgramFromId](#) returns a handle to the program in \*program corresponding to the program\_id supplied. If program\_id is not a valid program handle, \*program is set to *NULL*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if context is invalid or program\_id is not a valid program handle.

#### Parameters

in	context	The context the program should be originated from
in	program_id	The ID of the program to query
out	program	The return handle for the program object corresponding to the program_id

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtContextGetProgramFromId](#) was introduced in OptiX 3.6.

**See also** [rtProgramGetId](#)

3.11.3.64 **RTresult** RTAPI `rtContextGetRayGenerationProgram` ( **RTcontext** *context*, unsigned int *entry\_point\_index*, **RTprogram** \* *program* )

Queries the ray generation program associated with the given context and entry point.

#### Description

[rtContextGetRayGenerationProgram](#) passes back the ray generation program associated with the given context and entry point. This program is set via [rtContextSetRayGenerationProgram](#). Returns `RT_ERROR_INVALID_VALUE` if given an invalid entry point index or `NULL` pointer.

#### Parameters

in	<i>context</i>	The context node associated with the ray generation program
in	<i>entry_point_index</i>	The entry point index for the desired ray generation program
out	<i>program</i>	Return parameter to store the ray generation program

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtContextGetRayGenerationProgram](#) was introduced in OptiX 1.0.

**See also** [rtContextSetRayGenerationProgram](#)

3.11.3.65 **RTresult** RTAPI `rtContextGetRayTypeCount` ( **RTcontext** *context*, unsigned int \* *num\_ray\_types* )

Query the number of ray types associated with this context.

#### Description

[rtContextGetRayTypeCount](#) passes back the number of entry points associated with this context in *num\_ray\_types*. Returns `RT_ERROR_INVALID_VALUE` if passed a `NULL` pointer.

#### Parameters

in	<i>context</i>	The context node to be queried
out	<i>num_ray_types</i>	Return parameter to store the number of ray types

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtContextGetRayTypeCount](#) was introduced in OptiX 1.0.

**See also** [rtContextSetRayTypeCount](#)

3.11.3.66 **RTresult** RTAPI `rtContextGetRunningState` ( **RTcontext** *context*, int \* *running* )

Query whether the given context is currently running.

#### Description

This function is currently unimplemented and it is provided as a placeholder for a future implementation.

## Parameters

in	<i>context</i>	The context node to be queried
out	<i>running</i>	Return parameter to store the running state

## Return values

Since unimplemented, this function will always throw an assertion failure.

## History

[rtContextGetRunningState](#) was introduced in OptiX 1.0.

**See also** [rtContextLaunch1D](#), [rtContextLaunch2D](#), [rtContextLaunch3D](#)

## 3.11.3.67 RTresult RTAPI rtContextGetStackSize ( RTcontext context, RTsize \* stack\_size\_bytes )

Query the stack size for this context.

## Description

[rtContextGetStackSize](#) passes back the stack size associated with this context in *stack\_size\_bytes*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if passed a *NULL* pointer.

## Parameters

in	<i>context</i>	The context node to be queried
out	<i>stack_size_bytes</i>	Return parameter to store the size of the stack

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextGetStackSize](#) was introduced in OptiX 1.0.

**See also** [rtContextSetStackSize](#)

## 3.11.3.68 RTresult RTAPI rtContextGetTextureSamplerFromId ( RTcontext context, int sampler\_id, RTtexturesampler \* sampler )

Gets an [RTtexturesampler](#) corresponding to the texture id.

## Description

[rtTextureSamplerGetId](#) returns a handle to the texture sampler in *\*sampler* corresponding to the *sampler\_id* supplied. If *sampler\_id* does not map to a valid texture handle, *\*sampler* is *NULL* or if *context* is invalid, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

in	<i>context</i>	The context the sampler should be originated from
in	<i>sampler_id</i>	The ID of the sampler to query
out	<i>sampler</i>	The return handle for the sampler object corresponding to the <i>sampler_id</i>

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextGetTextureSamplerFromId](#) was introduced in OptiX 3.5.

See also [rtTextureSamplerGetId](#)

3.11.3.69 **RTresult** RTAPI [rtContextGetVariable](#) ( **RTcontext** *context*, unsigned int *index*, **RTvariable** \* *v* )

Queries an indexed variable associated with this context.

#### Description

[rtContextGetVariable](#) queries the variable at position *index* in the variable array from *context* and stores the result in the parameter *v*. A variable must be declared first with [rtContextDeclareVariable](#) and *index* must be in the range  $[0, \text{rtContextGetVariableCount} - 1]$ .

#### Parameters

in	<i>context</i>	The context node to be queried for an indexed variable
in	<i>index</i>	The index that identifies the variable to be queried
out	<i>v</i>	Return value to store the queried variable

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtContextGetVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryGetVariable](#), [rtGeometryInstanceGetVariable](#), [rtMaterialGetVariable](#), [rtProgramGetVariable](#), [rtSelectorGetVariable](#), [rtContextDeclareVariable](#), [rtContextGetVariableCount](#), [rtContextQueryVariable](#), [rtContextRemoveVariable](#)

3.11.3.70 **RTresult** RTAPI [rtContextGetVariableCount](#) ( **RTcontext** *context*, unsigned int \* *count* )

Returns the number of variables associated with this context.

#### Description

[rtContextGetVariableCount](#) returns the number of variables that are currently attached to *context*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if passed a *NULL* pointer.

#### Parameters

in	<i>context</i>	The context to be queried for number of attached variables
out	<i>count</i>	Return parameter to store the number of variables

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtContextGetVariableCount](#) was introduced in OptiX 1.0.

See also [rtGeometryGetVariableCount](#), [rtGeometryInstanceGetVariableCount](#), [rtMaterialGetVariableCount](#), [rtProgramGetVariableCount](#), [rtSelectorGetVariable](#), [rtContextDeclareVariable](#), [rtContextGetVariable](#), [rtContextQueryVariable](#), [rtContextRemoveVariable](#)

3.11.3.71 **RTresult** RTAPI [rtContextLaunch1D](#) ( **RTcontext** *context*, unsigned int *entry\_point\_index*, **RTsize** *image\_width* )

Executes the computation kernel for a given context.

#### Description

`rtContextLaunch` functions execute the computation kernel associated with the given context. If the context has not yet been compiled, or if the context has been modified since the last compile, `rtContextLaunch` will recompile the kernel internally. Acceleration structures of the context which are marked dirty will be updated and their dirty flags will be cleared. Similarly, validation will occur if necessary. The ray generation program specified by `entry_point_index` will be invoked once for every element (pixel or voxel) of the computation grid specified by `image_width`, `image_height`, and `image_depth`.

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_INVALID\\_SOURCE](#)
- [RT\\_ERROR\\_LAUNCH\\_FAILED](#)

### History

`rtContextLaunch` was introduced in OptiX 1.0.

**See also** [rtContextGetRunningState](#), [rtContextCompile](#), [rtContextValidate](#)

#### Parameters

in	<code>context</code>	The context to be executed
in	<code>entry_point_index</code>	The initial entry point into kernel
in	<code>image_width</code>	Width of the computation grid

#### 3.11.3.72 `RTresult RTAPI rtContextLaunch2D ( RTcontext context, unsigned int entry_point_index, RTsize image_width, RTsize image_height )`

#### Parameters

in	<code>context</code>	The context to be executed
in	<code>entry_point_index</code>	The initial entry point into kernel
in	<code>image_width</code>	Width of the computation grid
in	<code>image_height</code>	Height of the computation grid

#### 3.11.3.73 `RTresult RTAPI rtContextLaunch3D ( RTcontext context, unsigned int entry_point_index, RTsize image_width, RTsize image_height, RTsize image_depth )`

#### Parameters

in	<code>context</code>	The context to be executed
in	<code>entry_point_index</code>	The initial entry point into kernel
in	<code>image_width</code>	Width of the computation grid
in	<code>image_height</code>	Height of the computation grid
in	<code>image_depth</code>	Depth of the computation grid

#### 3.11.3.74 `RTresult RTAPI rtContextLaunchProgressive2D ( RTcontext context, unsigned int entry_index, RTsize width, RTsize height, unsigned int max_subframes )`

Executes a Progressive Launch for a given context.

## Description

Starts the (potentially parallel) generation of subframes for progressive rendering. If *max\_subframes* is zero, there is no limit on the number of subframes generated. The generated subframes are automatically composited into a single result and streamed to the client at regular intervals, where they can be read by mapping an associated stream buffer. An application can therefore initiate a progressive launch, and then repeatedly map and display the contents of the stream buffer in order to visualize the progressive refinement of the image.

The call is nonblocking. A polling approach should be used to decide when to map and display the stream buffer contents (see [rtBufferGetProgressiveUpdateReady](#)). If a progressive launch is already in progress at the time of the call and its parameters match the initial launch, the call has no effect. Otherwise, the accumulated result will be reset and a new progressive launch will be started.

If any other OptiX function is called while a progressive launch is in progress, it will cause the launch to stop generating new subframes (however, subframes that have already been generated and are currently in flight may still arrive at the client). The only exceptions to this rule are the operations to map a stream buffer, issuing another progressive launch with unchanged parameters, and polling for an update. Those exceptions do not cause the progressive launch to stop generating subframes.

There is no guarantee that the call actually produces any subframes, especially if [rtContextLaunchProgressive2D](#) and other OptiX commands are called in short succession. For example, during an animation, `rtVariableSet` calls may be tightly interleaved with progressive launches, and when rendering remotely the server may decide to skip some of the launches in order to avoid a large backlog in the command pipeline.

## Parameters

<i>in</i>	<i>context</i>	The context in which the launch is to be executed
<i>in</i>	<i>entry_index</i>	The initial entry point into kernel
<i>in</i>	<i>width</i>	Width of the computation grid
<i>in</i>	<i>height</i>	Height of the computation grid
<i>in</i>	<i>max_subframes</i>	The maximum number of subframes to be generated. Set to zero to generate an unlimited number of subframes

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_LAUNCH\\_FAILED](#)

## History

[rtContextLaunchProgressive2D](#) was introduced in OptiX 3.8.

**See also** [rtContextStopProgressive](#) [rtBufferGetProgressiveUpdateReady](#)

### 3.11.3.75 `RTresult RTAPI rtContextQueryVariable ( RTcontext context, const char * name, RTvariable * v )`

Returns a named variable associated with this context.

## Description

[rtContextQueryVariable](#) queries a variable identified by the string *name* from *context* and stores the result in *\*v*. A variable must be declared with [rtContextDeclareVariable](#) before it can be queried, otherwise *\*v* will be set to `NULL`. [RT\\_ERROR\\_INVALID\\_VALUE](#) will be returned if *name* or *v* is `NULL`.

## Parameters



in	<i>context</i>	The context node to query a variable from
in	<i>name</i>	The name that identifies the variable to be queried
out	<i>v</i>	Return value to store the queried variable

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtContextQueryVariable](#) was introduced in OptiX 1.0.

**See also** [rtGeometryQueryVariable](#), [rtGeometryInstanceQueryVariable](#), [rtMaterialQueryVariable](#), [rtProgramQueryVariable](#), [rtSelectorQueryVariable](#), [rtContextDeclareVariable](#), [rtContextGetVariableCount](#), [rtContextGetVariable](#), [rtContextRemoveVariable](#)

### 3.11.3.76 RTresult RTAPI rtContextRemoveVariable ( RTcontext *context*, RTvariable *v* )

Removes a variable from the given context.

**Description**

[rtContextRemoveVariable](#) removes variable *v* from *context* if present. Returns [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#) if the variable is not attached to this context. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if passed an invalid variable.

**Parameters**

in	<i>context</i>	The context node from which to remove a variable
in	<i>v</i>	The variable to be removed

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

**History**

[rtContextRemoveVariable](#) was introduced in OptiX 1.0.

**See also** [rtGeometryRemoveVariable](#), [rtGeometryInstanceRemoveVariable](#), [rtMaterialRemoveVariable](#), [rtProgramRemoveVariable](#), [rtSelectorRemoveVariable](#), [rtContextDeclareVariable](#), [rtContextGetVariable](#), [rtContextGetVariableCount](#), [rtContextQueryVariable](#),

### 3.11.3.77 RTresult RTAPI rtContextSetAttribute ( RTcontext *context*, RTcontextattribute *attrib*, RTsize *size*, void \* *p* )

Set an attribute specific to an OptiX context.

**Description**

[rtContextSetAttribute](#) sets *p* as the value of the per context attribute specified by *attrib*.

Each attribute can have a different size. The sizes are given in the following list:

- [RT\\_CONTEXT\\_ATTRIBUTE\\_CPU\\_NUM\\_THREADS](#) sizeof(int)
- [RT\\_CONTEXT\\_ATTRIBUTE\\_GPU\\_PAGING\\_FORCED\\_OFF](#) sizeof(int)

[RT\\_CONTEXT\\_ATTRIBUTE\\_CPU\\_NUM\\_THREADS](#) sets the number of host CPU threads OptiX can use for various tasks.

[RT\\_CONTEXT\\_ATTRIBUTE\\_GPU\\_PAGING\\_FORCED\\_OFF](#) prohibits software paging of device memory. A value of 0 means that OptiX is allowed to activate paging if necessary, 1 means that paging is always off. Note that currently paging cannot be disabled once it has been activated.

#### Parameters

in	<i>context</i>	The context object to be modified
in	<i>attrib</i>	Attribute to set
in	<i>size</i>	Size of the attribute being set
in	<i>p</i>	Pointer to where the value of the attribute will be copied from. This must point to at least <i>size</i> bytes of memory

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#) - Can be returned if *size* does not match the proper size of the attribute, or if *p* is `NULL`

#### History

[rtContextSetAttribute](#) was introduced in OptiX 2.5.

See also [rtContextGetAttribute](#)

#### 3.11.3.78 `RTresult RTAPI rtContextSetDevices ( RTcontext context, unsigned int count, const int * devices )`

Specify a list of hardware devices to be used by the kernel.

#### Description

[rtContextSetDevices](#) specifies a list of hardware devices to be used during execution of the subsequent trace kernels.

#### Parameters

in	<i>context</i>	The context to which the hardware list is applied
in	<i>count</i>	The number of devices in the list
in	<i>devices</i>	The list of devices

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_NO\\_DEVICE](#)
- [RT\\_ERROR\\_INVALID\\_DEVICE](#)

#### History

[rtContextSetDevices](#) was introduced in OptiX 1.0.

See also [rtContextGetDevices](#), [rtContextGetDeviceCount](#)

#### 3.11.3.79 `RTresult RTAPI rtContextSetEntryPointCount ( RTcontext context, unsigned int num_entry_points )`

Set the number of entry points for a given context.

#### Description

[rtContextSetEntryPointCount](#) sets the number of entry points associated with the given context to *num\_entry\_points*.

## Parameters

<code>in</code>	<code>context</code>	The context to be modified
<code>in</code>	<code>num_entry_points</code>	The number of entry points to use

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextSetEntryPointCount](#) was introduced in OptiX 1.0.

See also [rtContextGetEntryPointCount](#)

### 3.11.3.80 `RTresult RTAPI rtContextSetExceptionEnabled ( RTcontext context, RTexception exception, int enabled )`

Enable or disable an exception.

## Description

[rtContextSetExceptionEnabled](#) is used to enable or disable specific exceptions. If an exception is enabled, the exception condition is checked for at runtime, and the exception program is invoked if the condition is met. The exception program can query the type of the caught exception by calling [rtGetExceptionCode](#). *exception* may take one of the following values:

- [RT\\_EXCEPTION\\_TEXTURE\\_ID\\_INVALID](#)
- [RT\\_EXCEPTION\\_BUFFER\\_ID\\_INVALID](#)
- [RT\\_EXCEPTION\\_INDEX\\_OUT\\_OF\\_BOUNDS](#)
- [RT\\_EXCEPTION\\_STACK\\_OVERFLOW](#)
- [RT\\_EXCEPTION\\_BUFFER\\_INDEX\\_OUT\\_OF\\_BOUNDS](#)
- [RT\\_EXCEPTION\\_INVALID\\_RAY](#)
- [RT\\_EXCEPTION\\_INTERNAL\\_ERROR](#)
- [RT\\_EXCEPTION\\_USER](#)
- [RT\\_EXCEPTION\\_ALL](#)

[RT\\_EXCEPTION\\_TEXTURE\\_ID\\_INVALID](#) verifies that every access of a texture id is valid, including use of `RT_TEXTURE_ID_NULL` and IDs out of bounds.

[RT\\_EXCEPTION\\_BUFFER\\_ID\\_INVALID](#) verifies that every access of a buffer id is valid, including use of `RT_BUFFER_ID_NULL` and IDs out of bounds.

[RT\\_EXCEPTION\\_INDEX\\_OUT\\_OF\\_BOUNDS](#) checks that [rtIntersectChild](#) and [rtReportIntersection](#) are called with a valid index.

[RT\\_EXCEPTION\\_STACK\\_OVERFLOW](#) checks the runtime stack against overflow. The most common cause for an overflow is a too deep [rtTrace](#) recursion tree.

[RT\\_EXCEPTION\\_BUFFER\\_INDEX\\_OUT\\_OF\\_BOUNDS](#) checks every read and write access to [rtBuffer](#) objects to be within valid bounds.

[RT\\_EXCEPTION\\_INVALID\\_RAY](#) checks the each ray's origin and direction values against *NaNs* and *infinity* values.

[RT\\_EXCEPTION\\_INTERNAL\\_ERROR](#) indicates an unexpected internal error in the runtime.

`RT_EXCEPTION_USER` is used to enable or disable all user-defined exceptions. The reserved range of exception codes for user-defined exceptions starts at `RT_EXCEPTION_USER` (`0x400`) and ends at `0xFFFF`. See `rtThrow` for more information.

`RT_EXCEPTION_ALL` is a placeholder value which can be used to enable or disable all possible exceptions with a single call to `rtContextSetExceptionEnabled`.

By default, `RT_EXCEPTION_STACK_OVERFLOW` is enabled and all other exceptions are disabled.

#### Parameters

in	<i>context</i>	The context for which the exception is to be enabled or disabled
in	<i>exception</i>	The exception which is to be enabled or disabled
in	<i>enabled</i>	Nonzero to enable the exception, <code>0</code> to disable the exception

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

#### History

`rtContextSetExceptionEnabled` was introduced in OptiX 1.1.

**See also** `rtContextGetExceptionEnabled`, `rtContextSetExceptionProgram`, `rtContextGetExceptionProgram`, `rtGetExceptionCode`, `rtThrow`, `rtPrintExceptionDetails`

**3.11.3.81** `RTresult RTAPI rtContextSetExceptionProgram ( RTcontext context, unsigned int entry_point_index, RTprogram program )`

Specifies the exception program for a given context entry point.

#### Description

`rtContextSetExceptionProgram` sets *context's* exception program at entry point *entry\_point\_index*. `RT_ERROR_INVALID_VALUE` is returned if *entry\_point\_index* is outside of the range `[0, rtContextGetEntryPointCount - 1]`.

#### Parameters

in	<i>context</i>	The context node to which the exception program will be added
in	<i>entry_point_index</i>	The entry point the program will be associated with
in	<i>program</i>	The exception program

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_TYPE_MISMATCH`

#### History

`rtContextSetExceptionProgram` was introduced in OptiX 1.0.

**See also** `rtContextGetEntryPointCount`, `rtContextGetExceptionProgram`, `rtContextSetExceptionEnabled`, `rtContextGetExceptionEnabled`, `rtGetExceptionCode`, `rtThrow`, `rtPrintExceptionDetails`

**3.11.3.82** `RTresult RTAPI rtContextSetMissProgram ( RTcontext context, unsigned int ray_type_index, RTprogram program )`

Specifies the miss program for a given context ray type.

### Description

[`rtContextSetMissProgram`](#) sets *context*'s miss program associated with ray type *ray\_type\_index*. [`RT\_ERROR\_INVALID\_VALUE`](#) is returned if *ray\_type\_index* is outside of the range `[0, rtContextGetRayTypeCount - 1]`.

#### Parameters

in	<i>context</i>	The context node to which the miss program will be added
in	<i>ray_type_index</i>	The ray type the program will be associated with
in	<i>program</i>	The miss program

### Return values

Relevant return values:

- [`RT\_SUCCESS`](#)
- [`RT\_ERROR\_INVALID\_VALUE`](#)
- [`RT\_ERROR\_MEMORY\_ALLOCATION\_FAILED`](#)
- [`RT\_ERROR\_TYPE\_MISMATCH`](#)

### History

[`rtContextSetMissProgram`](#) was introduced in OptiX 1.0.

**See also** [`rtContextGetRayTypeCount`](#), [`rtContextGetMissProgram`](#)

#### 3.11.3.83 `RTresult RTAPI rtContextSetPrintBufferSize ( RTcontext context, RTsize buffer_size_bytes )`

Set the size of the print buffer.

### Description

[`rtContextSetPrintBufferSize`](#) is used to set the buffer size available to hold data generated by [`rtPrintf`](#). The default size is 65536 bytes.

#### Parameters

in	<i>context</i>	The context for which to set the print buffer size
in	<i>buffer_size_ - bytes</i>	The print buffer size in bytes

### Return values

Relevant return values:

- [`RT\_SUCCESS`](#)
- [`RT\_ERROR\_INVALID\_VALUE`](#)

### History

[`rtContextSetPrintBufferSize`](#) was introduced in OptiX 1.0.

**See also** [`rtPrintf`](#), [`rtContextSetPrintEnabled`](#), [`rtContextGetPrintEnabled`](#), [`rtContextGetPrintBufferSize`](#), [`rtContextSetPrintLaunchIndex`](#), [`rtContextGetPrintLaunchIndex`](#)

#### 3.11.3.84 `RTresult RTAPI rtContextSetPrintEnabled ( RTcontext context, int enabled )`

Enable or disable text printing from programs.

### Description

[`rtContextSetPrintEnabled`](#) is used to control whether text printing in programs through [`rtPrintf`](#) is currently enabled for this context.

## Parameters

<code>in</code>	<code>context</code>	The context for which printing is to be enabled or disabled
<code>in</code>	<code>enabled</code>	Setting this parameter to a nonzero value enables printing, <code>0</code> disables printing

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextSetPrintEnabled](#) was introduced in OptiX 1.0.

**See also** [rtPrintf](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextGetPrintLaunchIndex](#)

### 3.11.3.85 RTresult RTAPI rtContextSetPrintLaunchIndex ( RTcontext *context*, int *x*, int *y*, int *z* )

Sets the active launch index to limit text output.

## Description

[rtContextSetPrintLaunchIndex](#) is used to control for which launch indices [rtPrintf](#) generates output. The initial value of (x,y,z) is (-1,-1,-1), which generates output for all indices.

## Parameters

<code>in</code>	<code>context</code>	The context for which to set the print launch index
<code>in</code>	<code>x</code>	The launch index in the x dimension to which to limit the output of <a href="#">rtPrintf</a> invocations. If set to <code>-1</code> , output is generated for all launch indices in the x dimension
<code>in</code>	<code>y</code>	The launch index in the y dimension to which to limit the output of <a href="#">rtPrintf</a> invocations. If set to <code>-1</code> , output is generated for all launch indices in the y dimension
<code>in</code>	<code>z</code>	The launch index in the z dimension to which to limit the output of <a href="#">rtPrintf</a> invocations. If set to <code>-1</code> , output is generated for all launch indices in the z dimension

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextSetPrintLaunchIndex](#) was introduced in OptiX 1.0.

**See also** [rtPrintf](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextGetPrintLaunchIndex](#)

### 3.11.3.86 RTresult RTAPI rtContextSetRayGenerationProgram ( RTcontext *context*, unsigned int *entry\_point\_index*, RTprogram *program* )

Specifies the ray generation program for a given context entry point.

## Description

[rtContextSetRayGenerationProgram](#) sets *context*'s ray generation program at entry point *entry\_point\_index*. [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned if *entry\_point\_index* is outside of the range `[0, rtContextGetEntryPointCount - 1]`.

**Parameters**

in	<i>context</i>	The context node to which the exception program will be added
in	<i>entry_point_index</i>	The entry point the program will be associated with
in	<i>program</i>	The ray generation program

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

**History**

[rtContextSetRayGenerationProgram](#) was introduced in OptiX 1.0.

**See also** [rtContextGetEntryPointCount](#), [rtContextGetRayGenerationProgram](#)

3.11.3.87 **RTresult** RTAPI [rtContextSetRayTypeCount](#) ( **RTcontext** *context*, unsigned int *num\_ray\_types* )

Sets the number of ray types for a given context.

**Description**

[rtContextSetRayTypeCount](#) Sets the number of ray types associated with the given context.

**Parameters**

in	<i>context</i>	The context node
in	<i>num_ray_types</i>	The number of ray types to be used

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtContextSetRayTypeCount](#) was introduced in OptiX 1.0.

**See also** [rtContextGetRayTypeCount](#)

3.11.3.88 **RTresult** RTAPI [rtContextSetRemoteDevice](#) ( **RTcontext** *context*, **RTremotedevice** *remote\_dev* )

Enable rendering on a remote device.

**Description**

Associates a context with a remote device. If successful, any further OptiX calls will be directed to the remote device and executed there. The context must be an empty, newly created context. In other words, in order to use a context remotely, the call to [rtContextSetRemoteDevice](#) should immediately follow the call to [rtContextCreate](#).

Note that a context that was used for remote rendering cannot be re-used for local rendering by changing devices. However, the Progressive API (that is, [rtContextLaunchProgressive2D](#), stream buffers, etc.) can be used locally by simply not creating a remote device and not calling [rtContextSetRemoteDevice](#).

Only a single remote device can be associated with a context. Switching between different remote devices is not supported.

## Parameters

<code>in</code>	<code>context</code>	Newly created context to use on the remote device
<code>in</code>	<code>remote_dev</code>	Remote device on which rendering is to be executed

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextSetRemoteDevice](#) was introduced in OptiX 3.8.

**See also** [rtRemoteDeviceCreate](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceReserve](#) [rtContextLaunchProgressive2D](#)

### 3.11.3.89 `RTresult RTAPI rtContextSetStackSize ( RTcontext context, RTsize stack_size_bytes )`

Set the stack size for a given context.

## Description

[rtContextSetStackSize](#) sets the stack size for the given context to `stack_size_bytes` bytes. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if context is not valid.

## Parameters

<code>in</code>	<code>context</code>	The context node to be modified
<code>in</code>	<code>stack_size_bytes</code>	The desired stack size in bytes

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtContextSetStackSize](#) was introduced in OptiX 1.0.

**See also** [rtContextGetStackSize](#)

### 3.11.3.90 `RTresult RTAPI rtContextSetTimeoutCallback ( RTcontext context, RTtimeoutcallback callback, double min_polling_seconds )`

Side timeout callback function.

## Description

[rtContextSetTimeoutCallback](#) sets an application-side callback function `callback` and a time interval `min_polling_seconds` in seconds. Long-running OptiX API calls such as [rtContextCompile](#) and [rtContextLaunch](#) call the callback function about every `min_polling_seconds` seconds. The core purpose of a timeout callback function is to give the application a chance to do whatever it might need to do frequently, such as handling GUI events.

If the callback function returns true, the API call tries to abort, leaving the context in a clean but unfinished state. Output buffers are left in an unpredictable state. In case an OptiX API call is terminated by a callback function, it returns [RT\\_TIMEOUT\\_CALLBACK](#).

As a side effect, timeout functions also help control the OptiX kernel run-time. This can in some cases prevent OptiX kernel launches from running so long that they cause driver timeouts. For example, if `min_polling_seconds` is 0.5 seconds then once the kernel has been running for 0.5 seconds it won't start any new launch indices (calls to a ray generation program). Thus, if the driver's timeout is 2 seconds (the default on Windows), then a launch index may take up to 1.5 seconds without triggering a driver timeout.



[RTtimeoutcallback](#) is defined as `int (*RTtimeoutcallback)(void)`.

To unregister a callback function, `callback` needs to be set to `NULL` and `min_polling_seconds` to 0.

Only one timeout callback function can be specified at any time.

Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if `context` is not valid, if `min_polling_seconds` is negative, if `callback` is `NULL` but `min_polling_seconds` is not 0, or if `callback` is not `NULL` but `min_polling_seconds` is 0.

#### Parameters

in	<code>context</code>	The context node to be modified
in	<code>callback</code>	The function to be called
in	<code>min_polling_seconds</code>	The timeout interval after which the function is called

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtContextSetTimeoutCallback](#) was introduced in OptiX 2.5.

See also [rtContextCompile](#), [rtContextLaunch](#)

#### 3.11.3.91 `RTresult RTAPI rtContextStopProgressive ( RTcontext context )`

Stops a Progressive Launch.

#### Description

If a progressive launch is currently in progress, calling [rtContextStopProgressive](#) terminates it. Otherwise, the call has no effect. If a launch is stopped using this function, no further subframes will arrive at the client, even if they have already been generated by the server and are currently in flight.

This call should only be used if the application must guarantee that frames generated by previous progressive launches won't be accessed. Do not call [rtContextStopProgressive](#) in the main rendering loop if the goal is only to change OptiX state (e.g. `rtVariable` values). The call is unnecessary in that case and will degrade performance.

#### Parameters

in	<code>context</code>	The context associated with the progressive launch
----	----------------------	--

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

#### History

[rtContextStopProgressive](#) was introduced in OptiX 3.8.

See also [rtContextLaunchProgressive2D](#)

#### 3.11.3.92 `RTresult RTAPI rtContextValidate ( RTcontext context )`

Checks the given context for valid internal state.

#### Description

[`rtContextValidate`](#) checks the the given context and all of its associated OptiX objects for a valid state. These checks include tests for presence of necessary programs (e.g. an intersection program for a geometry node), invalid internal state such as `NULL` children in graph nodes, and presence of variables required by all specified programs. [`rtContextGetErrorString`](#) can be used to retrieve a description of a validation failure.

#### Parameters

<code>in</code>	<code>context</code>	The context to be validated
-----------------	----------------------	-----------------------------

#### Return values

Relevant return values:

- [`RT\_SUCCESS`](#)
- [`RT\_ERROR\_INVALID\_CONTEXT`](#)
- [`RT\_ERROR\_INVALID\_VALUE`](#)
- [`RT\_ERROR\_INVALID\_SOURCE`](#)

#### History

[`rtContextValidate`](#) was introduced in OptiX 1.0.

**See also** [`rtContextGetErrorString`](#)

#### 3.11.3.93 `RTresult RTAPI rtDeviceGetAttribute ( int ordinal, RTdeviceattribute attrib, RTsize size, void * p )`

Returns an attribute specific to an OptiX device.

#### Description

[`rtDeviceGetAttribute`](#) returns in `p` the value of the per device attribute specified by `attrib` for device `ordinal`.

Each attribute can have a different size. The sizes are given in the following list:

- [`RT\_DEVICE\_ATTRIBUTE\_MAX\_THREADS\_PER\_BLOCK`](#) `sizeof(int)`
- [`RT\_DEVICE\_ATTRIBUTE\_CLOCK\_RATE`](#) `sizeof(int)`
- [`RT\_DEVICE\_ATTRIBUTE\_MULTIPROCESSOR\_COUNT`](#) `sizeof(int)`
- [`RT\_DEVICE\_ATTRIBUTE\_EXECUTION\_TIMEOUT\_ENABLED`](#) `sizeof(int)`
- [`RT\_DEVICE\_ATTRIBUTE\_MAX\_HARDWARE\_TEXTURE\_COUNT`](#) `sizeof(int)`
- [`RT\_DEVICE\_ATTRIBUTE\_NAME`](#) up to `size-1`
- [`RT\_DEVICE\_ATTRIBUTE\_COMPUTE\_CAPABILITY`](#) `sizeof(int2)`
- [`RT\_DEVICE\_ATTRIBUTE\_TOTAL\_MEMORY`](#) `sizeof(RTsize)`
- [`RT\_DEVICE\_ATTRIBUTE\_TCC\_DRIVER`](#) `sizeof(int)`
- [`RT\_DEVICE\_ATTRIBUTE\_CUDA\_DEVICE\_ORDINAL`](#) `sizeof(int)`

#### Parameters

<code>in</code>	<code>ordinal</code>	OptiX device ordinal
<code>in</code>	<code>attrib</code>	Attribute to query
<code>in</code>	<code>size</code>	Size of the attribute being queried. Parameter <code>p</code> must have at least this much memory allocated

out	<i>p</i>	Return pointer where the value of the attribute will be copied into. This must point to at least <i>size</i> bytes of memory
-----	----------	--

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#) - Can be returned if size does not match the proper size of the attribute, if *p* is *NULL*, or if *ordinal* does not correspond to an OptiX device

**History**

[rtDeviceGetAttribute](#) was introduced in OptiX 2.0. [RT\\_DEVICE\\_ATTRIBUTE\\_TCC\\_DRIVER](#) was introduced in OptiX 3.0. [RT\\_DEVICE\\_ATTRIBUTE\\_CUDA\\_DEVICE\\_ORDINAL](#) was introduced in OptiX 3.0.

**See also** [rtDeviceGetDeviceCount](#), [rtContextGetAttribute](#)

### 3.11.3.94 RTresult RTAPI rtDeviceGetDeviceCount ( unsigned int \* *count* )

Returns the number of OptiX capable devices.

**Description**

[rtDeviceGetDeviceCount](#) returns in *count* the number of compute devices that are available in the host system and will be used by OptiX.

**Parameters**

out	<i>count</i>	Number devices available for OptiX
-----	--------------	------------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtDeviceGetDeviceCount](#) was introduced in OptiX 1.0.

**See also** [rtGetVersion](#)

### 3.11.3.95 RTresult RTAPI rtGeometryCreate ( RTcontext *context*, RTgeometry \* *geometry* )

Creates a new geometry node.

**Description**

[rtGeometryCreate](#) creates a new geometry node within a context. *context* specifies the target context, and should be a value returned by [rtContextCreate](#). Sets \**geometry* to the handle of a newly created geometry within *context*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *geometry* is *NULL*.

**Parameters**

in	<i>context</i>	Specifies the rendering context of the Geometry node
out	<i>geometry</i>	New Geometry node handle

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryCreate](#) was introduced in OptiX 1.0.

**See also** [rtGeometryDestroy](#), [rtGeometrySetBoundingBoxProgram](#), [rtGeometrySetIntersectionProgram](#)

3.11.3.96 **RTresult** RTAPI [rtGeometryDeclareVariable](#) ( **RTgeometry** *geometry*, **const char \*** *name*, **RTvariable \*** *v* )

Declares a new named variable associated with a geometry instance.

## Description

[rtGeometryDeclareVariable](#) declares a new variable associated with a geometry node. *geometry* specifies the target geometry node, and should be a value returned by [rtGeometryCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If there is currently no variable associated with *geometry* named *name*, a new variable named *name* will be created and associated with *geometry*. Returns the handle of the newly-created variable in *\*v* or *NULL* otherwise. After declaration, the variable can be queried with [rtGeometryQueryVariable](#) or [rtGeometryGetVariable](#). A declared variable does not have a type until its value is set with one of the [rtVariableSet](#) functions. Once a variable is set, its type cannot be changed anymore.

## Parameters

in	<i>geometry</i>	Specifies the associated Geometry node
in	<i>name</i>	The name that identifies the variable
out	<i>v</i>	Returns a handle to a newly declared variable

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_REDECLARED](#)
- [RT\\_ERROR\\_ILLEGAL\\_SYMBOL](#)

## History

[rtGeometryDeclareVariable](#) was introduced in OptiX 1.0.

**See also** Variables, [rtGeometryQueryVariable](#), [rtGeometryGetVariable](#), [rtGeometryRemoveVariable](#)

3.11.3.97 **RTresult** RTAPI [rtGeometryDestroy](#) ( **RTgeometry** *geometry* )

Destroys a geometry node.

## Description

[rtGeometryDestroy](#) removes *geometry* from its context and deletes it. *geometry* should be a value returned by [rtGeometryCreate](#). Associated variables declared via [rtGeometryDeclareVariable](#) are destroyed, but no child graph nodes are destroyed. After the call, *geometry* is no longer a valid handle.

## Parameters

in	<i>geometry</i>	Handle of the geometry node to destroy
----	-----------------	--

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)

- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtGeometryDestroy](#) was introduced in OptiX 1.0.

**See also** [rtGeometryCreate](#), [rtGeometrySetPrimitiveCount](#), [rtGeometryGetPrimitiveCount](#)

**3.11.3.98** **RTresult** RTAPI [rtGeometryGetBoundingBoxProgram](#) ( **RTgeometry** *geometry*, **RTprogram** \* *program* )

Returns the attached bounding box program.

### Description

[rtGeometryGetBoundingBoxProgram](#) returns the handle *program* for the attached bounding box program of *geometry*.

#### Parameters

in	<i>geometry</i>	Geometry node handle from which to query program
out	<i>program</i>	Handle to attached bounding box program

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtGeometryGetBoundingBoxProgram](#) was introduced in OptiX 1.0.

**See also** [rtGeometrySetBoundingBoxProgram](#)

**3.11.3.99** **RTresult** RTAPI [rtGeometryGetContext](#) ( **RTgeometry** *geometry*, **RTcontext** \* *context* )

Returns the context associated with a geometry node.

### Description

[rtGeometryGetContext](#) queries a geometry node for its associated context. *geometry* specifies the geometry node to query, and should be a value returned by [rtGeometryCreate](#). Sets \**context* to the context associated with *geometry*.

#### Parameters

in	<i>geometry</i>	Specifies the geometry to query
out	<i>context</i>	The context associated with <i>geometry</i>

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGetContext](#) was introduced in OptiX 1.0.

**See also** [rtGeometryCreate](#)

3.11.3.100 **RTresult** RTAPI `rtGeometryGetIntersectionProgram ( RTgeometry geometry, RTprogram * program )`

Returns the attached intersection program.

## Description

[rtGeometryGetIntersectionProgram](#) returns in *program* a handle of the attached intersection program.

## Parameters

in	<i>geometry</i>	Geometry node handle to query program
out	<i>program</i>	Handle to attached intersection program

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGetIntersectionProgram](#) was introduced in OptiX 1.0.

**See also** [rtGeometrySetIntersectionProgram](#), [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.11.3.101 **RTresult** RTAPI `rtGeometryGetPrimitiveCount ( RTgeometry geometry, unsigned int * num_primitives )`

Returns the number of primitives.

## Description

[rtGeometryGetPrimitiveCount](#) returns for *geometry* the number of set primitives. The number of primitives can be set with [rtGeometrySetPrimitiveCount](#).

## Parameters

in	<i>geometry</i>	Geometry node to query from the number of primitives
out	<i>num_primitives</i>	Number of primitives

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGetPrimitiveCount](#) was introduced in OptiX 1.0.

**See also** [rtGeometrySetPrimitiveCount](#)

3.11.3.102 `RTresult RTAPI rtGeometryGetPrimitiveIndexOffset ( RTgeometry geometry, unsigned int * index_offset )`

Returns the current primitive index offset.

**Description**

[rtGeometryGetPrimitiveIndexOffset](#) returns for *geometry* the primitive index offset. The primitive index offset can be set with [rtGeometrySetPrimitiveIndexOffset](#).

**Parameters**

in	<i>geometry</i>	Geometry node to query for the primitive index offset
out	<i>index_offset</i>	Primitive index offset

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtGeometryGetPrimitiveIndexOffset](#) was introduced in OptiX 3.5.

**See also** [rtGeometrySetPrimitiveIndexOffset](#)

3.11.3.103 `RTresult RTAPI rtGeometryGetVariable ( RTgeometry geometry, unsigned int index, RTvariable * v )`

Returns a handle to an indexed variable of a geometry node.

**Description**

[rtGeometryGetVariable](#) queries the handle of a geometry node's indexed variable. *geometry* specifies the target geometry and should be a value returned by [rtGeometryCreate](#). *index* specifies the index of the variable, and should be a value less than [rtGeometryGetVariableCount](#). If *index* is the index of a variable attached to *geometry*, returns its handle in \**v* or *NULL* otherwise. \**v* must be declared first with [rtGeometryDeclareVariable](#) before it can be queried.

**Parameters**

in	<i>geometry</i>	The geometry node from which to query a variable
in	<i>index</i>	The index that identifies the variable to be queried
out	<i>v</i>	Returns handle to indexed variable

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

**History**

[rtGeometryGetVariable](#) was introduced in OptiX 1.0.

**See also** [rtGeometryDeclareVariable](#), [rtGeometryGetVariableCount](#), [rtGeometryRemoveVariable](#), [rtGeometryQueryVariable](#)

**3.11.3.104** `RTresult RTAPI rtGeometryGetVariableCount ( RTgeometry geometry, unsigned int * count )`

Returns the number of attached variables.

**Description**

`rtGeometryGetVariableCount` queries the number of variables attached to a geometry node. *geometry* specifies the geometry node, and should be a value returned by `rtGeometryCreate`. After the call, the number of variables attached to *geometry* is returned to *\*count*.

**Parameters**

in	<i>geometry</i>	The Geometry node to query from the number of attached variables
out	<i>count</i>	Returns the number of attached variables

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

`rtGeometryGetVariableCount` was introduced in OptiX 1.0.

**See also** [rtGeometryGetVariableCount](#), [rtGeometryDeclareVariable](#), [rtGeometryRemoveVariable](#)

**3.11.3.105** `RTresult RTAPI rtGeometryGroupCreate ( RTcontext context, RTgeometrygroup * geometrygroup )`

Creates a new geometry group.

**Description**

`rtGeometryGroupCreate` creates a new geometry group within a context. *context* specifies the target context, and should be a value returned by `rtContextCreate`. Sets *\*geometrygroup* to the handle of a newly created geometry group within *context*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *geometrygroup* is *NULL*.

**Parameters**

in	<i>context</i>	Specifies a context within which to create a new geometry group
out	<i>geometrygroup</i>	Returns a newly created geometry group

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

`rtGeometryGroupCreate` was introduced in OptiX 1.0.

**See also** [rtGeometryGroupDestroy](#), [rtContextCreate](#)

**3.11.3.106** `RTresult RTAPI rtGeometryGroupDestroy ( RTgeometrygroup geometrygroup )`

Destroys a geometry group node.



### Description

[rtGeometryGroupDestroy](#) removes *geometrygroup* from its context and deletes it. *geometrygroup* should be a value returned by [rtGeometryGroupCreate](#). No child graph nodes are destroyed. After the call, *geometrygroup* is no longer a valid handle.

#### Parameters

in	<i>geometrygroup</i>	Handle of the geometry group node to destroy
----	----------------------	--

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtGeometryGroupDestroy](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGroupCreate](#)

3.11.3.107 **RTresult** RTAPI [rtGeometryGroupGetAcceleration](#) ( **RTgeometrygroup** *geometrygroup*, **RTacceleration** \* *acceleration* )

Returns the acceleration structure attached to a geometry group.

### Description

[rtGeometryGroupGetAcceleration](#) returns the acceleration structure attached to a geometry group using [rtGeometryGroupSetAcceleration](#). If no acceleration structure has previously been set, \**acceleration* is set to *NULL*.

#### Parameters

in	<i>geometrygroup</i>	The geometry group handle
out	<i>acceleration</i>	The returned acceleration structure object

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtGeometryGroupGetAcceleration](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGroupSetAcceleration](#), [rtAccelerationCreate](#)

3.11.3.108 **RTresult** RTAPI [rtGeometryGroupGetChild](#) ( **RTgeometrygroup** *geometrygroup*, unsigned int *index*, **RTgeometryinstance** \* *geometryinstance* )

Returns a child node of a geometry group.

### Description

[rtGeometryGroupGetChild](#) returns the child geometry instance at slot *index* of the parent *geometrygroup*. If no child has been assigned to the given slot, \**geometryinstance* is set to *NULL*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if given an invalid child index or *NULL* pointer.

## Parameters

in	<i>geometrygroup</i>	The parent geometry group handle
in	<i>index</i>	The index of the child slot to query
out	<i>geometryinstance</i>	The returned child geometry instance

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGroupGetChild](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGroupSetChild](#), [rtGeometryGroupSetChildCount](#), [rtGeometryGroupGetChildCount](#),

**3.11.3.109** `RTresult RTAPI rtGeometryGroupGetChildCount ( RTgeometrygroup geometrygroup, unsigned int * count )`

Returns the number of child slots for a group.

## Description

[rtGeometryGroupGetChildCount](#) returns the number of child slots allocated using [rtGeometryGroupSetChildCount](#). This includes empty slots which may not yet have actual children assigned by [rtGeometryGroupSetChild](#).

## Parameters

in	<i>geometrygroup</i>	The parent geometry group handle
out	<i>count</i>	Returned number of child slots

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGroupGetChildCount](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGroupSetChild](#), [rtGeometryGroupGetChild](#), [rtGeometryGroupSetChildCount](#)

**3.11.3.110** `RTresult RTAPI rtGeometryGroupGetContext ( RTgeometrygroup geometrygroup, RTcontext * context )`

Returns the context associated with a geometry group.

## Description

[rtGeometryGroupGetContext](#) queries a geometry group for its associated context. *geometrygroup* specifies the geometry group to query, and must be a value returned by [rtGeometryGroupCreate](#). Sets \**context* to the context associated with *geometrygroup*.

## Parameters

in	<i>geometrygroup</i>	Specifies the geometry group to query
out	<i>context</i>	Returns the context associated with the geometry group

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGroupGetContext](#) was introduced in OptiX 1.0.

**See also** [rtContextCreate](#), [rtGeometryGroupCreate](#)

### 3.11.3.111 `RTresult RTAPI rtGeometryGroupSetAcceleration ( RTgeometrygroup geometrygroup, RTacceleration acceleration )`

Set the acceleration structure for a group.

## Description

[rtGeometryGroupSetAcceleration](#) attaches an acceleration structure to a geometry group. The acceleration structure must have been previously created using [rtAccelerationCreate](#). Every geometry group is required to have an acceleration structure assigned in order to pass validation. The acceleration structure will be built over the primitives contained in all children of the geometry group. This enables a single acceleration structure to be built over primitives of multiple geometry instances. Note that it is legal to attach a single `RTacceleration` object to multiple geometry groups, as long as the underlying geometry of all children is the same. This corresponds to attaching an acceleration structure to multiple groups at higher graph levels using [rtGroupSetAcceleration](#).

## Parameters

in	<i>geometrygroup</i>	The geometry group handle
in	<i>acceleration</i>	The acceleration structure to attach to the geometry group

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGroupSetAcceleration](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGroupGetAcceleration](#), [rtAccelerationCreate](#), [rtGroupSetAcceleration](#)

### 3.11.3.112 `RTresult RTAPI rtGeometryGroupSetChild ( RTgeometrygroup geometrygroup, unsigned int index, RTgeometryinstance geometryinstance )`

Attaches a child node to a geometry group.

## Description

`rtGeometryGroupSetChild` attaches a new child node *geometryinstance* to the parent node *geometrygroup*. *index* specifies the number of the slot where the child node gets attached. The index value must be lower than the number previously set by `rtGeometryGroupSetChildCount`.

## Parameters

in	<i>geometrygroup</i>	The parent geometry group handle
in	<i>index</i>	The index in the parent's child slot array
in	<i>geometryinstance</i>	The child node to be attached

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGroupSetChild](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGroupSetChildCount](#), [rtGeometryGroupGetChildCount](#), [rtGeometryGroupGetChild](#)

3.11.3.113 **RTresult** RTAPI `rtGeometryGroupSetChildCount ( RTgeometrygroup geometrygroup, unsigned int count )`

Sets the number of child nodes to be attached to the group.

## Description

[rtGeometryGroupSetChildCount](#) specifies the number of child slots in this geometry group. Potentially existing links to children at indices greater than *count-1* are removed. If the call increases the number of slots, the newly created slots are empty and need to be filled using [rtGeometryGroupSetChild](#) before validation.

## Parameters

in	<i>geometrygroup</i>	The parent geometry group handle
in	<i>count</i>	Number of child slots to allocate for the geometry group

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGroupSetChildCount](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGroupGetChild](#), [rtGeometryGroupGetChildCount](#), [rtGeometryGroupSetChild](#)

3.11.3.114 **RTresult** RTAPI `rtGeometryGroupValidate ( RTgeometrygroup geometrygroup )`

Validates the state of the geometry group.

## Description

[rtGeometryGroupValidate](#) checks *geometrygroup* for completeness. If *geometrygroup* or any of the objects attached to *geometrygroup* are not valid, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

in	<i>geometrygroup</i>	Specifies the geometry group to be validated
----	----------------------	--

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryGroupValidate](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupCreate](#)

3.11.3.115 **RTresult** RTAPI `rtGeometryInstanceCreate ( RTcontext context, RTgeometryinstance * geometryinstance )`

Creates a new geometry instance node.

## Description

[rtGeometryInstanceCreate](#) creates a new geometry instance node within a context. *context* specifies the target context, and should be a value returned by [rtContextCreate](#). Sets *\*geometryinstance* to the handle of a newly created geometry instance within *context*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *geometryinstance* is *NULL*.

## Parameters

in	<i>context</i>	Specifies the rendering context of the GeometryInstance node
out	<i>geometryinstance</i>	New GeometryInstance node handle

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceCreate](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceDestroy](#), [rtGeometryInstanceDestroy](#), [rtGeometryInstanceGetContext](#)

3.11.3.116 **RTresult** RTAPI `rtGeometryInstanceDeclareVariable ( RTgeometryinstance geometryinstance, const char * name, RTvariable * v )`

Declares a new named variable associated with a geometry node.

## Description

[rtGeometryInstanceDeclareVariable](#) declares a new variable associated with a geometry instance node. *geometryinstance* specifies the target geometry node, and should be a value returned by [rtGeometryInstanceCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If there is currently no variable associated with *geometryinstance* named *name*, a new variable named *name* will be created and associated with *geometryinstance*. After the call, *\*v* will be set to the handle of the newly-created variable. Otherwise, *\*v* will be

set to `NULL`. After declaration, the variable can be queried with [rtGeometryInstanceQueryVariable](#) or [rtGeometryInstanceGetVariable](#). A declared variable does not have a type until its value is set with one of the `rtVariableSet` functions. Once a variable is set, its type cannot be changed anymore.

## Parameters

in	<i>geometryinstance</i>	Specifies the associated GeometryInstance node
in	<i>name</i>	The name that identifies the variable
out	<i>v</i>	Returns a handle to a newly declared variable

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceDeclareVariable](#) was introduced in OptiX 1.0.

**See also** Variables, [rtGeometryInstanceQueryVariable](#), [rtGeometryInstanceGetVariable](#), [rtGeometryInstanceRemoveVariable](#)

3.11.3.117 **RTresult** RTAPI `rtGeometryInstanceDestroy ( RTgeometryinstance geometryinstance )`

Destroys a geometry instance node.

## Description

[rtGeometryInstanceDestroy](#) removes *geometryinstance* from its context and deletes it. *geometryinstance* should be a value returned by [rtGeometryInstanceCreate](#). Associated variables declared via [rtGeometryInstanceDeclareVariable](#) are destroyed, but no child graph nodes are destroyed. After the call, *geometryinstance* is no longer a valid handle.

## Parameters

in	<i>geometryinstance</i>	Handle of the geometry instance node to destroy
----	-------------------------	---

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceDestroy](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceCreate](#)

3.11.3.118 **RTresult** RTAPI `rtGeometryInstanceGetContext ( RTgeometryinstance geometryinstance, RTcontext * context )`

Returns the context associated with a geometry instance node.

## Description

[rtGeometryInstanceGetContext](#) queries a geometry instance node for its associated context. *geometryinstance* specifies the geometry node to query, and should be a value returned by [rtGeometryInstanceCreate](#). Sets \**context* to the context associated with *geometryinstance*.



## Parameters

in	<i>geometryinstance</i>	Specifies the geometry instance
out	<i>context</i>	Handle for queried context

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceGetContext](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceGetContext](#)

3.11.3.119 **RTresult** RTAPI [rtGeometryInstanceGetGeometry](#) ( **RTgeometryinstance** *geometryinstance*, **RTgeometry** \* *geometry* )

Returns the attached Geometry node.

## Description

[rtGeometryInstanceGetGeometry](#) sets *geometry* to the handle of the attached Geometry node. If no Geometry node is attached, [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned, else [RT\\_SUCCESS](#).

## Parameters

in	<i>geometryinstance</i>	GeometryInstance node handle to query geometry
out	<i>geometry</i>	Handle to attached Geometry node

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceGetGeometry](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceCreate](#), [rtGeometryInstanceDestroy](#), [rtGeometryInstanceValidate](#), [rtGeometryInstanceSetGeometry](#)

3.11.3.120 **RTresult** RTAPI [rtGeometryInstanceGetMaterial](#) ( **RTgeometryinstance** *geometryinstance*, unsigned int *index*, **RTmaterial** \* *material* )

Returns a material handle.

## Description

[rtGeometryInstanceGetMaterial](#) returns handle *material* for the Material node at position *index* in the material list of *geometryinstance*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *index* is invalid.

## Parameters

in	<i>geometryinstance</i>	GeometryInstance node handle to query material
in	<i>index</i>	Index of material
out	<i>material</i>	Handle to material

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceGetMaterial](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceGetMaterialCount](#), [rtGeometryInstanceSetMaterial](#)

**3.11.3.121** **RTresult** RTAPI [rtGeometryInstanceGetMaterialCount](#) ( **RTgeometryinstance** *geometryinstance*, unsigned int \* *count* )

Returns the number of attached materials.

## Description

[rtGeometryInstanceGetMaterialCount](#) returns for *geometryinstance* the number of attached Material nodes *count*. The number of materies can be set with [rtGeometryInstanceSetMaterialCount](#).

## Parameters

in	<i>geometryinstance</i>	GeometryInstance node to query from the number of materials
out	<i>count</i>	Number of attached materials

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGeometryInstanceGetMaterialCount](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceSetMaterialCount](#)

**3.11.3.122** **RTresult** RTAPI [rtGeometryInstanceGetVariable](#) ( **RTgeometryinstance** *geometryinstance*, unsigned int *index*, **RTvariable** \* *v* )

Returns a handle to an indexed variable of a geometry instance node.

## Description

[rtGeometryInstanceGetVariable](#) queries the handle of a geometry instance's indexed variable. *geometryinstance* specifies the target geometry instance and should be a value returned by [rtGeometryInstanceCreate](#). *index* specifies the index of the variable, and should be a value less than [rtGeometryInstanceGetVariableCount](#). If *index* is the index of a variable attached to *geometryinstance*, returns a handle to that variable in \**v*, and *NULL* otherwise. \**v* must be declared first with [rtGeometryInstanceDeclareVariable](#) before it can be queried.

## Parameters

in	<i>geometryinstance</i>	The GeometryInstance node from which to query a variable
in	<i>index</i>	The index that identifies the variable to be queried
out	<i>v</i>	Returns handle to indexed variable

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

## History

[rtGeometryInstanceGetVariable](#) was introduced in OptiX 1.0.

**See also** [rtGeometryDeclareVariable](#), [rtGeometryGetVariableCount](#), [rtGeometryRemoveVariable](#), [rtGeometryQueryVariable](#)

**3.11.3.123** `RTresult RTAPI rtGeometryInstanceGetVariableCount ( RTgeometryinstance geometryinstance, unsigned int * count )`

Returns the number of attached variables.

## Description

[rtGeometryInstanceGetVariableCount](#) queries the number of variables attached to a geometry instance. *geometryinstance* specifies the geometry instance, and should be a value returned by [rtGeometryInstanceCreate](#). After the call, the number of variables attached to *geometryinstance* is returned to *\*count*.

## Parameters

in	<i>geometryinstance</i>	The GeometryInstance node to query from the number of attached variables
out	<i>count</i>	Returns the number of attached variables

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceGetVariableCount](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceGetVariableCount](#), [rtGeometryInstanceDeclareVariable](#), [rtGeometryInstanceRemoveVariable](#)

**3.11.3.124** `RTresult RTAPI rtGeometryInstanceQueryVariable ( RTgeometryinstance geometryinstance, const char * name, RTvariable * v )`

Returns a handle to a named variable of a geometry node.

## Description

[rtGeometryInstanceQueryVariable](#) queries the handle of a geometry instance node's named variable. *geometryinstance* specifies the target geometry instance node, as returned by [rtGeometryInstanceCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If *name* is the name of a variable attached to *geometryinstance*, returns a handle to that variable in *\*v*, otherwise *NULL*. Geometry instance variables have to be declared with [rtGeometryInstanceDeclareVariable](#) before they can be queried.

### Parameters

in	<i>geometryinstance</i>	The GeometryInstance node to query from a variable
in	<i>name</i>	The name that identifies the variable to be queried
out	<i>v</i>	Returns the named variable

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtGeometryInstanceQueryVariable](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceDeclareVariable](#), [rtGeometryInstanceRemoveVariable](#), [rtGeometryInstanceGetVariableCount](#), [rtGeometryInstanceGetVariable](#)

3.11.3.125 **RTresult** RTAPI [rtGeometryInstanceRemoveVariable](#) ( **RTgeometryinstance** *geometryinstance*, **RTvariable** *v* )

Removes a named variable from a geometry instance node.

## Description

[rtGeometryInstanceRemoveVariable](#) removes a named variable from a geometry instance. The target geometry instance is specified by *geometryinstance*, which should be a value returned by [rtGeometryInstanceCreate](#). The variable to be removed is specified by *v*, which should be a value returned by [rtGeometryInstanceDeclareVariable](#). Once a variable has been removed from this geometry instance, another variable with the same name as the removed variable may be declared.

### Parameters

in	<i>geometryinstance</i>	The GeometryInstance node from which to remove a variable
in	<i>v</i>	The variable to be removed

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

## History

[rtGeometryInstanceRemoveVariable](#) was introduced in OptiX 1.0.

**See also** [rtContextRemoveVariable](#), [rtGeometryInstanceDeclareVariable](#)

**3.11.3.126** `RTresult RTAPI rtGeometryInstanceSetGeometry ( RTgeometryinstance geometryinstance, RTgeometry geometry )`

Attaches a Geometry node.

## Description

[rtGeometryInstanceSetGeometry](#) attaches a Geometry node to a GeometryInstance. Only *one* Geometry node can be attached to a GeometryInstance. However, it is at any time possible to attach a different Geometry node.

## Parameters

in	<i>geometryinstance</i>	GeometryInstance node handle to attach geometry
in	<i>geometry</i>	Geometry handle to attach to <i>geometryinstance</i>

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceSetGeometry](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceGetGeometry](#)

**3.11.3.127** `RTresult RTAPI rtGeometryInstanceSetMaterial ( RTgeometryinstance geometryinstance, unsigned int index, RTmaterial material )`

Sets a material.

## Description

[rtGeometryInstanceSetMaterial](#) attaches *material* to *geometryinstance* at position *index* in its internal Material node list. *index* must be in the range 0 to [rtGeometryInstanceGetMaterialCount](#) - 1.

## Parameters

in	<i>geometryinstance</i>	GeometryInstance node for which to set a material
in	<i>index</i>	Index into the material list
in	<i>material</i>	Material handle to attach to <i>geometryinstance</i>

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceSetMaterial](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceGetMaterialCount](#), [rtGeometryInstanceSetMaterialCount](#)

**3.11.3.128** **RTresult** RTAPI `rtGeometryInstanceSetMaterialCount` ( **RTgeometryinstance** *geometryinstance*, unsigned int *count* )

Sets the number of materials.

## Description

[rtGeometryInstanceSetMaterialCount](#) sets the number of materials *count* that will be attached to *geometryinstance*. The number of attached materials can be changed at any time. Increasing the number of materials will not modify already assigned materials. Decreasing the number of materials will not modify the remaining already assigned materials.

## Parameters

in	<i>geometryinstance</i>	GeometryInstance node to set number of materials
in	<i>count</i>	Number of materials to be set

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceSetMaterialCount](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceGetMaterialCount](#)

**3.11.3.129** **RTresult** RTAPI `rtGeometryInstanceValidate` ( **RTgeometryinstance** *geometryinstance* )

Checks a GeometryInstance node for internal consistency.

## Description

[rtGeometryInstanceValidate](#) checks *geometryinstance* for completeness. If *geometryinstance* or any of the objects attached to *geometry* are not valid, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

in	<i>geometryinstance</i>	GeometryInstance node of a model sub-tree to be validated
----	-------------------------	---

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometryInstanceValidate](#) was introduced in OptiX 1.0.

**See also** [rtGeometryInstanceCreate](#)

#### 3.11.3.130 `RTresult RTAPI rtGeometryIsDirty ( RTgeometry geometry, int * dirty )`

Returns the dirty flag.

##### Description

[rtGeometryIsDirty](#) returns the dirty flag of *geometry*. The dirty flag for geometry nodes can be set with [rtGeometryMarkDirty](#). By default the flag is *1* for a new geometry node, indicating dirty. After a call to `rtContextLaunch` the flag is automatically set to *0*. When the dirty flag is set, the geometry data is uploaded automatically to the device while a `rtContextLaunch` call.

##### Parameters

in	<i>geometry</i>	The geometry node to query from the dirty flag
out	<i>dirty</i>	Dirty flag

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

##### History

[rtGeometryIsDirty](#) was introduced in OptiX 1.0.

**See also** `rtContextLaunch`, [rtGeometryMarkDirty](#)

#### 3.11.3.131 `RTresult RTAPI rtGeometryMarkDirty ( RTgeometry geometry )`

Sets the dirty flag.

##### Description

[rtGeometryMarkDirty](#) sets for *geometry* the dirty flag. By default the dirty flag is set for a new Geometry node. After a call to `rtContextLaunch` the flag is automatically cleared. When the dirty flag is set, the geometry data is uploaded automatically to the device while a `rtContextLaunch` call.

##### Parameters

in	<i>geometry</i>	The geometry node to mark as dirty
----	-----------------	------------------------------------

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

##### History

[rtGeometryMarkDirty](#) was introduced in OptiX 1.0.

**See also** [rtGeometryIsDirty](#)

**3.11.3.132** `RTresult RTAPI rtGeometryQueryVariable ( RTgeometry geometry, const char * name, RTvariable * v )`

Returns a handle to a named variable of a geometry node.

**Description**

`rtGeometryQueryVariable` queries the handle of a geometry node's named variable. *geometry* specifies the target geometry node and should be a value returned by `rtGeometryCreate`. *name* specifies the name of the variable, and should be a *NULL-terminated* string. If *name* is the name of a variable attached to *geometry*, returns a handle to that variable in *\*v* or *NULL* otherwise. Geometry variables must be declared with `rtGeometryDeclareVariable` before they can be queried.

**Parameters**

in	<i>geometry</i>	The geometry node to query from a variable
in	<i>name</i>	The name that identifies the variable to be queried
out	<i>v</i>	Returns the named variable

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`
- `RT_ERROR_VARIABLE_NOT_FOUND`

**History**

`rtGeometryQueryVariable` was introduced in OptiX 1.0.

**See also** `rtGeometryDeclareVariable`, `rtGeometryRemoveVariable`, `rtGeometryGetVariableCount`, `rtGeometryGetVariable`

**3.11.3.133** `RTresult RTAPI rtGeometryRemoveVariable ( RTgeometry geometry, RTvariable v )`

Removes a named variable from a geometry node.

**Description**

`rtGeometryRemoveVariable` removes a named variable from a geometry node. The target geometry is specified by *geometry*, which should be a value returned by `rtGeometryCreate`. The variable to remove is specified by *v*, which should be a value returned by `rtGeometryDeclareVariable`. Once a variable has been removed from this geometry node, another variable with the same name as the removed variable may be declared.

**Parameters**

in	<i>geometry</i>	The geometry node from which to remove a variable
in	<i>v</i>	The variable to be removed

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`
- `RT_ERROR_VARIABLE_NOT_FOUND`



## History

[rtGeometryRemoveVariable](#) was introduced in OptiX 1.0.

**See also** [rtContextRemoveVariable](#)

### 3.11.3.134 `RTresult RTAPI rtGeometrySetBoundingBoxProgram ( RTgeometry geometry, RTprogram program )`

Sets the bounding box program.

## Description

[rtGeometrySetBoundingBoxProgram](#) sets for *geometry* the *program* that computes an axis aligned bounding box for each attached primitive to *geometry*. RTprogram's can be either generated with [rtProgramCreateFromPTXFile](#) or [rtProgramCreateFromPTXString](#). A bounding box program is mandatory for every geometry node.

## Parameters

in	<i>geometry</i>	The geometry node for which to set the bounding box program
in	<i>program</i>	Handle to the bounding box program

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

## History

[rtGeometrySetBoundingBoxProgram](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGetBoundingBoxProgram](#), [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

### 3.11.3.135 `RTresult RTAPI rtGeometrySetIntersectionProgram ( RTgeometry geometry, RTprogram program )`

Sets the intersection program.

## Description

[rtGeometrySetIntersectionProgram](#) sets for *geometry* the *program* that performs ray primitive intersections. RTprogram's can be either generated with [rtProgramCreateFromPTXFile](#) or [rtProgramCreateFromPTXString](#). An intersection program is mandatory for every geometry node.

## Parameters

in	<i>geometry</i>	The geometry node for which to set the intersection program
in	<i>program</i>	A handle to the ray primitive intersection program

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

## History

[rtGeometrySetIntersectionProgram](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGetIntersectionProgram](#), [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.11.3.136 **RTresult** RTAPI [rtGeometrySetPrimitiveCount](#) ( **RTgeometry** *geometry*, unsigned int *num\_primitives* )

Sets the number of primitives.

## Description

[rtGeometrySetPrimitiveCount](#) sets the number of primitives *num\_primitives* in *geometry*.

### Parameters

in	<i>geometry</i>	The geometry node for which to set the number of primitives
in	<i>num_primitives</i>	The number of primitives

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtGeometrySetPrimitiveCount](#) was introduced in OptiX 1.0.

**See also** [rtGeometryGetPrimitiveCount](#)

3.11.3.137 **RTresult** RTAPI [rtGeometrySetPrimitiveIndexOffset](#) ( **RTgeometry** *geometry*, unsigned int *index\_offset* )

Sets the primitive index offset.

## Description

[rtGeometrySetPrimitiveIndexOffset](#) sets the primitive index offset *index\_offset* in *geometry*. In the past, a Geometry object's primitive index range always started at zero (e.g., a Geometry with *N* primitives would have a primitive index range of [0,N-1]). The index offset is used to allow Geometry objects to have primitive index ranges starting at non-zero positions (e.g., a Geometry with *N* primitives and an index offset of *M* would have a primitive index range of [M,M+N-1]). This feature enables the sharing of vertex index buffers between multiple Geometry objects.

### Parameters

in	<i>geometry</i>	The geometry node for which to set the primitive index offset
in	<i>index_offset</i>	The primitive index offset

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGeometrySetPrimitiveIndexOffset](#) was introduced in OptiX 3.5.

**See also** [rtGeometryGetPrimitiveIndexOffset](#)

**3.11.3.138** `RTresult RTAPI rtGeometryValidate ( RTgeometry geometry )`

Validates the geometry nodes integrity.

**Description**

`rtGeometryValidate` checks *geometry* for completeness. If *geometry* or any of the objects attached to *geometry* are not valid, returns `RT_ERROR_INVALID_VALUE`.

**Parameters**

<code>in</code>	<code>geometry</code>	The geometry node to be validated
-----------------	-----------------------	-----------------------------------

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`

**History**

`rtGeometryValidate` was introduced in OptiX 1.0.

**See also** `rtContextValidate`

**3.11.3.139** `RTresult RTAPI rtGetVersion ( unsigned int * version )`

Returns the current OptiX version.

**Description**

`rtGetVersion` returns in *version* a numerically comparable version number of the current OptiX library.

**Parameters**

<code>out</code>	<code>version</code>	OptiX version number
------------------	----------------------	----------------------

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

**History**

`rtGetVersion` was introduced in OptiX 1.0.

**See also** `rtDeviceGetDeviceCount`

**3.11.3.140** `RTresult RTAPI rtGroupCreate ( RTcontext context, RTgroup * group )`

Creates a new group.

**Description**

`rtGroupCreate` creates a new group within a context. *context* specifies the target context, and should be a value returned by `rtContextCreate`. Sets *\*group* to the handle of a newly created group within *context*. Returns `RT_ERROR_INVALID_VALUE` if *group* is `NULL`.

## Parameters

in	<i>context</i>	Specifies a context within which to create a new group
out	<i>group</i>	Returns a newly created group

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGroupCreate](#) was introduced in OptiX 1.0.

See also [rtGroupDestroy](#), [rtContextCreate](#)

3.11.3.141 `RTresult RTAPI rtGroupDestroy ( RTgroup group )`

Destroys a group node.

## Description

[rtGroupDestroy](#) removes *group* from its context and deletes it. *group* should be a value returned by [rtGroupCreate](#). No child graph nodes are destroyed. After the call, *group* is no longer a valid handle.

## Parameters

in	<i>group</i>	Handle of the group node to destroy
----	--------------	-------------------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGroupDestroy](#) was introduced in OptiX 1.0.

See also [rtGroupCreate](#)

3.11.3.142 `RTresult RTAPI rtGroupGetAcceleration ( RTgroup group, RTacceleration * acceleration )`

Returns the acceleration structure attached to a group.

## Description

[rtGroupGetAcceleration](#) returns the acceleration structure attached to a group using [rtGroupSetAcceleration](#). If no acceleration structure has previously been set, *\*acceleration* is set to `NULL`.

## Parameters

in	<i>group</i>	The group handle
out	<i>acceleration</i>	The returned acceleration structure object

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGroupGetAcceleration](#) was introduced in OptiX 1.0.

**See also** [rtGroupSetAcceleration](#), [rtAccelerationCreate](#)

3.11.3.143 **RTresult** RTAPI `rtGroupGetChild ( RTgroup group, unsigned int index, RObject * child )`

Returns a child node of a group.

## Description

[rtGroupGetChild](#) returns the child object at slot *index* of the parent *group*. If no child has been assigned to the given slot, \**child* is set to `NULL`. Returns `RT_ERROR_INVALID_VALUE` if given an invalid child index or `NULL` pointer.

## Parameters

in	<i>group</i>	The parent group handle
in	<i>index</i>	The index of the child slot to query
out	<i>child</i>	The returned child object

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGroupGetChild](#) was introduced in OptiX 1.0.

**See also** [rtGroupSetChild](#), [rtGroupSetChildCount](#), [rtGroupGetChildCount](#), [rtGroupGetChildType](#)

3.11.3.144 **RTresult** RTAPI `rtGroupGetChildCount ( RTgroup group, unsigned int * count )`

Returns the number of child slots for a group.

## Description

[rtGroupGetChildCount](#) returns the number of child slots allocated using [rtGroupSetChildCount](#). This includes empty slots which may not yet have actual children assigned by [rtGroupSetChild](#). Returns `RT_ERROR_INVALID_VALUE` if given a `NULL` pointer.

## Parameters

in	<i>group</i>	The parent group handle
out	<i>count</i>	Returned number of child slots

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGroupGetChildCount](#) was introduced in OptiX 1.0.

**See also** [rtGroupSetChild](#), [rtGroupGetChild](#), [rtGroupSetChildCount](#), [rtGroupGetChildType](#)

3.11.3.145 **RTresult** RTAPI `rtGroupGetChildType ( RTgroup group, unsigned int index, RObjecttype * type )`

Get the type of a group child.

## Description

`rtGroupGetChildType` returns the type of the group child at slot *index*. If no child is associated with the given index, *\*type* is set to `RT_OBJECTTYPE_UNKNOWN` and `RT_ERROR_INVALID_VALUE` is returned. Returns `RT_ERROR_INVALID_VALUE` if given a `NULL` pointer.

#### Parameters

in	<i>group</i>	The parent group handle
in	<i>index</i>	The index of the child slot to query
out	<i>type</i>	The returned child type

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

#### History

`rtGroupGetChildType` was introduced in OptiX 1.0.

**See also** `rtGroupSetChild`, `rtGroupGetChild`, `rtGroupSetChildCount`, `rtGroupGetChildCount`

#### 3.11.3.146 `RTresult RTAPI rtGroupGetContext ( RTgroup group, RTcontext * context )`

Returns the context associated with a group.

#### Description

`rtGroupGetContext` queries a group for its associated context. *group* specifies the group to query, and must be a value returned by `rtGroupCreate`. Sets *\*context* to the context associated with *group*.

#### Parameters

in	<i>group</i>	Specifies the group to query
out	<i>context</i>	Returns the context associated with the group

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

#### History

`rtGroupGetContext` was introduced in OptiX 1.0.

**See also** `rtContextCreate`, `rtGroupCreate`

#### 3.11.3.147 `RTresult RTAPI rtGroupSetAcceleration ( RTgroup group, RTacceleration acceleration )`

Set the acceleration structure for a group.

#### Description

`rtGroupSetAcceleration` attaches an acceleration structure to a group. The acceleration structure must have been previously created using `rtAccelerationCreate`. Every group is required to have an acceleration structure assigned in order to pass validation. The acceleration structure will be built over the children of the group. For example, if an acceleration structure is attached to a group that has a selector, a geometry group, and a transform child, the acceleration structure will be built over the bounding volumes of these three objects.

Note that it is legal to attach a single `RTacceleration` object to multiple groups, as long as the underlying bounds of the children are the same. For example, if another group has three children which are known to have the same bounding volumes as the ones in the example above, the two groups can share an acceleration structure, thus saving build time. This is true even if the details of the children, such as the actual type of a node or its geometry

content, differ from the first set of group children. All that is required is for a child node at a given index to have the same bounds as the other group's child node at the same index.

Sharing an acceleration structure this way corresponds to attaching an acceleration structure to multiple geometry groups at lower graph levels using [rtGeometryGroupSetAcceleration](#).

#### Parameters

<i>in</i>	<i>group</i>	The group handle
<i>in</i>	<i>acceleration</i>	The acceleration structure to attach to the group

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtGroupSetAcceleration](#) was introduced in OptiX 1.0.

**See also** [rtGroupGetAcceleration](#), [rtAccelerationCreate](#), [rtGeometryGroupSetAcceleration](#)

#### 3.11.3.148 `RTresult RTAPI rtGroupSetChild ( RTgroup group, unsigned int index, RTOBJECT child )`

Attaches a child node to a group.

#### Description

Attaches a new child node *child* to the parent node *group*. *index* specifies the number of the slot where the child node gets attached. A sufficient number of slots must be allocated using [rtGroupSetChildCount](#). Legal child node types are [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), and [RTtransform](#).

#### Parameters

<i>in</i>	<i>group</i>	The parent group handle
<i>in</i>	<i>index</i>	The index in the parent's child slot array
<i>in</i>	<i>child</i>	The child node to be attached. Can be of type { <a href="#">RTgroup</a> , <a href="#">RTselector</a> , <a href="#">RTgeometrygroup</a> , <a href="#">RTtransform</a> }

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtGroupSetChild](#) was introduced in OptiX 1.0.

**See also** [rtGroupSetChildCount](#), [rtGroupGetChildCount](#), [rtGroupGetChild](#), [rtGroupGetChildType](#)

#### 3.11.3.149 `RTresult RTAPI rtGroupSetChildCount ( RTgroup group, unsigned int count )`

Sets the number of child nodes to be attached to the group.

#### Description

[rtGroupSetChildCount](#) specifies the number of child slots in this group. Potentially existing links to children at indices greater than *count-1* are removed. If the call increases the number of slots, the newly created slots are empty and need to be filled using [rtGroupSetChild](#) before validation.

## Parameters

in	<i>group</i>	The parent group handle
in	<i>count</i>	Number of child slots to allocate for the group

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGroupSetChildCount](#) was introduced in OptiX 1.0.

**See also** [rtGroupGetChild](#), [rtGroupGetChildCount](#), [rtGroupGetChildType](#), [rtGroupSetChild](#)

3.11.3.150 `RTresult RTAPI rtGroupValidate ( RTgroup group )`

Verifies the state of the group.

## Description

[rtGroupValidate](#) checks *group* for completeness. If *group* or any of the objects attached to *group* are not valid, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

in	<i>group</i>	Specifies the group to be validated
----	--------------	-------------------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtGroupValidate](#) was introduced in OptiX 1.0.

**See also** [rtGroupCreate](#)

3.11.3.151 `RTresult RTAPI rtMaterialCreate ( RTcontext context, RTmaterial * material )`

Creates a new material.

## Description

[rtMaterialCreate](#) creates a new material within a context. *context* specifies the target context, as returned by [rtContextCreate](#). Sets *\*material* to the handle of a newly created material within *context*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *material* is *NULL*.

## Parameters

in	<i>context</i>	Specifies a context within which to create a new material
out	<i>material</i>	Returns a newly created material

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)



- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtMaterialCreate](#) was introduced in OptiX 1.0.

See also [rtMaterialDestroy](#), [rtContextCreate](#)

3.11.3.152 **RTresult** RTAPI [rtMaterialDeclareVariable](#) ( **RTmaterial** *material*, **const char \*** *name*, **RTvariable \*** *v* )

Declares a new named variable to be associated with a material.

### Description

[rtMaterialDeclareVariable](#) declares a new variable to be associated with a material. *material* specifies the target material, and should be a value returned by [rtMaterialCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If there is currently no variable associated with *material* named *name*, and *v* is not *NULL*, a new variable named *name* will be created and associated with *material* and *\*v* will be set to the handle of the newly-created variable. Otherwise, this call has no effect and returns either [RT\\_ERROR\\_INVALID\\_VALUE](#) if either *name* or *v* is *NULL* or [RT\\_ERROR\\_VARIABLE\\_REDECLARED](#) if *name* is the name of an existing variable associated with the material.

### Parameters

in	<i>material</i>	Specifies the material to modify
in	<i>name</i>	Specifies the name of the variable
out	<i>v</i>	Returns a handle to a newly declared variable

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_REDECLARED](#)
- [RT\\_ERROR\\_ILLEGAL\\_SYMBOL](#)

### History

[rtMaterialDeclareVariable](#) was introduced in OptiX 1.0.

See also [rtMaterialGetVariable](#), [rtMaterialQueryVariable](#), [rtMaterialCreate](#)

3.11.3.153 **RTresult** RTAPI [rtMaterialDestroy](#) ( **RTmaterial** *material* )

Destroys a material object.

### Description

[rtMaterialDestroy](#) removes *material* from its context and deletes it. *material* should be a value returned by [rtMaterialCreate](#). Associated variables declared via [rtMaterialDeclareVariable](#) are destroyed, but no child graph nodes are destroyed. After the call, *material* is no longer a valid handle.

### Parameters

in	<i>material</i>	Handle of the material node to destroy
----	-----------------	--

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)

- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtMaterialDestroy](#) was introduced in OptiX 1.0.

See also [rtMaterialCreate](#)

**3.11.3.154 RTresult RTAPI rtMaterialGetAnyHitProgram ( RTmaterial *material*, unsigned int *ray\_type\_index*, RTprogram \* *program* )**

Returns the any hit program associated with a (material, ray type) tuple.

### Description

[rtMaterialGetAnyHitProgram](#) queries the any hit program associated with a (material, ray type) tuple. *material* specifies the material of interest and should be a value returned by [rtMaterialCreate](#). *ray\_type\_index* specifies the target ray type and should be a value less than the value returned by [rtContextGetRayTypeCount](#). If all parameters are valid, \**program* sets to the handle of the any hit program associated with the tuple (*material*, *ray\_type\_index*). Otherwise, the call has no effect and returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

### Parameters

in	<i>material</i>	Specifies the material of the (material, ray type) tuple to query
in	<i>ray_type_index</i>	Specifies the type of ray of the (material, ray type) tuple to query
out	<i>program</i>	Returns the any hit program associated with the (material, ray type) tuple

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtMaterialGetAnyHitProgram](#) was introduced in OptiX 1.0.

See also [rtMaterialSetAnyHitProgram](#), [rtMaterialCreate](#), [rtContextGetRayTypeCount](#)

**3.11.3.155 RTresult RTAPI rtMaterialGetClosestHitProgram ( RTmaterial *material*, unsigned int *ray\_type\_index*, RTprogram \* *program* )**

Returns the closest hit program associated with a (material, ray type) tuple.

### Description

[rtMaterialGetClosestHitProgram](#) queries the closest hit program associated with a (material, ray type) tuple. *material* specifies the material of interest and should be a value returned by [rtMaterialCreate](#). *ray\_type\_index* specifies the target ray type and should be a value less than the value returned by [rtContextGetRayTypeCount](#). If all parameters are valid, \**program* sets to the handle of the any hit program associated with the tuple (*material*, *ray\_type\_index*). Otherwise, the call has no effect and returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

### Parameters

in	<i>material</i>	Specifies the material of the (material, ray type) tuple to query
in	<i>ray_type_index</i>	Specifies the type of ray of the (material, ray type) tuple to query
out	<i>program</i>	Returns the closest hit program associated with the (material, ray type) tuple

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtMaterialGetClosestHitProgram](#) was introduced in OptiX 1.0.

**See also** [rtMaterialSetClosestHitProgram](#), [rtMaterialCreate](#), [rtContextGetRayTypeCount](#)

3.11.3.156 **RTresult** RTAPI [rtMaterialGetContext](#) ( **RTmaterial** *material*, **RTcontext** \* *context* )

Returns the context associated with a material.

## Description

[rtMaterialGetContext](#) queries a material for its associated context. *material* specifies the material to query, and should be a value returned by [rtMaterialCreate](#). If both parameters are valid, \**context* sets to the context associated with *material*. Otherwise, the call has no effect and returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

### Parameters

in	<i>material</i>	Specifies the material to query
out	<i>context</i>	Returns the context associated with the material

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtMaterialGetContext](#) was introduced in OptiX 1.0.

**See also** [rtMaterialCreate](#)

3.11.3.157 **RTresult** RTAPI [rtMaterialGetVariable](#) ( **RTmaterial** *material*, unsigned int *index*, **RTvariable** \* *v* )

Returns a handle to an indexed variable of a material.

## Description

[rtMaterialGetVariable](#) queries the handle of a material's indexed variable. *material* specifies the target material and should be a value returned by [rtMaterialCreate](#). *index* specifies the index of the variable, and should be a value less than [rtMaterialGetVariableCount](#). If *material* is a valid material and *index* is the index of a variable attached to *material*, \**v* is set to a handle to that variable. Otherwise, \**v* is set to *NULL* and either [RT\\_ERROR\\_INVALID\\_VALUE](#) or [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#) is returned depending on the validity of *material*, or *index*, respectively.

### Parameters

in	<i>material</i>	Specifies the material to query
in	<i>index</i>	Specifies the index of the variable to query
out	<i>v</i>	Returns the indexed variable

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

## History

[rtMaterialGetVariable](#) was introduced in OptiX 1.0.

**See also** [rtMaterialQueryVariable](#), [rtMaterialGetVariableCount](#), [rtMaterialCreate](#)

### 3.11.3.158 RTresult RTAPI rtMaterialGetVariableCount ( RTmaterial *material*, unsigned int \* *count* )

Returns the number of variables attached to a material.

#### Description

[rtMaterialGetVariableCount](#) queries the number of variables attached to a material. *material* specifies the material, and should be a value returned by [rtMaterialCreate](#). After the call, if both parameters are valid, the number of variables attached to *material* is returned to \**count*. Otherwise, the call has no effect and returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

#### Parameters

in	<i>material</i>	Specifies the material to query
out	<i>count</i>	Returns the number of variables

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtMaterialGetVariableCount](#) was introduced in OptiX 1.0.

See also [rtMaterialCreate](#)

### 3.11.3.159 RTresult RTAPI rtMaterialQueryVariable ( RTmaterial *material*, const char \* *name*, RTvariable \* *v* )

Queries for the existence of a named variable of a material.

#### Description

[rtMaterialQueryVariable](#) queries for the existence of a material's named variable. *material* specifies the target material and should be a value returned by [rtMaterialCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If *material* is a valid material and *name* is the name of a variable attached to *material*, \**v* is set to a handle to that variable after the call. Otherwise, \**v* is set to *NULL*. If *material* is not a valid material, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

#### Parameters

in	<i>material</i>	Specifies the material to query
in	<i>name</i>	Specifies the name of the variable to query
out	<i>v</i>	Returns a the named variable, if it exists

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtMaterialQueryVariable](#) was introduced in OptiX 1.0.

See also [rtMaterialGetVariable](#), [rtMaterialCreate](#)

### 3.11.3.160 RTresult RTAPI rtMaterialRemoveVariable ( RTmaterial *material*, RTvariable *v* )

Removes a variable from a material.

#### Description

[rtMaterialRemoveVariable](#) removes a variable from a material. The material of interest is specified by *material*, which should be a value returned by [rtMaterialCreate](#). The variable to remove is specified by *v*, which should be a value returned by [rtMaterialDeclareVariable](#). Once a variable has been removed from this material, another variable with the same name as the removed variable may be declared. If *material* does not refer to a valid material, this call has no effect and returns [RT\\_ERROR\\_INVALID\\_VALUE](#). If *v* is not a valid variable or does not belong to *material*, this call has no effect and returns [RT\\_ERROR\\_INVALID\\_VALUE](#) or [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#), respectively.

#### Parameters

in	<i>material</i>	Specifies the material to modify
in	<i>v</i>	Specifies the variable to remove

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

#### History

[rtMaterialRemoveVariable](#) was introduced in OptiX 1.0.

See also [rtMaterialDeclareVariable](#), [rtMaterialCreate](#)

3.11.3.161 **RTresult** RTAPI [rtMaterialSetAnyHitProgram](#) ( **RTmaterial** *material*, unsigned int *ray\_type\_index*, **RTprogram** *program* )

Sets the any hit program associated with a (material, ray type) tuple.

#### Description

[rtMaterialSetAnyHitProgram](#) specifies an any hit program to associate with a (material, ray type) tuple. *material* specifies the target material and should be a value returned by [rtMaterialCreate](#). *ray\_type\_index* specifies the type of ray to which the program applies and should be a value less than the value returned by [rtContextGetRayTypeCount](#). *program* specifies the target any hit program which applies to the tuple (*material*, *ray\_type\_index*) and should be a value returned by either [rtProgramCreateFromPTXString](#) or [rtProgramCreateFromPTXFile](#).

#### Parameters

in	<i>material</i>	Specifies the material of the (material, ray type) tuple to modify
in	<i>ray_type_index</i>	Specifies the type of ray of the (material, ray type) tuple to modify
in	<i>program</i>	Specifies the any hit program to associate with the (material, ray type) tuple

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

#### History

[rtMaterialSetAnyHitProgram](#) was introduced in OptiX 1.0.

**See also** [rtMaterialGetAnyHitProgram](#), [rtMaterialCreate](#), [rtContextGetRayTypeCount](#), [rtProgramCreateFromPTXString](#), [rtProgramCreateFromPTXFile](#)

**3.11.3.162** `RTresult RTAPI rtMaterialSetClosestHitProgram ( RTmaterial material, unsigned int ray_type_index, RTprogram program )`

Sets the closest hit program associated with a (material, ray type) tuple.

#### Description

[rtMaterialSetClosestHitProgram](#) specifies a closest hit program to associate with a (material, ray type) tuple. *material* specifies the material of interest and should be a value returned by [rtMaterialCreate](#). *ray\_type\_index* specifies the type of ray to which the program applies and should be a value less than the value returned by [rtContextGetRayTypeCount](#). *program* specifies the target closest hit program which applies to the tuple (*material*, *ray\_type\_index*) and should be a value returned by either [rtProgramCreateFromPTXString](#) or [rtProgramCreateFromPTXFile](#).

#### Parameters

in	<i>material</i>	Specifies the material of the (material, ray type) tuple to modify
in	<i>ray_type_index</i>	Specifies the ray type of the (material, ray type) tuple to modify
in	<i>program</i>	Specifies the closest hit program to associate with the (material, ray type) tuple

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

#### History

[rtMaterialSetClosestHitProgram](#) was introduced in OptiX 1.0.

**See also** [rtMaterialGetClosestHitProgram](#), [rtMaterialCreate](#), [rtContextGetRayTypeCount](#), [rtProgramCreateFromPTXString](#), [rtProgramCreateFromPTXFile](#)

**3.11.3.163** `RTresult RTAPI rtMaterialValidate ( RTmaterial material )`

Verifies the state of a material.

#### Description

[rtMaterialValidate](#) checks *material* for completeness. If *material* or any of the objects attached to *material* are not valid, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

#### Parameters

in	<i>material</i>	Specifies the material to be validated
----	-----------------	--

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtMaterialValidate](#) was introduced in OptiX 1.0.

**See also** [rtMaterialCreate](#)

3.11.3.164 **RTresult** RTAPI [rtProgramCreateFromPTXFile](#) ( **RTcontext** *context*, **const char \*** *filename*, **const char \*** *program\_name*, **RTprogram \*** *program* )

Creates a new program object.

## Description

[rtProgramCreateFromPTXFile](#) allocates and returns a handle to a new program object. The program is created from PTX code held in *filename* from function *program\_name*.

### Parameters

in	<i>context</i>	The context to create the program in
in	<i>filename</i>	Path to the file containing the PTX code
in	<i>program_name</i>	The name of the PTX function to create the program from
in	<i>program</i>	Handle to the program to be created

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_INVALID\\_SOURCE](#)
- [RT\\_ERROR\\_FILE\\_NOT\\_FOUND](#)

## History

[rtProgramCreateFromPTXFile](#) was introduced in OptiX 1.0.

**See also** [RT\\_PROGRAM](#), [rtProgramCreateFromPTXString](#), [rtProgramDestroy](#)

3.11.3.165 **RTresult** RTAPI [rtProgramCreateFromPTXString](#) ( **RTcontext** *context*, **const char \*** *ptx*, **const char \*** *program\_name*, **RTprogram \*** *program* )

Creates a new program object.

## Description

[rtProgramCreateFromPTXString](#) allocates and returns a handle to a new program object. The program is created from PTX code held in the *NULL-terminated* string *ptx* from function *program\_name*.

### Parameters

in	<i>context</i>	The context to create the program in
in	<i>ptx</i>	The string containing the PTX code
in	<i>program_name</i>	The name of the PTX function to create the program from
in	<i>program</i>	Handle to the program to be created

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)

- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_INVALID\\_SOURCE](#)

### History

[rtProgramCreateFromPTXString](#) was introduced in OptiX 1.0.

**See also** [RT\\_PROGRAM](#), [rtProgramCreateFromPTXFile](#), [rtProgramDestroy](#)

**3.11.3.166** `RTresult RTAPI rtProgramDeclareVariable ( RTprogram program, const char * name, RTvariable * v )`

Declares a new named variable associated with a program.

### Description

[rtProgramDeclareVariable](#) declares a new variable, *name*, and associates it with the program. A variable can only be declared with the same name once on the program. Any attempt to declare multiple variables with the same name will cause the call to fail and return [RT\\_ERROR\\_VARIABLE\\_REDECLARED](#). If *name* or *v* is *NULL* returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

### Parameters

in	<i>program</i>	The program the declared variable will be attached to
in	<i>name</i>	The name of the variable to be created
out	<i>v</i>	Return handle to the variable to be created

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_REDECLARED](#)
- [RT\\_ERROR\\_ILLEGAL\\_SYMBOL](#)

### History

[rtProgramDeclareVariable](#) was introduced in OptiX 1.0.

**See also** [rtProgramRemoveVariable](#), [rtProgramGetVariable](#), [rtProgramGetVariableCount](#), [rtProgramQueryVariable](#)

**3.11.3.167** `RTresult RTAPI rtProgramDestroy ( RTprogram program )`

Destroys a program object.

### Description

[rtProgramDestroy](#) removes *program* from its context and deletes it. *program* should be a value returned by *rtProgramCreate\**. Associated variables declared via [rtProgramDeclareVariable](#) are destroyed. After the call, *program* is no longer a valid handle.

### Parameters



<code>in</code>	<code>program</code>	Handle of the program to destroy
-----------------	----------------------	----------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtProgramDestroy](#) was introduced in OptiX 1.0.

**See also** [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.11.3.168 **RTresult** RTAPI [rtProgramGetContext](#) ( **RTprogram** *program*, **RTcontext** \* *context* )

Gets the context object that created a program.

**Description**

[rtProgramGetContext](#) returns a handle to the context object that was used to create *program*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *context* is `NULL`.

**Parameters**

<code>in</code>	<code>program</code>	The program to be queried for its context object
<code>out</code>	<code>context</code>	The return handle for the requested context object

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtProgramGetContext](#) was introduced in OptiX 1.0.

**See also** [rtContextCreate](#)

3.11.3.169 **RTresult** RTAPI [rtProgramGetId](#) ( **RTprogram** *program*, **int** \* *program\_id* )

Returns the ID for the Program object.

**Description**

[rtProgramGetId](#) returns an ID for the provided program. The returned ID is used to reference *program* from device code. If *program\_id* is `NULL` or the *program* is not a valid *RTprogram*, returns [RT\\_ERROR\\_INVALID\\_VALUE](#). [RT\\_PROGRAM\\_ID\\_NULL](#) can be used as a sentinel for a non-existent program, since this value will never be returned as a valid program id.

**Parameters**

in	<i>program</i>	The program to be queried for its id
out	<i>program_id</i>	The returned ID of the program.

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtProgramGetId](#) was introduced in OptiX 3.6.

See also [rtContextGetProgramFromId](#)

3.11.3.170 **RTresult** RTAPI [rtProgramGetVariable](#) ( **RTprogram** *program*, unsigned int *index*, **RTvariable** \* *v* )

Returns a handle to a variable attached to a program by index.

**Description**

[rtProgramGetVariable](#) returns a handle to a variable in \**v* attached to *program* with [rtProgramDeclareVariable](#) by *index*. *index* must be between 0 and one less than the value returned by [rtProgramGetVariableCount](#). The order in which variables are enumerated is not constant and may change as variables are attached and removed from the program object.

**Parameters**

in	<i>program</i>	The program to be queried for the indexed variable object
in	<i>index</i>	The index of the variable to return
out	<i>v</i>	Return handle to the variable object specified by the index

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

**History**

[rtProgramGetVariable](#) was introduced in OptiX 1.0.

See also [rtProgramDeclareVariable](#), [rtProgramRemoveVariable](#), [rtProgramGetVariableCount](#), [rtProgramQueryVariable](#)

3.11.3.171 **RTresult** RTAPI [rtProgramGetVariableCount](#) ( **RTprogram** *program*, unsigned int \* *count* )

Returns the number of variables attached to a program.

**Description**

[rtProgramGetVariableCount](#) returns, in \**count*, the number of variable objects that have been attached to *program*.

**Parameters**

in	<i>program</i>	The program to be queried for its variable count
out	<i>count</i>	The return handle for the number of variables attached to this program

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtProgramGetVariableCount](#) was introduced in OptiX 1.0.

**See also** [rtProgramDeclareVariable](#), [rtProgramRemoveVariable](#), [rtProgramGetVariable](#), [rtProgramQueryVariable](#)

3.11.3.172 **RTresult** RTAPI [rtProgramQueryVariable](#) ( **RTprogram** *program*, **const char \*** *name*, **RTvariable \*** *v* )

Returns a handle to the named variable attached to a program.

**Description**

[rtProgramQueryVariable](#) returns a handle to a variable object, in *\*v*, attached to *program* referenced by the *NULL-terminated* string *name*. If *name* is not the name of a variable attached to *program*, *\*v* will be *NULL* after the call.

**Parameters**

in	<i>program</i>	The program to be queried for the named variable
in	<i>name</i>	The name of the program to be queried for
out	<i>v</i>	The return handle to the variable object
	<i>program</i>	Handle to the program to be created

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtProgramQueryVariable](#) was introduced in OptiX 1.0.

**See also** [rtProgramDeclareVariable](#), [rtProgramRemoveVariable](#), [rtProgramGetVariable](#), [rtProgramGetVariableCount](#)

3.11.3.173 **RTresult** RTAPI [rtProgramRemoveVariable](#) ( **RTprogram** *program*, **RTvariable** *v* )

Removes the named variable from a program.

**Description**

[rtProgramRemoveVariable](#) removes variable *v* from the *program* object. Once a variable has been removed from this program, another variable with the same name as the removed variable may be declared.

## Parameters

in	<i>program</i>	The program to remove the variable from
in	<i>v</i>	The variable to remove

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

## History

[rtProgramRemoveVariable](#) was introduced in OptiX 1.0.

**See also** [rtProgramDeclareVariable](#), [rtProgramGetVariable](#), [rtProgramGetVariableCount](#), [rtProgramQueryVariable](#)

3.11.3.174 **RTresult** RTAPI **rtProgramValidate** ( **RTprogram** *program* )

Validates the state of a program.

## Description

[rtProgramValidate](#) checks *program* for completeness. If *program* or any of the objects attached to *program* are not valid, returns [RT\\_ERROR\\_INVALID\\_CONTEXT](#).

## Parameters

in	<i>program</i>	The program to be validated
----	----------------	-----------------------------

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtProgramValidate](#) was introduced in OptiX 1.0.

**See also** [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.11.3.175 **RTresult** RTAPI **rtRemoteDeviceCreate** ( **const char \****url*, **const char \****username*, **const char \****password*, **RTremotedevice \****remote\_dev* )

Create a device for remote rendering on VCAs.

## Description

Establishes a connection to a remote OptiX device, e.g. a VCA or cluster of VCAs. This opens a connection to the cluster manager software running at *address*, using *username* and *password* as authentication strings. *address* is a WebSocket URL of the form "ws://localhost:80" or "wss://localhost:443", *username* and *password* as plain text strings for authenticating on the remote device. If successful, it initializes a new [RTremotedevice](#) object.

In order to use this newly created remote device, a rendering instance needs to be configured by selecting a software configuration and reserving a number of nodes in the VCA. See [rtRemoteDeviceReserve](#) for more details.

After a rendering instance is properly initialized, a remote device must be associated with a context to be used. Calling `rtContextSetDevices` creates this association. Any further OptiX calls will be directed to the remote device.

**Parameters**

in	<i>url</i>	The WebSocket URL to connect to
in	<i>username</i>	Username in plain text
in	<i>password</i>	Password in plain text
out	<i>remote_dev</i>	A handle to the new remote device object

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_CONNECTION\\_FAILED](#)
- [RT\\_ERROR\\_AUTHENTICATION\\_FAILED](#)

**History**

[rtRemoteDeviceCreate](#) was introduced in OptiX 3.8.

**See also** [rtRemoteDeviceDestroy](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceReserve](#) [rtRemoteDeviceRelease](#) [rtContextSetRemoteDevice](#)

**3.11.3.176 RTresult RTAPI rtRemoteDeviceDestroy ( RTremotedevice *remote\_dev* )**

Destroys a remote device.

**Description**

Closes the network connection to the remote device and destroys the corresponding [RTremotedevice](#) object.

**Parameters**

in	<i>remote_dev</i>	The remote device object to destroy
----	-------------------	-------------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtRemoteDeviceDestroy](#) was introduced in OptiX 3.8.

**See also** [rtRemoteDeviceCreate](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceReserve](#) [rtRemoteDeviceRelease](#) [rtContextSetRemoteDevice](#)

**3.11.3.177 RTresult RTAPI rtRemoteDeviceGetAttribute ( RTremotedevice *remote\_dev*, RTremotedeviceattribute *attrib*, RTsize *size*, void \* *p* )**

Queries attributes of a remote device.

**Description**

In order to gather information about a remote device, several attributes can be queried through [rtRemoteDeviceGetAttribute](#).

Each attribute can have a different size. The sizes are given in the following list:

- [RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_CLUSTER\\_URL](#) size of provided destination buffer
- [RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_HEAD\\_NODE\\_URL](#) size of provided destination buffer

- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS` `sizeof(int)`
- `RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS` size of provided destination buffer
- `RT_REMOTEDEVICE_ATTRIBUTE_STATUS` `sizeof(RTremotedevicestatus)`
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES` `sizeof(int)`
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES` `sizeof(int)`
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES` `sizeof(int)`
- `RT_REMOTEDEVICE_ATTRIBUTE_NAME` size of provided destination buffer
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS` `sizeof(int)`
- `RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY` `sizeof(RTsize)`

The following attributes can be queried when a remote device is connected:

- `RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL`
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS`
- `RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS`
- `RT_REMOTEDEVICE_ATTRIBUTE_STATUS`
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES`
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES`
- `RT_REMOTEDEVICE_ATTRIBUTE_NAME`
- `RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY`

The following attributes require a valid reservation to be queried:

- `RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL`
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES`
- `RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS`

`RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL` The URL of the Cluster Manager associated with this remote device.

`RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL` The URL of the rendering instance being used, once it has been reserved and initialized.

`RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS` Number of compatible software configurations available in the remote device.

`RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS` Base entry for a list of compatible software configurations in the device. A configuration is a text description for a software package installed in the remote device, intended as a guide to the user in selecting from the pool of compatible configurations. This list is already filtered and it only contains entries on the remote device compatible with the client library being used. Each entry can be accessed as the attribute (`RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS + index`), with `index` being zero-based. The configuration description for the given `index` is copied into the destination buffer. A suggested size for the destination buffer is 256 characters. The number of entries in the list is given by the value of `RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS`. Only configurations compatible with the client version being used are listed.

`RT_REMOTEDEVICE_ATTRIBUTE_STATUS` Returns the current status of the remote device, as one of the following:

- [RT\\_REMOTEDEVICE\\_STATUS\\_READY](#) The remote device is ready for use.
- [RT\\_REMOTEDEVICE\\_STATUS\\_CONNECTED](#) The remote device is connected to a cluster manager, but no reservation exists.
- [RT\\_REMOTEDEVICE\\_STATUS\\_RESERVED](#) The remote device has a rendering instance reserved, but it is not yet ready.
- [RT\\_REMOTEDEVICE\\_STATUS\\_DISCONNECTED](#) The remote device has disconnected.

[RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_NUM\\_TOTAL\\_NODES](#) Total number of nodes in the cluster of VCAs.

[RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_NUM\\_FREE\\_NODES](#) Number of free nodes available.

[RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_NUM\\_RESERVED\\_NODES](#) Number of nodes used by the current reservation.

[RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_NUM\\_GPUS](#) Number of GPUs used by the current reservation.

[RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_NAME](#) Common name assigned the Remote Device.

[RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_GPU\\_TOTAL\\_MEMORY](#) Total amount of memory on each GPU, in bytes.

#### Parameters

<code>in</code>	<code>remote_dev</code>	The remote device to query
-----------------	-------------------------	----------------------------

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtRemoteDeviceGetAttribute](#) was introduced in OptiX 3.8.

**See also** [rtRemoteDeviceCreate](#) [rtRemoteDeviceReserve](#) [rtRemoteDeviceRelease](#) [rtContextSetRemoteDevice](#)

3.11.3.178 **RTresult** RTAPI `rtRemoteDeviceRelease ( RTremotedevice remote_dev )`

Release reserved nodes on a remote device.

#### Description

Releases an existing reservation on the remote device. The rendering instance on the remote device is destroyed, and all its remote context information is lost. Further OptiX calls will no longer be directed to the device. A new reservation can take place.

#### Parameters

<code>in</code>	<code>remote_dev</code>	The remote device on which the reservation was made
-----------------	-------------------------	---

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtRemoteDeviceRelease](#) was introduced in OptiX 3.8.

**See also** [rtRemoteDeviceCreate](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceReserve](#) [rtContextSetRemoteDevice](#)



### 3.11.3.179 `RTresult RTAPI rtRemoteDeviceReserve ( RTremotedevice remote_dev, unsigned int num_nodes, unsigned int configuration )`

Reserve nodes for rendering on a remote device.

#### Description

Reserves nodes in the remote device to form a rendering instance. Receives *num\_nodes* as the number of nodes to reserve, and *configuration* as the index of the software package to use for the created instance. Both the number of available nodes and the list of available configurations in a remote device can be retrieved by [rtRemoteDeviceGetAttribute](#).

After successfully reserving the nodes, the [RT\\_REMOTEDEVICE\\_ATTRIBUTE\\_STATUS](#) attribute should be polled repeatedly. The rendering instance is ready for use when that attribute is set to `RT_REMOTE_DEVICE_STATUS_READY`.

Only a single reservation per remote device and user can exist at any given time (i.e. a user can have only one rendering instance per remote device). This includes reservations performed through other means, like previous runs that were not properly released, or manual reservations over the cluster manager web interface.

#### Parameters

in	<i>remote_dev</i>	The remote device on which to reserve nodes
in	<i>num_nodes</i>	The number of nodes to reserve
in	<i>configuration</i>	The index of the software configuration to use

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtRemoteDeviceReserve](#) was introduced in OptiX 3.8.

**See also** [rtRemoteDeviceCreate](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceRelease](#) [rtContextSetRemoteDevice](#)

### 3.11.3.180 `RTresult RTAPI rtSelectorCreate ( RTcontext context, RTselector * selector )`

Creates a Selector node.

#### Description

Creates a new Selector node within *context*. After calling [rtSelectorCreate](#) the new node is in a "raw" state. For the node to be functional, a visit program must be assigned using [rtSelectorSetVisitProgram](#). Furthermore, a number of (zero or more) children can be attached by using [rtSelectorSetChildCount](#) and [rtSelectorSetChild](#). Sets *\*selector* to the handle of a newly created selector within *context*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *selector* is `NULL`.

#### Parameters

in	<i>context</i>	Specifies the rendering context of the Selector node
out	<i>selector</i>	New Selector node handle

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtSelectorCreate](#) was introduced in OptiX 1.0.

**See also** [rtSelectorDestroy](#), [rtSelectorValidate](#), [rtSelectorGetContext](#), [rtSelectorSetVisitProgram](#), [rtSelectorSetChildCount](#), [rtSelectorSetChild](#)

3.11.3.181 **RTresult** RTAPI [rtSelectorDeclareVariable](#) ( **RTselector** *selector*, **const char \*** *name*, **RTvariable \*** *v* )

Declares a variable associated with a Selector node.

## Description

Declares a new variable identified by *name*, and associates it with the Selector node *selector*. The new variable handle is returned in *v*. After declaration, a variable does not have a type until its value is set by an [rtVariableSet{...}](#) function. Once a variable type has been set, it cannot be changed, i.e., only [rtVariableSet{...}](#) functions of the same type can be used to change the value of the variable.

## Parameters

in	<i>selector</i>	Selector node handle
in	<i>name</i>	Variable identifier
out	<i>v</i>	New variable handle

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_REDECLARED](#)
- [RT\\_ERROR\\_ILLEGAL\\_SYMBOL](#)

## History

[rtSelectorDeclareVariable](#) was introduced in OptiX 1.0.

**See also** [rtSelectorQueryVariable](#), [rtSelectorRemoveVariable](#), [rtSelectorGetVariableCount](#), [rtSelectorGetVariable](#), [rtVariableSet{...}](#)

3.11.3.182 **RTresult** RTAPI [rtSelectorDestroy](#) ( **RTselector** *selector* )

Destroys a selector node.

## Description

[rtSelectorDestroy](#) removes *selector* from its context and deletes it. *selector* should be a value returned by [rtSelectorCreate](#). Associated variables declared via [rtSelectorDeclareVariable](#) are destroyed, but no child graph nodes are destroyed. After the call, *selector* is no longer a valid handle.

## Parameters

in	<i>selector</i>	Handle of the selector node to destroy
----	-----------------	--

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtSelectorDestroy](#) was introduced in OptiX 1.0.

**See also** [rtSelectorCreate](#), [rtSelectorValidate](#), [rtSelectorGetContext](#)

**3.11.3.183** **RTresult** RTAPI [rtSelectorGetChild](#) ( **RTselector** *selector*, unsigned int *index*, **RObject** \* *child* )

Returns a child node that is attached to a Selector node.

### Description

[rtSelectorGetChild](#) returns in *child* a handle of the child node currently attached to *selector* at slot *index*. The index value must be lower than the number previously set by [rtSelectorSetChildCount](#), thus it must be in the range from 0 to [rtSelectorGetChildCount](#) - 1. The returned pointer is of generic type [RObject](#) and needs to be cast to the actual child type, which can be [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), or [RTtransform](#). The actual type of *child* can be queried using [rtSelectorGetChildType](#);

### Parameters

in	<i>selector</i>	Selector node handle
in	<i>index</i>	Child node index
out	<i>child</i>	Child node handle. Can be { <a href="#">RTgroup</a> , <a href="#">RTselector</a> , <a href="#">RTgeometrygroup</a> , <a href="#">RTtransform</a> }

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtSelectorGetChild](#) was introduced in OptiX 1.0.

**See also** [rtSelectorSetChildCount](#), [rtSelectorGetChildCount](#), [rtSelectorSetChild](#), [rtSelectorGetChildType](#)

**3.11.3.184** **RTresult** RTAPI [rtSelectorGetChildCount](#) ( **RTselector** *selector*, unsigned int \* *count* )

Returns the number of child node slots of a Selector node.

### Description

[rtSelectorGetChildCount](#) returns in *count* the number of child node slots that have been previously reserved for the Selector node *selector* by [rtSelectorSetChildCount](#). The value of *count* does not reflect the actual number of child nodes that have so far been attached to the Selector node using [rtSelectorSetChild](#).

### Parameters

in	<i>selector</i>	Selector node handle
out	<i>count</i>	Number of child node slots reserved for <i>selector</i>

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtSelectorGetChildCount](#) was introduced in OptiX 1.0.

**See also** [rtSelectorSetChildCount](#), [rtSelectorSetChild](#), [rtSelectorGetChild](#), [rtSelectorGetChildType](#)

3.11.3.185 **RTresult** RTAPI [rtSelectorGetChildType](#) ( **RTselector** *selector*, unsigned int *index*, **RObjectType** \* *type* )

Returns type information about a Selector child node.

### Description

[rtSelectorGetChildType](#) queries the type of the child node attached to *selector* at slot *index*. If no child is associated with the given index, \**type* is set to [RT\\_OBJECTTYPE\\_UNKNOWN](#) and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if given a *NULL* pointer. The returned type is one of:

[RT\\_OBJECTTYPE\\_GROUP](#) [RT\\_OBJECTTYPE\\_GEOMETRY\\_GROUP](#) [RT\\_OBJECTTYPE\\_TRANSFORM](#) [RT\\_OBJECTTYPE\\_SELECTOR](#)

### Parameters

in	<i>selector</i>	Selector node handle
in	<i>index</i>	Child node index
out	<i>type</i>	Type of the child node

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtSelectorGetChildType](#) was introduced in OptiX 1.0.

**See also** [rtSelectorSetChildCount](#), [rtSelectorGetChildCount](#), [rtSelectorSetChild](#), [rtSelectorGetChild](#)

3.11.3.186 **RTresult** RTAPI [rtSelectorGetContext](#) ( **RTselector** *selector*, **RTcontext** \* *context* )

Returns the context of a Selector node.

### Description

[rtSelectorGetContext](#) returns in *context* the rendering context in which the Selector node *selector* has been created.

### Parameters

in	<i>selector</i>	Selector node handle
out	<i>context</i>	The context, <i>selector</i> belongs to

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtSelectorGetContext](#) was introduced in OptiX 1.0.

**See also** [rtSelectorCreate](#), [rtSelectorDestroy](#), [rtSelectorValidate](#)

3.11.3.187 **RTresult** RTAPI [rtSelectorGetVariable](#) ( **RTselector** *selector*, unsigned int *index*, **RTvariable** \* *v* )

Returns a variable associated with a Selector node.

## Description

Returns in *v* a handle to the variable located at position *index* in the Selectors's variable array. *index* is a sequential number depending on the order of variable declarations. The index must be in the range from 0 to [rtSelectorGetVariableCount](#) - 1. The current value of a variable can be retrieved from its handle by using an appropriate [rtVariableGet{...}](#) function matching the variable's type.

## Parameters

in	<i>selector</i>	Selector node handle
in	<i>index</i>	Variable index
out	<i>v</i>	Variable handle

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtSelectorGetVariable](#) was introduced in OptiX 1.0.

**See also** [rtSelectorDeclareVariable](#), [rtSelectorQueryVariable](#), [rtSelectorRemoveVariable](#), [rtSelectorGetVariableCount](#), [rtVariableGet{...}](#)

3.11.3.188 **RTresult** RTAPI [rtSelectorGetVariableCount](#) ( **RTselector** *selector*, unsigned int \* *count* )

Returns the number of variables attached to a Selector node.

## Description

[rtSelectorGetVariableCount](#) returns in *count* the number of variables that are currently attached to the Selector node *selector*.

## Parameters

in	<i>selector</i>	Selector node handle
out	<i>count</i>	Number of variables associated with <i>selector</i>

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtSelectorGetVariableCount](#) was introduced in OptiX 1.0.

**See also** [rtSelectorDeclareVariable](#), [rtSelectorQueryVariable](#), [rtSelectorRemoveVariable](#), [rtSelectorGetVariable](#)

3.11.3.189 **RTresult** RTAPI [rtSelectorGetVisitProgram](#) ( **RTselector** *selector*, **RTprogram** \* *program* )

Returns the currently assigned visit program.

#### Description

[rtSelectorGetVisitProgram](#) returns in *program* a handle of the visit program currently bound to *selector*.

#### Parameters

in	<i>selector</i>	Selector node handle
out	<i>program</i>	Current visit program assigned to <i>selector</i>

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtSelectorGetVisitProgram](#) was introduced in OptiX 1.0.

**See also** [rtSelectorSetVisitProgram](#)

3.11.3.190 **RTresult** RTAPI [rtSelectorQueryVariable](#) ( **RTselector** *selector*, **const char** \* *name*, **RTvariable** \* *v* )

Returns a variable associated with a Selector node.

#### Description

Returns in *v* a handle to the variable identified by *name*, which is associated with the Selector node *selector*. The current value of a variable can be retrieved from its handle by using an appropriate *rtVariableGet{...}* function matching the variable's type.

#### Parameters

in	<i>selector</i>	Selector node handle
in	<i>name</i>	Variable identifier
out	<i>v</i>	Variable handle

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtSelectorQueryVariable](#) was introduced in OptiX 1.0.

**See also** [rtSelectorDeclareVariable](#), [rtSelectorRemoveVariable](#), [rtSelectorGetVariableCount](#), [rtSelectorGetVariable](#), [rtVariableGet{...}](#)

3.11.3.191 `RTresult RTAPI rtSelectorRemoveVariable ( RTselector selector, RTvariable v )`

Removes a variable from a Selector node.

**Description**

[rtSelectorRemoveVariable](#) removes the variable *v* from the Selector node *selector* and deletes it. The handle *v* must be considered invalid afterwards.

**Parameters**

in	<i>selector</i>	Selector node handle
in	<i>v</i>	Variable handle

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_VARIABLE\\_NOT\\_FOUND](#)

**History**

[rtSelectorRemoveVariable](#) was introduced in OptiX 1.0.

**See also** [rtSelectorDeclareVariable](#), [rtSelectorQueryVariable](#), [rtSelectorGetVariableCount](#), [rtSelectorGetVariable](#)

3.11.3.192 `RTresult RTAPI rtSelectorSetChild ( RTselector selector, unsigned int index, RObject child )`

Attaches a child node to a Selector node.

**Description**

Attaches a new child node *child* to the parent node *selector*. *index* specifies the number of the slot where the child node gets attached. The index value must be lower than the number previously set by [rtSelectorSetChildCount](#), thus it must be in the range from 0 to [rtSelectorGetChildCount](#) - 1. Legal child node types are [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), and [RTtransform](#).

**Parameters**

in	<i>selector</i>	Selector node handle
in	<i>index</i>	Index of the parent slot the node <i>child</i> gets attached to
in	<i>child</i>	Child node to be attached. Can be { <a href="#">RTgroup</a> , <a href="#">RTselector</a> , <a href="#">RTgeometrygroup</a> , <a href="#">RTtransform</a> }

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtSelectorSetChild](#) was introduced in OptiX 1.0.

**See also** [rtSelectorSetChildCount](#), [rtSelectorGetChildCount](#), [rtSelectorGetChild](#), [rtSelectorGetChildType](#)

3.11.3.193 RTresult RTAPI rtSelectorSetChildCount ( RTselector *selector*, unsigned int *count* )

Specifies the number of child nodes to be attached to a Selector node.

**Description**

[rtSelectorSetChildCount](#) allocates a number of children slots, i.e., it pre-defines the exact number of child nodes the parent Selector node *selector* will have. Child nodes have to be attached to the Selector node using [rtSelectorSetChild](#). Empty slots will cause a validation error.

**Parameters**

in	<i>selector</i>	Selector node handle
in	<i>count</i>	Number of child nodes to be attached to <i>selector</i>

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtSelectorSetChildCount](#) was introduced in OptiX 1.0.

**See also** [rtSelectorValidate](#), [rtSelectorGetChildCount](#), [rtSelectorSetChild](#), [rtSelectorGetChild](#), [rtSelectorGetChildType](#)

3.11.3.194 RTresult RTAPI rtSelectorSetVisitProgram ( RTselector *selector*, RTprogram *program* )

Assigns a visit program to a Selector node.

**Description**

[rtSelectorSetVisitProgram](#) specifies a visit program that is executed when the Selector node *selector* gets visited by a ray during traversal of the model graph. A visit program steers how traversal of the Selectors's children is performed. It usually chooses only a single child to continue traversal, but is also allowed to process zero or multiple children. Programs can be created from PTX files using [rtProgramCreateFromPTXFile](#).

**Parameters**

in	<i>selector</i>	Selector node handle
in	<i>program</i>	Program handle associated with a visit program

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

**History**

[rtSelectorSetVisitProgram](#) was introduced in OptiX 1.0.

**See also** [rtSelectorGetVisitProgram](#), [rtProgramCreateFromPTXFile](#)



**3.11.3.195** `RTresult RTAPI rtSelectorValidate ( RTselector selector )`

Checks a Selector node for internal consistency.

**Description**

`rtSelectorValidate` recursively checks consistency of the Selector node *selector* and its children, i.e., it tries to validate the whole model sub-tree with *selector* as root. For a Selector node to be valid, it must be assigned a visit program, and the number of its children must match the number specified by `rtSelectorSetChildCount`.

**Parameters**

in	<i>selector</i>	Selector root node of a model sub-tree to be validated
----	-----------------	--

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`

**History**

`rtSelectorValidate` was introduced in OptiX 1.0.

**See also** `rtSelectorCreate`, `rtSelectorDestroy`, `rtSelectorGetContext`, `rtSelectorSetVisitProgram`, `rtSelectorSetChildCount`, `rtSelectorSetChild`

**3.11.3.196** `RTresult RTAPI rtTextureSamplerCreate ( RTcontext context, RTtexturesampler * texturesampler )`

Creates a new texture sampler object.

**Description**

`rtTextureSamplerCreate` allocates a texture sampler object. Sets *\*texturesampler* to the handle of a newly created texture sampler within *context*. Returns `RT_ERROR_INVALID_VALUE` if *texturesampler* is `NULL`.

**Parameters**

in	<i>context</i>	The context the texture sampler object will be created in
out	<i>texturesampler</i>	The return handle to the new texture sampler object

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`

**History**

`rtTextureSamplerCreate` was introduced in OptiX 1.0.

**See also** `rtTextureSamplerDestroy`

**3.11.3.197** `RTresult RTAPI rtTextureSamplerDestroy ( RTtexturesampler texturesampler )`

Destroys a texture sampler object.

**Description**

[`rtTextureSamplerDestroy`](#) removes *texturesampler* from its context and deletes it. *texturesampler* should be a value returned by [`rtTextureSamplerCreate`](#). After the call, *texturesampler* is no longer a valid handle. Any API object that referenced *texturesampler* will have its reference invalidated.

#### Parameters

in	<i>texturesampler</i>	Handle of the texture sampler to destroy
----	-----------------------	--

#### Return values

Relevant return values:

- [`RT\_SUCCESS`](#)
- [`RT\_ERROR\_INVALID\_CONTEXT`](#)
- [`RT\_ERROR\_INVALID\_VALUE`](#)
- [`RT\_ERROR\_MEMORY\_ALLOCATION\_FAILED`](#)

#### History

[`rtTextureSamplerDestroy`](#) was introduced in OptiX 1.0.

**See also** [`rtTextureSamplerCreate`](#)

3.11.3.198 **RTresult** RTAPI `rtTextureSamplerGetArraySize` ( **RTtexturesampler** *texturesampler*, unsigned int \* *num\_textures\_in\_array* )

Gets the number of array slices present in a texture sampler.

**Deprecated in OptiX 3.9**

#### Description

[`rtTextureSamplerGetArraySize`](#) gets the number of texture array slices in *texturesampler* and stores it in \**num\_textures\_in\_array*.

#### Parameters

in	<i>texturesampler</i>	The texture sampler object to be queried
out	<i>num_textures_in_array</i>	The return handle for the number of texture slices the texture sampler

#### Return values

Relevant return values:

- [`RT\_SUCCESS`](#)
- [`RT\_ERROR\_INVALID\_CONTEXT`](#)
- [`RT\_ERROR\_INVALID\_VALUE`](#)

#### History

[`rtTextureSamplerGetArraySize`](#) was introduced in OptiX 1.0.

**See also** [`rtTextureSamplerSetArraySize`](#)

3.11.3.199 **RTresult** RTAPI `rtTextureSamplerGetBuffer` ( **RTtexturesampler** *texturesampler*, unsigned int *deprecated0*, unsigned int *deprecated1*, **RTbuffer** \* *buffer* )

Gets a buffer object handle from a texture sampler.

**Deprecated**

#### Description

[`rtTextureSamplerGetBuffer`](#) gets a buffer object from *texturesampler* and stores it in \**buffer*.

## Parameters

in	<i>texturesampler</i>	The texture sampler object to be queried for the buffer
in	<i>deprecated0</i>	Must be 0
in	<i>deprecated1</i>	Must be 0
out	<i>buffer</i>	The return handle to the buffer attached to the texture sampler

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTextureSamplerGetBuffer](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerSetBuffer](#)

3.11.3.200 **RTresult** RTAPI `rtTextureSamplerGetContext` ( **RTtexturesampler** *texturesampler*, **RTcontext** \* *context* )

Gets the context object that created this texture sampler.

## Description

[rtTextureSamplerGetContext](#) returns a handle to the context object that was used to create *texturesampler*. If *context* is *NULL*, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

in	<i>texturesampler</i>	The texture sampler object to be queried for its context
out	<i>context</i>	The return handle for the context object of the texture sampler

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTextureSamplerGetContext](#) was introduced in OptiX 1.0.

**See also** [rtContextCreate](#)

3.11.3.201 **RTresult** RTAPI `rtTextureSamplerGetFilteringModes` ( **RTtexturesampler** *texturesampler*, **RTfiltermode** \* *minification*, **RTfiltermode** \* *magnification*, **RTfiltermode** \* *mipmapping* )

Gets the filtering modes of a texture sampler.

## Description

[rtTextureSamplerGetFilteringModes](#) gets the minification, magnification and MIP mapping filtering modes from *texturesampler* and stores them in \**minification*, \**magnification* and \**mipmapping*, respectively. See [rtTextureSamplerSetFilteringModes](#) for the values [RTfiltermode](#) may take.

**Parameters**

in	<i>texturesampler</i>	The texture sampler object to be queried
out	<i>minification</i>	The return handle for the minification filtering mode of the texture sampler
out	<i>magnification</i>	The return handle for the magnification filtering mode of the texture sampler
out	<i>mipmapping</i>	The return handle for the MIP mapping filtering mode of the texture sampler

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtTextureSamplerGetFilteringModes](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerSetFilteringModes](#)

### 3.11.3.202 **RTresult** RTAPI `rtTextureSamplerGetId ( RTtexturesampler texturesampler, int * texture_id )`

Returns the texture ID of this texture sampler.

**Description**

[rtTextureSamplerGetId](#) returns a handle to the texture sampler *texturesampler* to be used in OptiX programs on the device to reference the associated texture. The returned ID cannot be used on the host side. If *texture\_id* is *NULL*, returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

**Parameters**

in	<i>texturesampler</i>	The texture sampler object to be queried for its ID
out	<i>texture_id</i>	The returned device-side texture ID of the texture sampler

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtTextureSamplerGetId](#) was introduced in OptiX 3.0.

**See also** [rtTextureSamplerCreate](#)

### 3.11.3.203 **RTresult** RTAPI `rtTextureSamplerGetIndexingMode ( RTtexturesampler texturesampler, RTtextureindexmode * indexmode )`

Gets the indexing mode of a texture sampler.

**Description**

[rtTextureSamplerGetIndexingMode](#) gets the indexing mode of *texturesampler* and stores it in *\*indexmode*. See [rtTextureSamplerSetIndexingMode](#) for the values [RTtextureindexmode](#) may take.

**Parameters**

in	<i>texturesampler</i>	The texture sampler object to be queried
out	<i>indexmode</i>	The return handle for the indexing mode of the texture sampler

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtTextureSamplerGetIndexingMode](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerSetIndexingMode](#)

### 3.11.3.204 **RTresult** RTAPI `rtTextureSamplerGetMaxAnisotropy` ( **RTtexturesampler** *texturesampler*, float \* *value* )

Gets the maximum anisotropy level for a texture sampler.

**Description**

[rtTextureSamplerGetMaxAnisotropy](#) gets the maximum anisotropy level for *texturesampler* and stores it in \**value*.

**Parameters**

in	<i>texturesampler</i>	The texture sampler object to be queried
out	<i>value</i>	The return handle for the maximum anisotropy level of the texture sampler

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtTextureSamplerGetMaxAnisotropy](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerSetMaxAnisotropy](#)

### 3.11.3.205 **RTresult** RTAPI `rtTextureSamplerGetMipLevelBias` ( **RTtexturesampler** *texturesampler*, float \* *value* )

Gets the mipmap offset for a texture sampler.

**Description**

[rtTextureSamplerGetMipLevelBias](#) gets the mipmap offset for *texturesampler* and stores it in \**value*.

**Parameters**

in	<i>texturesampler</i>	The texture sampler object to be queried
out	<i>value</i>	The return handle for the mipmap offset of the texture sampler

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtTextureSamplerGetMipLevelBias](#) was introduced in OptiX 3.9.

**See also** [rtTextureSamplerSetMipLevelBias](#)

3.11.3.206 **RTresult** RTAPI [rtTextureSamplerGetMipLevelClamp](#) ( **RTtexturesampler** *texturesampler*, float \* *minLevel*, float \* *maxLevel* )

Gets the minimum and the maximum MIP level access range for a texture sampler.

## Description

[rtTextureSamplerGetMipLevelClamp](#) gets the minimum and the maximum MIP level access range for *texturesampler* and stores it in \**minLevel* and *maxLevel*.

## Parameters

in	<i>texturesampler</i>	The texture sampler object to be queried
out	<i>minLevel</i>	The return handle for the minimum mipmap level of the texture sampler
out	<i>maxLevel</i>	The return handle for the maximum mipmap level of the texture sampler

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtTextureSamplerGetMipLevelClamp](#) was introduced in OptiX 3.9.

**See also** [rtTextureSamplerSetMipLevelClamp](#)

3.11.3.207 **RTresult** RTAPI [rtTextureSamplerGetMipLevelCount](#) ( **RTtexturesampler** *texturesampler*, unsigned int \* *num\_mip\_levels* )

Gets the number of MIP levels in a texture sampler.

## Deprecated in OptiX 3.9

## Description

[rtTextureSamplerGetMipLevelCount](#) gets the number of MIP levels contained in *texturesampler* and stores it in \**num\_mip\_levels*.

## Parameters

in	<i>texturesampler</i>	The texture sampler object to be queried
out	<i>num_mip_levels</i>	The return handle for the number of MIP levels in the texture sampler

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

## History

[rtTextureSamplerGetMipLevelCount](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerSetMipLevelCount](#)

3.11.3.208 **RTresult** RTAPI `rtTextureSamplerGetReadMode` ( `RTtexturesampler` *texturesampler*, `RTtexturereadmode` \* *readmode* )

Gets the read mode of a texture sampler.

#### Description

`rtTextureSamplerGetReadMode` gets the read mode of *texturesampler* and stores it in \**readmode*. See `rtTextureSamplerSetReadMode` for a list of values `RTtexturereadmode` can take.

#### Parameters

in	<i>texturesampler</i>	The texture sampler object to be queried
out	<i>readmode</i>	The return handle for the read mode of the texture sampler

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

#### History

`rtTextureSamplerGetReadMode` was introduced in OptiX 1.0.

**See also** `rtTextureSamplerSetReadMode`

3.11.3.209 **RTresult** RTAPI `rtTextureSamplerGetWrapMode` ( `RTtexturesampler` *texturesampler*, unsigned int *dimension*, `RTwrapmode` \* *wrapmode* )

Gets the wrap mode of a texture sampler.

#### Description

`rtTextureSamplerGetWrapMode` gets the texture wrapping mode of *texturesampler* and stores it in \**wrapmode*. See `rtTextureSamplerSetWrapMode` for a list of values `RTwrapmode` can take.

#### Parameters

in	<i>texturesampler</i>	The texture sampler object to be queried
in	<i>dimension</i>	Dimension for the wrapping
out	<i>wrapmode</i>	The return handle for the wrap mode of the texture sampler

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

#### History

`rtTextureSamplerGetWrapMode` was introduced in OptiX 1.0.

**See also** `rtTextureSamplerSetWrapMode`

3.11.3.210 **RTresult** RTAPI `rtTextureSamplerSetArraySize` ( `RTtexturesampler` *texturesampler*, unsigned int *num\_textures\_in\_array* )

Sets the array size of a texture sampler.

**Deprecated in OptiX 3.9**

### Description

[rtTextureSamplerSetArraySize](#) specifies the number of texture array slices present in *texturesampler* as *num\_textures\_in\_array*. After changing the number of slices in the array, buffers must be reassociated with *texturesampler* via [rtTextureSamplerSetBuffer](#).

#### Parameters

in	<i>texturesampler</i>	The texture sampler object to be changed
in	<i>num_textures_in_array</i>	The new number of array slices of the texture sampler

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtTextureSamplerSetArraySize](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerGetArraySize](#)

3.11.3.211 **RTresult** RTAPI [rtTextureSamplerSetBuffer](#) ( **RTtexturesampler** *texturesampler*, unsigned int *deprecated0*, unsigned int *deprecated1*, **RTbuffer** *buffer* )

Attaches a buffer object to a texture sampler.

### Deprecated in OptiX 3.9

### Description

[rtTextureSamplerSetBuffer](#) attaches *buffer* to *texturesampler*.

#### Parameters

in	<i>texturesampler</i>	The texture sampler object that will contain the buffer
in	<i>deprecated0</i>	Must be 0
in	<i>deprecated1</i>	Must be 0
in	<i>buffer</i>	The buffer to be attached to the texture sampler

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

### History

[rtTextureSamplerSetBuffer](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerGetBuffer](#)

3.11.3.212 **RTresult** RTAPI [rtTextureSamplerSetFilteringModes](#) ( **RTtexturesampler** *texturesampler*, **RTfiltermode** *minification*, **RTfiltermode** *magnification*, **RTfiltermode** *mipmapping* )

Sets the filtering modes of a texture sampler.

### Description



`rtTextureSamplerSetFilteringModes` sets the minification, magnification and MIP mapping filter modes for *texture-sampler*. `RTfiltermode` must be one of the following values:

- `RT_FILTER_NEAREST`
- `RT_FILTER_LINEAR`
- `RT_FILTER_NONE`

These filter modes specify how the texture sampler will interpolate buffer data that has been attached to it. *minification* and *magnification* must be one of `RT_FILTER_NEAREST` or `RT_FILTER_LINEAR`. *mipmapping* may be any of the three values but must be `RT_FILTER_NONE` if the texture sampler contains only a single MIP level or one of `RT_FILTER_NEAREST` or `RT_FILTER_LINEAR` if the texture sampler contains more than one MIP level.

#### Parameters

in	<i>texturesampler</i>	The texture sampler object to be changed
in	<i>minification</i>	The new minification filter mode of the texture sampler
in	<i>magnification</i>	The new magnification filter mode of the texture sampler
in	<i>mipmapping</i>	The new MIP mapping filter mode of the texture sampler

#### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

#### History

`rtTextureSamplerSetFilteringModes` was introduced in OptiX 1.0.

See also `rtTextureSamplerGetFilteringModes`

3.11.3.213 **RTresult** RTAPI `rtTextureSamplerSetIndexingMode` ( `RTtexturesampler texturesampler`,  
`RTtextureindexmode indexmode` )

Sets whether texture coordinates for this texture sampler are normalized.

#### Description

`rtTextureSamplerSetIndexingMode` sets the indexing mode of *texturesampler* to *indexmode*. *indexmode* can take on one of the following values:

- `RT_TEXTURE_INDEX_NORMALIZED_COORDINATES`,
- `RT_TEXTURE_INDEX_ARRAY_INDEX`

These values are used to control the interpretation of texture coordinates. If the index mode is set to `RT_TEXTURE_INDEX_NORMALIZED_COORDINATES`, the texture is parameterized over [0,1]. If the index mode is set to `RT_TEXTURE_INDEX_ARRAY_INDEX` then texture coordinates are interpreted as array indices into the contents of the underlying buffer objects.

#### Parameters

in	<i>texturesampler</i>	The texture sampler object to be changed
in	<i>indexmode</i>	The new indexing mode of the texture sampler

#### Return values

Relevant return values:

- `RT_SUCCESS`

- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtTextureSamplerSetIndexingMode](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerGetIndexingMode](#)

#### 3.11.3.214 `RTresult RTAPI rtTextureSamplerSetMaxAnisotropy ( RTtexturesampler texturesampler, float value )`

Sets the maximum anisotropy of a texture sampler.

### Description

[rtTextureSamplerSetMaxAnisotropy](#) sets the maximum anisotropy of *texturesampler* to *value*. A float value specifies the maximum anisotropy ratio to be used when doing anisotropic filtering. This value will be clamped to the range [1,16]

### Parameters

<code>in</code>	<i>texturesampler</i>	The texture sampler object to be changed
<code>in</code>	<i>value</i>	The new maximum anisotropy level of the texture sampler

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtTextureSamplerSetMaxAnisotropy](#) was introduced in OptiX 1.0.

**See also** [rtTextureSamplerGetMaxAnisotropy](#)

#### 3.11.3.215 `RTresult RTAPI rtTextureSamplerSetMipLevelBias ( RTtexturesampler texturesampler, float value )`

Sets the mipmap offset of a texture sampler.

### Description

[rtTextureSamplerSetMipLevelBias](#) sets the offset to be applied to the calculated mipmap level.

### Parameters

<code>in</code>	<i>texturesampler</i>	The texture sampler object to be changed
<code>in</code>	<i>value</i>	The new mipmap offset of the texture sampler

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtTextureSamplerSetMipLevelBias](#) was introduced in OptiX 3.9.

**See also** [rtTextureSamplerGetMipLevelBias](#)

3.11.3.216 **RTresult** RTAPI `rtTextureSamplerSetMipLevelClamp` ( `RTtexturesampler` *texturesampler*, float *minLevel*, float *maxLevel* )

Sets the minimum and the maximum MIP level access range of a texture sampler.

#### Description

`rtTextureSamplerSetMipLevelClamp` sets lower end and the upper end of the MIP level range to clamp access to.

#### Parameters

in	<i>texturesampler</i>	The texture sampler object to be changed
in	<i>minLevel</i>	The new minimum mipmap level of the texture sampler
in	<i>maxLevel</i>	The new maximum mipmap level of the texture sampler

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

`rtTextureSamplerSetMipLevelClamp` was introduced in OptiX 3.9.

**See also** [rtTextureSamplerGetMipLevelClamp](#)

3.11.3.217 **RTresult** RTAPI `rtTextureSamplerSetMipLevelCount` ( `RTtexturesampler` *texturesampler*, unsigned int *num\_mip\_levels* )

Sets the number of MIP levels in a texture sampler.

#### Deprecated in OptiX 3.9

#### Description

`rtTextureSamplerSetMipLevelCount` sets the number of MIP levels in *texturesampler* to *num\_mip\_levels*.

#### Parameters

in	<i>texturesampler</i>	The texture sampler object to be changed
in	<i>num_mip_levels</i>	The new number of MIP levels of the texture sampler

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

`rtTextureSamplerSetMipLevelCount` was introduced in OptiX 1.0.

**See also** [rtTextureSamplerGetMipLevelCount](#)

3.11.3.218 **RTresult** RTAPI `rtTextureSamplerSetReadMode` ( `RTtexturesampler` *texturesampler*, `RTtexturereadmode` *readmode* )

Sets the read mode of a texture sampler.

## Description

`rtTextureSamplerSetReadMode` sets the data read mode of *texturesampler* to *readmode*. *readmode* can take one of the following values:

- `RT_TEXTURE_READ_ELEMENT_TYPE`
- `RT_TEXTURE_READ_NORMALIZED_FLOAT`
- `RT_TEXTURE_READ_ELEMENT_TYPE_SRGB`
- `RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB`

`RT_TEXTURE_READ_ELEMENT_TYPE_SRGB` and `RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB` were introduced in OptiX 3.9 and apply sRGB to linear conversion during texture read for 8-bit integer buffer formats. *readmode* controls the returned value of the texture sampler when it is used to sample textures. `RT_TEXTURE_READ_ELEMENT_TYPE` will return data of the type of the underlying buffer objects. `RT_TEXTURE_READ_NORMALIZED_FLOAT` will return floating point values normalized by the range of the underlying type. If the underlying type is floating point, `RT_TEXTURE_READ_NORMALIZED_FLOAT` and `RT_TEXTURE_READ_ELEMENT_TYPE` are equivalent, always returning the unmodified floating point value.

For example, a texture sampler that samples a buffer of type `RT_FORMAT_UNSIGNED_BYTE` with a read mode of `RT_TEXTURE_READ_NORMALIZED_FLOAT` will convert integral values from the range [0,255] to floating point values in the range [0,1] automatically as the buffer is sampled from.

### Parameters

<code>in</code>	<i>texturesampler</i>	The texture sampler object to be changed
<code>in</code>	<i>readmode</i>	The new read mode of the texture sampler

### Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

### History

`rtTextureSamplerSetReadMode` was introduced in OptiX 1.0.

See also `rtTextureSamplerGetReadMode`

3.11.3.219 **RTresult** RTAPI `rtTextureSamplerSetWrapMode` ( `RTtexturesampler texturesampler`, unsigned int *dimension*, `RTwrapmode wrapmode` )

Sets the wrapping mode of a texture sampler.

### Description

`rtTextureSamplerSetWrapMode` sets the wrapping mode of *texturesampler* to *wrapmode* for the texture dimension specified by *dimension*. *wrapmode* can take one of the following values:

- `RT_WRAP_REPEAT`
- `RT_WRAP_CLAMP_TO_EDGE`
- `RT_WRAP_MIRROR`
- `RT_WRAP_CLAMP_TO_BORDER`

The wrapping mode controls the behavior of the texture sampler as texture coordinates wrap around the range specified by the indexing mode. These values mirror the CUDA behavior of textures. See CUDA programming guide for details.

**Parameters**

<code>in</code>	<code>texturesampler</code>	The texture sampler object to be changed
<code>in</code>	<code>dimension</code>	Dimension of the texture
<code>in</code>	<code>wrapmode</code>	The new wrap mode of the texture sampler

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtTextureSamplerSetWrapMode](#) was introduced in OptiX 1.0. [RT\\_WRAP\\_MIRROR](#) and [RT\\_WRAP\\_CLAMP\\_TO\\_BORDER](#) were introduced in OptiX 3.0.

**See also** [rtTextureSamplerGetWrapMode](#)

### 3.11.3.220 `RTresult RTAPI rtTextureSamplerValidate ( RTtexturesampler texturesampler )`

Validates the state of a texture sampler.

**Description**

[rtTextureSamplerValidate](#) checks *texturesampler* for completeness. If *texturesampler* does not have buffers attached to all of its MIP levels and array slices or if the filtering modes are incompatible with the current MIP level and array slice configuration then returns [RT\\_ERROR\\_INVALID\\_CONTEXT](#).

**Parameters**

<code>in</code>	<code>texturesampler</code>	The texture sampler to be validated
-----------------	-----------------------------	-------------------------------------

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtTextureSamplerValidate](#) was introduced in OptiX 1.0.

**See also** [rtContextValidate](#)

### 3.11.3.221 `RTresult RTAPI rtTransformCreate ( RTcontext context, RTtransform * transform )`

Creates a new Transform node.

**Description**

Creates a new Transform node within the given context. For the node to be functional, a child node must be attached using [rtTransformSetChild](#). A transformation matrix can be associated with the transform node with [rtTransformSetMatrix](#). Sets *\*transform* to the handle of a newly created transform within *context*. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if *transform* is *NULL*.

## Parameters

in	<i>context</i>	Specifies the rendering context of the Transform node
out	<i>transform</i>	New Transform node handle

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTransformCreate](#) was introduced in OptiX 1.0.

**See also** [rtTransformDestroy](#), [rtTransformValidate](#), [rtTransformGetContext](#), [rtTransformSetMatrix](#), [rtTransformGetMatrix](#), [rtTransformSetChild](#), [rtTransformGetChild](#), [rtTransformGetChildType](#)

3.11.3.222 `RTresult RTAPI rtTransformDestroy ( RTtransform transform )`

Destroys a transform node.

## Description

[rtTransformDestroy](#) removes *transform* from its context and deletes it. *transform* should be a value returned by [rtTransformCreate](#). No child graph nodes are destroyed. After the call, *transform* is no longer a valid handle.

## Parameters

in	<i>transform</i>	Handle of the transform node to destroy
----	------------------	---

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTransformDestroy](#) was introduced in OptiX 1.0.

**See also** [rtTransformCreate](#), [rtTransformValidate](#), [rtTransformGetContext](#)

3.11.3.223 `RTresult RTAPI rtTransformGetChild ( RTtransform transform, RObject * child )`

Returns the child node that is attached to a Transform node.

## Description

[rtTransformGetChild](#) returns in *child* a handle of the child node currently attached to *transform*. The returned pointer is of generic type [RObject](#) and needs to be cast to the actual child type, which can be [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), or [RTtransform](#). The actual type of *child* can be queried using [rtTransformGetChildType](#). Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if given a *NULL* pointer.

## Parameters

in	<i>transform</i>	Transform node handle
out	<i>child</i>	Child node handle. Can be {RTgroup, RTselector, RTgeometrygroup, RTtransform}

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTransformGetChild](#) was introduced in OptiX 1.0.

**See also** [rtTransformSetChild](#), [rtTransformGetChildType](#)

3.11.3.224 **RTresult** RTAPI [rtTransformGetChildType](#) ( **RTtransform** *transform*, **RObjecttype** \* *type* )

Returns type information about a Transform child node.

## Description

[rtTransformGetChildType](#) queries the type of the child node attached to *transform*. If no child is attached, \**type* is set to [RT\\_OBJECTTYPE\\_UNKNOWN](#) and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if given a *NULL* pointer. The returned type is one of:

- [RT\\_OBJECTTYPE\\_GROUP](#)
- [RT\\_OBJECTTYPE\\_GEOMETRY\\_GROUP](#)
- [RT\\_OBJECTTYPE\\_TRANSFORM](#)
- [RT\\_OBJECTTYPE\\_SELECTOR](#)

## Parameters

in	<i>transform</i>	Transform node handle
out	<i>type</i>	Type of the child node

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTransformGetChildType](#) was introduced in OptiX 1.0.

**See also** [rtTransformSetChild](#), [rtTransformGetChild](#)

### 3.11.3.225 `RTresult RTAPI rtTransformGetContext ( RTtransform transform, RTcontext * context )`

Returns the context of a Transform node.

#### Description

[rtTransformGetContext](#) queries a transform node for its associated context. *transform* specifies the transform node to query, and should be a value returned by [rtTransformCreate](#). Sets \**context* to the context associated with *transform*.

#### Parameters

in	<i>transform</i>	Transform node handle
out	<i>context</i>	The context associated with <i>transform</i>

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtTransformGetContext](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformDestroy](#), [rtTransformValidate](#)

### 3.11.3.226 `RTresult RTAPI rtTransformGetMatrix ( RTtransform transform, int transpose, float * matrix, float * inverse_matrix )`

Returns the affine matrix and its inverse associated with a Transform node.

#### Description

[rtTransformGetMatrix](#) returns in *matrix* the affine matrix that is currently used to perform a transformation of the geometry contained in the sub-tree with *transform* as root. The corresponding inverse matrix will be returned in *inverse\_matrix*. One or both pointers are allowed to be *NULL*. If *transpose* is 0, matrices are returned in row-major format, i.e., matrix rows are contiguously laid out in memory. If *transpose* is non-zero, matrices are returned in column-major format. If non-*NULL*, matrix pointers must point to a float array of at least 16 elements.

#### Parameters

in	<i>transform</i>	Transform node handle
in	<i>transpose</i>	Flag indicating whether <i>matrix</i> and <i>inverse_matrix</i> should be transposed
out	<i>matrix</i>	Affine matrix (4x4 float array)
out	<i>inverse_matrix</i>	Inverted form of <i>matrix</i>

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

#### History

[rtTransformGetMatrix](#) was introduced in OptiX 1.0.

See also [rtTransformSetMatrix](#)



3.11.3.227 **RTresult** RTAPI `rtTransformSetChild ( RTtransform transform, RObject child )`

Attaches a child node to a Transform node.

**Description**

Attaches a child node *child* to the parent node *transform*. Legal child node types are [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), and [RTtransform](#). A transform node must have exactly one child. If a transformation matrix has been attached to *transform* with [rtTransformSetMatrix](#), it is effective on the model sub-tree with *child* as root node.

**Parameters**

in	<i>transform</i>	Transform node handle
in	<i>child</i>	Child node to be attached. Can be { <a href="#">RTgroup</a> , <a href="#">RTselector</a> , <a href="#">RTgeometrygroup</a> , <a href="#">RTtransform</a> }

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtTransformSetChild](#) was introduced in OptiX 1.0.

**See also** [rtTransformSetMatrix](#), [rtTransformGetChild](#), [rtTransformGetChildType](#)

3.11.3.228 **RTresult** RTAPI `rtTransformSetMatrix ( RTtransform transform, int transpose, const float * matrix, const float * inverse_matrix )`

Associates an affine transformation matrix with a Transform node.

**Description**

[rtTransformSetMatrix](#) associates a 4x4 matrix with the Transform node *transform*. The provided transformation matrix results in a corresponding affine transformation of all geometry contained in the sub-tree with *transform* as root. At least one of the pointers *matrix* and *inverse\_matrix* must be non-NULL. If exactly one pointer is valid, the other matrix will be computed. If both are valid, the matrices will be used as-is. If *transpose* is 0, source matrices are expected to be in row-major format, i.e., matrix rows are contiguously laid out in memory:

```
float matrix[4*4] = { a11, a12, a13, a14, a21, a22, a23, a24, a31, a32, a33, a34, a41, a42, a43, a44 };
```

Here, the translational elements *a14*, *a24*, and *a34* are at the 4th, 8th, and 12th position the matrix array. If the supplied matrices are in column-major format, a non-0 *transpose* flag can be used to trigger an automatic transpose of the input matrices.

**Parameters**

in	<i>transform</i>	Transform node handle
in	<i>transpose</i>	Flag indicating whether <i>matrix</i> and <i>inverse_matrix</i> should be transposed
in	<i>matrix</i>	Affine matrix (4x4 float array)
in	<i>inverse_matrix</i>	Inverted form of <i>matrix</i>

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTransformSetMatrix](#) was introduced in OptiX 1.0.

See also [rtTransformGetMatrix](#)

### 3.11.3.229 `RTresult RTAPI rtTransformValidate ( RTtransform transform )`

Checks a Transform node for internal consistency.

## Description

[rtTransformValidate](#) recursively checks consistency of the Transform node *transform* and its child, i.e., it tries to validate the whole model sub-tree with *transform* as root. For a Transform node to be valid, it must have a child node attached. It is, however, not required to explicitly set a transformation matrix. Without a specified transformation matrix, the identity matrix is applied.

## Parameters

<i>in</i>	<i>transform</i>	Transform root node of a model sub-tree to be validated
-----------	------------------	---

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

## History

[rtTransformValidate](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformDestroy](#), [rtTransformGetContext](#), [rtTransformSetMatrix](#), [rtTransformSetChild](#)

### 3.11.3.230 `RTresult RTAPI rtVariableGet1f ( RTvariable v, float * f1 )`

Functions designed to modify the value of a program variable.

## Description

`rtVariableGet` functions return the value of a program variable or variable array. The target variable is specified by *v*.

The commands `rtVariableGet{1-2-3-4}{f-i-ui}v` are used to query the value of a program variable specified by *v* using the pointers passed as arguments as return locations for each component of the vector-typed variable. The number specified in the command should match the number of components in the data type of the specified program variable (e.g., 1 for float, int, unsigned int; 2 for float2, int2, uint2, etc.). The suffix *f* indicates that floating-point values are expected to be returned, the suffix *i* indicates that integer values are expected, and the suffix *ui* indicates that unsigned integer values are expected, and this type should also match the data type of the specified program variable. The *f* variants of this function should be used to query values for program variables defined as float, float2, float3, float4, or arrays of these. The *i* variants of this function should be used to query values for program variables defined as int, int2, int3, int4, or arrays of these. The *ui* variants of this function should be used to query values for program variables defined as unsigned int, uint2, uint3, uint4, or arrays of these. The *v* variants of this function should be used to return the program variable's value to the array specified by parameter *v*. In this case, the array *v* should be large enough to accommodate all of the program variable's components.

The commands `rtVariableGetMatrix{2-3-4}x{2-3-4}fv` are used to query the value of a program variable whose data type is a matrix. The numbers in the command names are interpreted as the dimensionality of the matrix. For example, `2x4` indicates a 2 x 4 matrix with 2 columns and 4 rows (i.e., 8 values). If *transpose* is 0, the matrix is returned in row major order, otherwise in column major order.

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

rtVariableGet were introduced in OptiX 1.0.

**See also** [rtVariableSet](#), [rtVariableGetType](#), [rtContextDeclareVariable](#)

**Parameters**

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>f1</i>	Float value to be returned

**3.11.3.231 RTresult RTAPI rtVariableGet1fv ( RTvariable *v*, float \* *f* )****Parameters**

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>f</i>	Array of float value(s) to be returned

**3.11.3.232 RTresult RTAPI rtVariableGet1i ( RTvariable *v*, int \* *i1* )****Parameters**

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>i1</i>	Integer value to be returned

**3.11.3.233 RTresult RTAPI rtVariableGet1iv ( RTvariable *v*, int \* *i* )****Parameters**

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>i</i>	Array of integer values to be returned

**3.11.3.234 RTresult RTAPI rtVariableGet1ui ( RTvariable *v*, unsigned int \* *u1* )****Parameters**

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>u1</i>	Unsigned integer value to be returned

**3.11.3.235 RTresult RTAPI rtVariableGet1uiv ( RTvariable *v*, unsigned int \* *u* )****Parameters**

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>u</i>	Array of unsigned integer values to be returned

**3.11.3.236 RTresult RTAPI rtVariableGet2f ( RTvariable *v*, float \* *f1*, float \* *f2* )**

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>f1</i>	Float value to be returned
in	<i>f2</i>	Float value to be returned

3.11.3.237 **RTresult** RTAPI rtVariableGet2fv ( **RTvariable** *v*, float \* *f* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>f</i>	Array of float value(s) to be returned

3.11.3.238 **RTresult** RTAPI rtVariableGet2i ( **RTvariable** *v*, int \* *i1*, int \* *i2* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>i1</i>	Integer value to be returned
in	<i>i2</i>	Integer value to be returned

3.11.3.239 **RTresult** RTAPI rtVariableGet2iv ( **RTvariable** *v*, int \* *i* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>i</i>	Array of integer values to be returned

3.11.3.240 **RTresult** RTAPI rtVariableGet2ui ( **RTvariable** *v*, unsigned int \* *u1*, unsigned int \* *u2* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>u1</i>	Unsigned integer value to be returned
in	<i>u2</i>	Unsigned integer value to be returned

3.11.3.241 **RTresult** RTAPI rtVariableGet2uiv ( **RTvariable** *v*, unsigned int \* *u* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>u</i>	Array of unsigned integer values to be returned

3.11.3.242 **RTresult** RTAPI rtVariableGet3f ( **RTvariable** *v*, float \* *f1*, float \* *f2*, float \* *f3* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>f1</i>	Float value to be returned
in	<i>f2</i>	Float value to be returned
in	<i>f3</i>	Float value to be returned

3.11.3.243 **RTresult** RTAPI rtVariableGet3fv ( **RTvariable** *v*, float \* *f* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>f</i>	Array of float value(s) to be returned

3.11.3.244 RTresult RTAPI rtVariableGet3i ( RTvariable *v*, int \* *i1*, int \* *i2*, int \* *i3* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>i1</i>	Integer value to be returned
in	<i>i2</i>	Integer value to be returned
in	<i>i3</i>	Integer value to be returned

3.11.3.245 RTresult RTAPI rtVariableGet3iv ( RTvariable *v*, int \* *i* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>i</i>	Array of integer values to be returned

3.11.3.246 RTresult RTAPI rtVariableGet3ui ( RTvariable *v*, unsigned int \* *u1*, unsigned int \* *u2*, unsigned int \* *u3* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>u1</i>	Unsigned integer value to be returned
in	<i>u2</i>	Unsigned integer value to be returned
in	<i>u3</i>	Unsigned integer value to be returned

3.11.3.247 RTresult RTAPI rtVariableGet3uiv ( RTvariable *v*, unsigned int \* *u* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>u</i>	Array of unsigned integer values to be returned

3.11.3.248 RTresult RTAPI rtVariableGet4f ( RTvariable *v*, float \* *f1*, float \* *f2*, float \* *f3*, float \* *f4* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>f1</i>	Float value to be returned
in	<i>f2</i>	Float value to be returned
in	<i>f3</i>	Float value to be returned
in	<i>f4</i>	Float value to be returned

3.11.3.249 RTresult RTAPI rtVariableGet4fv ( RTvariable *v*, float \* *f* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>f</i>	Array of float value(s) to be returned

3.11.3.250 RTresult RTAPI rtVariableGet4i ( RTvariable *v*, int \* *i1*, int \* *i2*, int \* *i3*, int \* *i4* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>i1</i>	Integer value to be returned
in	<i>i2</i>	Integer value to be returned
in	<i>i3</i>	Integer value to be returned
in	<i>i4</i>	Integer value to be returned

3.11.3.251 `RTresult RTAPI rtVariableGet4iv ( RTvariable v, int * i )`

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>i</i>	Array of integer values to be returned

3.11.3.252 `RTresult RTAPI rtVariableGet4ui ( RTvariable v, unsigned int * u1, unsigned int * u2, unsigned int * u3, unsigned int * u4 )`

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>u1</i>	Unsigned integer value to be returned
in	<i>u2</i>	Unsigned integer value to be returned
in	<i>u3</i>	Unsigned integer value to be returned
in	<i>u4</i>	Unsigned integer value to be returned

3.11.3.253 `RTresult RTAPI rtVariableGet4uiv ( RTvariable v, unsigned int * u )`

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>u</i>	Array of unsigned integer values to be returned

3.11.3.254 `RTresult RTAPI rtVariableGetAnnotation ( RTvariable v, const char ** annotation_return )`

Queries the annotation string of a program variable.

## Description

`rtVariableGetAnnotation` queries a program variable's annotation string. A pointer to the string containing the annotation is returned in `*annotation_return`. If *v* is not a valid variable, this call sets `*annotation_return` to `NULL` and returns `RT_ERROR_INVALID_VALUE`. `*annotation_return` will point to valid memory until another API function that returns a string is called.

## Parameters

in	<i>v</i>	Specifies the program variable to be queried
out	<i>annotation_return</i>	Returns the program variable's annotation string

## Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`

## History

[rtVariableGetAnnotation](#) was introduced in OptiX 1.0.

**See also** [rtDeclareVariable](#), [rtDeclareAnnotation](#)

#### 3.11.3.255 `RTresult RTAPI rtVariableGetContext ( RTvariable v, RTcontext * context )`

Returns the context associated with a program variable.

##### Description

[rtVariableGetContext](#) queries the context associated with a program variable. The target variable is specified by `v`. The context of the program variable is returned to `*context` if the pointer `context` is not `NULL`. If `v` is not a valid variable, `*context` is set to `NULL` and [RT\\_ERROR\\_INVALID\\_VALUE](#) is returned.

##### Parameters

in	<code>v</code>	Specifies the program variable to be queried
out	<code>context</code>	Returns the context associated with the program variable

##### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

##### History

[rtVariableGetContext](#) was introduced in OptiX 1.0.

**See also** [rtContextDeclareVariable](#)

#### 3.11.3.256 `RTresult RTAPI rtVariableGetMatrix2x2fv ( RTvariable v, int transpose, float * m )`

##### Parameters

in	<code>v</code>	Specifies the program variable whose value is to be returned
in	<code>transpose</code>	Specify(ies) row-major or column-major order
in	<code>m</code>	Array of float values to be returned

#### 3.11.3.257 `RTresult RTAPI rtVariableGetMatrix2x3fv ( RTvariable v, int transpose, float * m )`

##### Parameters

in	<code>v</code>	Specifies the program variable whose value is to be returned
in	<code>transpose</code>	Specify(ies) row-major or column-major order
in	<code>m</code>	Array of float values to be returned

#### 3.11.3.258 `RTresult RTAPI rtVariableGetMatrix2x4fv ( RTvariable v, int transpose, float * m )`

##### Parameters

in	<code>v</code>	Specifies the program variable whose value is to be returned
in	<code>transpose</code>	Specify(ies) row-major or column-major order
in	<code>m</code>	Array of float values to be returned

#### 3.11.3.259 `RTresult RTAPI rtVariableGetMatrix3x2fv ( RTvariable v, int transpose, float * m )`

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>transpose</i>	Specify(ies) row-major or column-major order
in	<i>m</i>	Array of float values to be returned

3.11.3.260 RTresult RTAPI rtVariableGetMatrix3x3fv ( RTvariable *v*, int *transpose*, float \* *m* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>transpose</i>	Specify(ies) row-major or column-major order
in	<i>m</i>	Array of float values to be returned

3.11.3.261 RTresult RTAPI rtVariableGetMatrix3x4fv ( RTvariable *v*, int *transpose*, float \* *m* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>transpose</i>	Specify(ies) row-major or column-major order
in	<i>m</i>	Array of float values to be returned

3.11.3.262 RTresult RTAPI rtVariableGetMatrix4x2fv ( RTvariable *v*, int *transpose*, float \* *m* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>transpose</i>	Specify(ies) row-major or column-major order
in	<i>m</i>	Array of float values to be returned

3.11.3.263 RTresult RTAPI rtVariableGetMatrix4x3fv ( RTvariable *v*, int *transpose*, float \* *m* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>transpose</i>	Specify(ies) row-major or column-major order
in	<i>m</i>	Array of float values to be returned

3.11.3.264 RTresult RTAPI rtVariableGetMatrix4x4fv ( RTvariable *v*, int *transpose*, float \* *m* )

## Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>transpose</i>	Specify(ies) row-major or column-major order
in	<i>m</i>	Array of float values to be returned

3.11.3.265 RTresult RTAPI rtVariableGetName ( RTvariable *v*, const char \*\* *name\_return* )

Queries the name of a program variable.

## Description

Queries a program variable's name. The variable of interest is specified by *variable*, which should be a value returned by [rtContextDeclareVariable](#). A pointer to the string containing the name of the variable is returned in *\*name\_return*. If *v* is not a valid variable, this call sets *\*name\_return* to *NULL* and returns [RT\\_ERROR\\_INVALID\\_VALUE](#). *\*name\_return* will point to valid memory until another API function that returns a string is called.



**Parameters**

in	<i>v</i>	Specifies the program variable to be queried
out	<i>name_return</i>	Returns the program variable's name

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

**History**

[rtVariableGetName](#) was introduced in OptiX 1.0.

**See also** [rtContextDeclareVariable](#)

3.11.3.266 **RTresult** RTAPI [rtVariableGetObject](#) ( **RTvariable** *v*, **RtObject** \* *object* )

Returns the value of a OptiX object program variable.

**Description**

[rtVariableGetObject](#) queries the value of a program variable whose data type is a OptiX object. The target variable is specified by *v*. The value of the program variable is returned in \**object*. The concrete type of the program variable can be queried using [rtVariableGetType](#), and the **RtObject** handle returned by [rtVariableGetObject](#) may safely be cast to an OptiX handle of corresponding type. If *v* is not a valid variable, this call sets \**object* to *NULL* and returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

**Parameters**

in	<i>v</i>	Specifies the program variable to be queried
out	<i>object</i>	Returns the value of the program variable

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

**History**

[rtVariableGetObject](#) was introduced in OptiX 1.0.

**See also** [rtVariableSetObject](#), [rtVariableGetType](#), [rtContextDeclareVariable](#)

3.11.3.267 **RTresult** RTAPI [rtVariableGetSize](#) ( **RTvariable** *v*, **RTsize** \* *size* )

Queries the size, in bytes, of a variable.

**Description**

[rtVariableGetSize](#) queries a declared program variable for its size in bytes. This is most often used to query the size of a variable that has a user-defined type. Builtin types (int, float, unsigned int, etc.) may be queried, but object typed variables, such as buffers, texture samplers and graph nodes, cannot be queried and will return [RT\\_ERROR\\_INVALID\\_VALUE](#).

**Parameters**

<code>in</code>	<code>v</code>	Specifies the program variable to be queried
<code>out</code>	<code>size</code>	Specifies a pointer where the size of the variable, in bytes, will be returned

**Return values**

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

**History**

[rtVariableGetSize](#) was introduced in OptiX 1.0.

**See also** [rtVariableGetUserData](#), [rtContextDeclareVariable](#)

3.11.3.268 **RTresult** RTAPI `rtVariableGetType` ( **RTvariable** *v*, **RTOjecttype** \* *type\_return* )

Returns type information about a program variable.

**Description**

[rtVariableGetType](#) queries a program variable's type. The variable of interest is specified by *v*. The program variable's type enumeration is returned in \**type\_return*, if it is not *NULL*. It is one of the following:

- [RT\\_OBJECTTYPE\\_UNKNOWN](#)
- [RT\\_OBJECTTYPE\\_GROUP](#)
- [RT\\_OBJECTTYPE\\_GEOMETRY\\_GROUP](#)
- [RT\\_OBJECTTYPE\\_TRANSFORM](#)
- [RT\\_OBJECTTYPE\\_SELECTOR](#)
- [RT\\_OBJECTTYPE\\_GEOMETRY\\_INSTANCE](#)
- [RT\\_OBJECTTYPE\\_BUFFER](#)
- [RT\\_OBJECTTYPE\\_TEXTURE\\_SAMPLER](#)
- [RT\\_OBJECTTYPE\\_OBJECT](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT2x2](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT2x3](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT2x4](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT3x2](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT3x3](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT3x4](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT4x2](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT4x3](#)
- [RT\\_OBJECTTYPE\\_MATRIX\\_FLOAT4x4](#)
- [RT\\_OBJECTTYPE\\_FLOAT](#)
- [RT\\_OBJECTTYPE\\_FLOAT2](#)

- [RT\\_OBJECTTYPE\\_FLOAT3](#)
- [RT\\_OBJECTTYPE\\_FLOAT4](#)
- [RT\\_OBJECTTYPE\\_INT](#)
- [RT\\_OBJECTTYPE\\_INT2](#)
- [RT\\_OBJECTTYPE\\_INT3](#)
- [RT\\_OBJECTTYPE\\_INT4](#)
- [RT\\_OBJECTTYPE\\_UNSIGNED\\_INT](#)
- [RT\\_OBJECTTYPE\\_UNSIGNED\\_INT2](#)
- [RT\\_OBJECTTYPE\\_UNSIGNED\\_INT3](#)
- [RT\\_OBJECTTYPE\\_UNSIGNED\\_INT4](#)
- [RT\\_OBJECTTYPE\\_USER](#)

Sets *\*type\_return* to [RT\\_OBJECTTYPE\\_UNKNOWN](#) if *v* is not a valid variable. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if given a *NULL* pointer.

#### Parameters

in	<i>v</i>	Specifies the program variable to be queried
out	<i>type_return</i>	Returns the type of the program variable

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

#### History

[rtVariableGetType](#) was introduced in OptiX 1.0.

**See also** [rtContextDeclareVariable](#)

3.11.3.269 **RTresult** RTAPI [rtVariableGetUserData](#) ( **RTvariable** *v*, **RTsize** *size*, **void \****ptr* )

Defined.

#### Description

[rtVariableGetUserData](#) queries the value of a program variable whose data type is user-defined. The variable of interest is specified by *v*. The size of the variable's value must match the value given by the parameter *size*. The value of the program variable is copied to the memory region pointed to by *ptr*. The storage at location *ptr* must be large enough to accommodate all of the program variable's value data. If *v* is not a valid variable, this call has no effect and returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

#### Parameters

in	<i>v</i>	Specifies the program variable to be queried
in	<i>size</i>	Specifies the size of the program variable, in bytes
out	<i>ptr</i>	Location in which to store the value of the variable

#### Return values

Relevant return values:

- [RT\\_SUCCESS](#)

- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

[rtVariableGetUserData](#) was introduced in OptiX 1.0.

**See also** [rtVariableSetUserData](#), [rtContextDeclareVariable](#)

#### 3.11.3.270 `RTresult RTAPI rtVariableSet1f ( RTvariable v, float f1 )`

Functions designed to modify the value of a program variable.

### Description

`rtVariableSet` functions modify the value of a program variable or variable array. The target variable is specified by `v`, which should be a value returned by [rtContextGetVariable](#).

The commands `rtVariableSet{1-2-3-4}{f-i-ui}v` are used to modify the value of a program variable specified by `v` using the values passed as arguments. The number specified in the command should match the number of components in the data type of the specified program variable (e.g., 1 for float, int, unsigned int; 2 for float2, int2, uint2, etc.). The suffix `f` indicates that `v` has floating point type, the suffix `i` indicates that `v` has integral type, and the suffix `ui` indicates that `v` has unsigned integral type. The `v` variants of this function should be used to load the program variable's value from the array specified by parameter `v`. In this case, the array `v` should contain as many elements as there are program variable components.

The commands `rtVariableSetMatrix{2-3-4}x{2-3-4}fv` are used to modify the value of a program variable whose data type is a matrix. The numbers in the command names are the number of rows and columns, respectively. For example, `2x4` indicates a matrix with 2 rows and 4 columns (i.e., 8 values). If *transpose* is 0, the matrix is specified in row-major order, otherwise in column-major order or, equivalently, as a matrix with the number of rows and columns swapped in row-major order.

If `v` is not a valid variable, these calls have no effect and return [RT\\_ERROR\\_INVALID\\_VALUE](#)

### Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)

### History

`rtVariableSet` were introduced in OptiX 1.0.

**See also** [rtVariableGet](#), [rtVariableSet](#), [rtDeclareVariable](#)

#### Parameters

in	<code>v</code>	Specifies the program variable to be modified
in	<code>f1</code>	Specifies the new float value of the program variable

#### 3.11.3.271 `RTresult RTAPI rtVariableSet1fv ( RTvariable v, const float * f )`

#### Parameters

in	<code>v</code>	Specifies the program variable to be modified
----	----------------	---

in	<i>f</i>	Array of float values to set the variable to
----	----------	--

### 3.11.3.272 RTresult RTAPI rtVariableSet1i ( RTvariable *v*, int *i1* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>i1</i>	Specifies the new integer value of the program variable

### 3.11.3.273 RTresult RTAPI rtVariableSet1iv ( RTvariable *v*, const int \* *i* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>i</i>	Array of integer values to set the variable to

### 3.11.3.274 RTresult RTAPI rtVariableSet1ui ( RTvariable *v*, unsigned int *u1* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>u1</i>	Specifies the new unsigned integer value of the program variable

### 3.11.3.275 RTresult RTAPI rtVariableSet1uiv ( RTvariable *v*, const unsigned int \* *u* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>u</i>	Array of unsigned integer values to set the variable to

### 3.11.3.276 RTresult RTAPI rtVariableSet2f ( RTvariable *v*, float *f1*, float *f2* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>f1</i>	Specifies the new float value of the program variable
in	<i>f2</i>	Specifies the new float value of the program variable

### 3.11.3.277 RTresult RTAPI rtVariableSet2fv ( RTvariable *v*, const float \* *f* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>f</i>	Array of float values to set the variable to

### 3.11.3.278 RTresult RTAPI rtVariableSet2i ( RTvariable *v*, int *i1*, int *i2* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>i1</i>	Specifies the new integer value of the program variable
in	<i>i2</i>	Specifies the new integer value of the program variable

### 3.11.3.279 RTresult RTAPI rtVariableSet2iv ( RTvariable *v*, const int \* *i* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>i</i>	Array of integer values to set the variable to

**3.11.3.280** **RTresult** RTAPI rtVariableSet2ui ( **RTvariable** *v*, unsigned int *u1*, unsigned int *u2* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>u1</i>	Specifies the new unsigned integer value of the program variable
in	<i>u2</i>	Specifies the new unsigned integer value of the program variable

**3.11.3.281** **RTresult** RTAPI rtVariableSet2uiv ( **RTvariable** *v*, const unsigned int \* *u* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>u</i>	Array of unsigned integer values to set the variable to

**3.11.3.282** **RTresult** RTAPI rtVariableSet3f ( **RTvariable** *v*, float *f1*, float *f2*, float *f3* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>f1</i>	Specifies the new float value of the program variable
in	<i>f2</i>	Specifies the new float value of the program variable
in	<i>f3</i>	Specifies the new float value of the program variable

**3.11.3.283** **RTresult** RTAPI rtVariableSet3fv ( **RTvariable** *v*, const float \* *f* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>f</i>	Array of float values to set the variable to

**3.11.3.284** **RTresult** RTAPI rtVariableSet3i ( **RTvariable** *v*, int *i1*, int *i2*, int *i3* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>i1</i>	Specifies the new integer value of the program variable
in	<i>i2</i>	Specifies the new integer value of the program variable
in	<i>i3</i>	Specifies the new integer value of the program variable

**3.11.3.285** **RTresult** RTAPI rtVariableSet3iv ( **RTvariable** *v*, const int \* *i* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>i</i>	Array of integer values to set the variable to

**3.11.3.286** **RTresult** RTAPI rtVariableSet3ui ( **RTvariable** *v*, unsigned int *u1*, unsigned int *u2*, unsigned int *u3* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>u1</i>	Specifies the new unsigned integer value of the program variable
in	<i>u2</i>	Specifies the new unsigned integer value of the program variable
in	<i>u3</i>	Specifies the new unsigned integer value of the program variable

3.11.3.287 RTresult RTAPI rtVariableSet3uiv ( RTvariable *v*, const unsigned int \* *u* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>u</i>	Array of unsigned integer values to set the variable to

3.11.3.288 RTresult RTAPI rtVariableSet4f ( RTvariable *v*, float *f1*, float *f2*, float *f3*, float *f4* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>f1</i>	Specifies the new float value of the program variable
in	<i>f2</i>	Specifies the new float value of the program variable
in	<i>f3</i>	Specifies the new float value of the program variable
in	<i>f4</i>	Specifies the new float value of the program variable

3.11.3.289 RTresult RTAPI rtVariableSet4fv ( RTvariable *v*, const float \* *f* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>f</i>	Array of float values to set the variable to

3.11.3.290 RTresult RTAPI rtVariableSet4i ( RTvariable *v*, int *i1*, int *i2*, int *i3*, int *i4* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>i1</i>	Specifies the new integer value of the program variable
in	<i>i2</i>	Specifies the new integer value of the program variable
in	<i>i3</i>	Specifies the new integer value of the program variable
in	<i>i4</i>	Specifies the new integer value of the program variable

3.11.3.291 RTresult RTAPI rtVariableSet4iv ( RTvariable *v*, const int \* *i* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>i</i>	Array of integer values to set the variable to

3.11.3.292 RTresult RTAPI rtVariableSet4ui ( RTvariable *v*, unsigned int *u1*, unsigned int *u2*, unsigned int *u3*, unsigned int *u4* )

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
----	----------	---

in	<i>u1</i>	Specifies the new unsigned integer value of the program variable
in	<i>u2</i>	Specifies the new unsigned integer value of the program variable
in	<i>u3</i>	Specifies the new unsigned integer value of the program variable
in	<i>u4</i>	Specifies the new unsigned integer value of the program variable

### 3.11.3.293 RTresult RTAPI rtVariableSet4uiv ( RTvariable *v*, const unsigned int \* *u* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>u</i>	Array of unsigned integer values to set the variable to

### 3.11.3.294 RTresult RTAPI rtVariableSetMatrix2x2fv ( RTvariable *v*, int *transpose*, const float \* *m* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

### 3.11.3.295 RTresult RTAPI rtVariableSetMatrix2x3fv ( RTvariable *v*, int *transpose*, const float \* *m* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

### 3.11.3.296 RTresult RTAPI rtVariableSetMatrix2x4fv ( RTvariable *v*, int *transpose*, const float \* *m* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

### 3.11.3.297 RTresult RTAPI rtVariableSetMatrix3x2fv ( RTvariable *v*, int *transpose*, const float \* *m* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

### 3.11.3.298 RTresult RTAPI rtVariableSetMatrix3x3fv ( RTvariable *v*, int *transpose*, const float \* *m* )

#### Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

### 3.11.3.299 RTresult RTAPI rtVariableSetMatrix3x4fv ( RTvariable *v*, int *transpose*, const float \* *m* )



## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

3.11.3.300 `RTresult RTAPI rtVariableSetMatrix4x2fv ( RTvariable v, int transpose, const float * m )`

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

3.11.3.301 `RTresult RTAPI rtVariableSetMatrix4x3fv ( RTvariable v, int transpose, const float * m )`

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

3.11.3.302 `RTresult RTAPI rtVariableSetMatrix4x4fv ( RTvariable v, int transpose, const float * m )`

## Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>transpose</i>	Specifies row-major or column-major order
in	<i>m</i>	Array of float values to set the matrix to

3.11.3.303 `RTresult RTAPI rtVariableSetObject ( RTvariable v, RObject object )`

Sets a program variable value to a OptiX object.

## Description

`rtVariableSetObject` sets a program variable to an OptiX object value. The target variable is specified by *v*. The new value of the program variable is specified by *object*. The concrete type of *object* can be one of [RTbuffer](#), [RTtexturesampler](#), [RTgroup](#), [RTprogram](#), [RTselector](#), [RTgeometrygroup](#), or [RTtransform](#). If *v* is not a valid variable or *object* is not a valid OptiX object, this call has no effect and returns [RT\\_ERROR\\_INVALID\\_VALUE](#).

## Parameters

in	<i>v</i>	Specifies the program variable to be set
in	<i>object</i>	Specifies the new value of the program variable

## Return values

Relevant return values:

- [RT\\_SUCCESS](#)
- [RT\\_ERROR\\_INVALID\\_CONTEXT](#)
- [RT\\_ERROR\\_INVALID\\_VALUE](#)
- [RT\\_ERROR\\_TYPE\\_MISMATCH](#)

## History

`rtVariableSetObject` was introduced in OptiX 1.0. The ability to bind an [RTprogram](#) to a variable was introduced in OptiX 3.0.

See also [rtVariableGetObject](#), [rtContextDeclareVariable](#)

3.11.3.304 `RTresult RTAPI rtVariableSetUserData ( RTvariable v, RTsize size, const void * ptr )`

Defined.

**Description**

`rtVariableSetUserData` modifies the value of a program variable whose data type is user-defined. The value copied into the variable is defined by an arbitrary region of memory, pointed to by *ptr*. The size of the memory region is given by *size*. The target variable is specified by *v*. If *v* is not a valid variable, this call has no effect and returns `RT_ERROR_INVALID_VALUE`.

**Parameters**

in	<i>v</i>	Specifies the program variable to be modified
in	<i>size</i>	Specifies the size of the new value, in bytes
in	<i>ptr</i>	Specifies a pointer to the new value of the program variable

**Return values**

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`
- `RT_ERROR_TYPE_MISMATCH`

**History**

`rtVariableSetUserData` was introduced in OptiX 1.0.

**See also** `rtVariableGetUserData`, `rtContextDeclareVariable`

3.12 `optix_prime.h` File Reference

## 3.12.1 Detailed Description

OptiX Prime public API.

**Author**

NVIDIA Corporation OptiX Prime public API

**Macros**

- `#define OPTIX_PRIME_VERSION`
- `#define RTPAPI __declspec(dllimport)`

**Typedefs**

- `typedef unsigned int RTPsize`
- `typedef struct RTPcontext_api * RTPcontext`
- `typedef struct RTPmodel_api * RTPmodel`
- `typedef struct RTPquery_api * RTPquery`
- `typedef struct RTPbufferdesc_api * RTPbufferdesc`
- `typedef struct CUstream_st * cudaStream_t`

## Functions

- [RTPresult](#) RTPAPI [rtpContextCreate](#) ([RTPcontexttype](#) type, [RTPcontext](#) \*context)
- [RTPresult](#) RTPAPI [rtpContextSetCudaDeviceNumbers](#) ([RTPcontext](#) context, unsigned deviceCount, const unsigned \*deviceNumbers)
- [RTPresult](#) RTPAPI [rtpContextSetCpuThreads](#) ([RTPcontext](#) context, unsigned numThreads)
- [RTPresult](#) RTPAPI [rtpContextDestroy](#) ([RTPcontext](#) context)
- [RTPresult](#) RTPAPI [rtpContextGetLastErrorString](#) ([RTPcontext](#) context, const char \*\*return\_string)
- [RTPresult](#) RTPAPI [rtpBufferDescCreate](#) ([RTPcontext](#) context, [RTPbufferformat](#) format, [RTPbuffertype](#) type, void \*buffer, [RTPbufferdesc](#) \*desc)
- [RTPresult](#) RTPAPI [rtpBufferDescGetContext](#) ([RTPbufferdesc](#) desc, [RTPcontext](#) \*context)
- [RTPresult](#) RTPAPI [rtpBufferDescSetRange](#) ([RTPbufferdesc](#) desc, RTPsize begin, RTPsize end)
- [RTPresult](#) RTPAPI [rtpBufferDescSetStride](#) ([RTPbufferdesc](#) desc, unsigned strideBytes)
- [RTPresult](#) RTPAPI [rtpBufferDescSetCudaDeviceNumber](#) ([RTPbufferdesc](#) desc, unsigned deviceNumber)
- [RTPresult](#) RTPAPI [rtpBufferDescDestroy](#) ([RTPbufferdesc](#) desc)
- [RTPresult](#) RTPAPI [rtpModelCreate](#) ([RTPcontext](#) context, [RTPmodel](#) \*model)
- [RTPresult](#) RTPAPI [rtpModelGetContext](#) ([RTPmodel](#) model, [RTPcontext](#) \*context)
- [RTPresult](#) RTPAPI [rtpModelSetTriangles](#) ([RTPmodel](#) model, [RTPbufferdesc](#) indices, [RTPbufferdesc](#) vertices)
- [RTPresult](#) RTPAPI [rtpModelSetInstances](#) ([RTPmodel](#) model, [RTPbufferdesc](#) instances, [RTPbufferdesc](#) transforms)
- [RTPresult](#) RTPAPI [rtpModelUpdate](#) ([RTPmodel](#) model, unsigned hints)
- [RTPresult](#) RTPAPI [rtpModelFinish](#) ([RTPmodel](#) model)
- [RTPresult](#) RTPAPI [rtpModelGetFinished](#) ([RTPmodel](#) model, int \*isFinished)
- [RTPresult](#) RTPAPI [rtpModelCopy](#) ([RTPmodel](#) model, [RTPmodel](#) srcModel)
- [RTPresult](#) RTPAPI [rtpModelSetBuilderParameter](#) ([RTPmodel](#) model\_api, [RTPbuilderparam](#) param, RTPsize size, const void \*ptr)
- [RTPresult](#) RTPAPI [rtpModelDestroy](#) ([RTPmodel](#) model)
- [RTPresult](#) RTPAPI [rtpQueryCreate](#) ([RTPmodel](#) model, [RTPquerytype](#) queryType, [RTPquery](#) \*query)
- [RTPresult](#) RTPAPI [rtpQueryGetContext](#) ([RTPquery](#) query, [RTPcontext](#) \*context)
- [RTPresult](#) RTPAPI [rtpQuerySetRays](#) ([RTPquery](#) query, [RTPbufferdesc](#) rays)
- [RTPresult](#) RTPAPI [rtpQuerySetHits](#) ([RTPquery](#) query, [RTPbufferdesc](#) hits)
- [RTPresult](#) RTPAPI [rtpQueryExecute](#) ([RTPquery](#) query, unsigned hints)
- [RTPresult](#) RTPAPI [rtpQueryFinish](#) ([RTPquery](#) query)
- [RTPresult](#) RTPAPI [rtpQueryGetFinished](#) ([RTPquery](#) query, int \*isFinished)
- [RTPresult](#) RTPAPI [rtpQuerySetCudaStream](#) ([RTPquery](#) query, cudaStream\_t stream)
- [RTPresult](#) RTPAPI [rtpQueryDestroy](#) ([RTPquery](#) query)
- [RTPresult](#) RTPAPI [rtpHostBufferLock](#) (void \*buffer, RTPsize size)
- [RTPresult](#) RTPAPI [rtpHostBufferUnlock](#) (void \*buffer)
- [RTPresult](#) RTPAPI [rtpGetErrorString](#) ([RTPresult](#) errorCode, const char \*\*errorString)
- [RTPresult](#) RTPAPI [rtpGetVersion](#) (unsigned \*version)
- [RTPresult](#) RTPAPI [rtpGetVersionString](#) (const char \*\*versionString)

## 3.12.2 Macro Definition Documentation

## 3.12.2.1 #define OPTIX\_PRIME\_VERSION

## Value:

```
3090 /* 3.9.0 (major = OPTIX_PRIME_VERSION/1000,      *
          *          minor = (OPTIX_PRIME_VERSION%1000)/10, *
          *          micro = OPTIX_PRIME_VERSION%10      */
```

## 3.12.3 Typedef Documentation

3.12.3.1 `typedef struct RTPbufferdesc_api* RTPbufferdesc`

Opaque type. Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged.

3.12.3.2 `typedef struct RTPcontext_api* RTPcontext`

Opaque type. Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged.

3.12.3.3 `typedef struct RTPmodel_api* RTPmodel`

Opaque type. Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged.

3.12.3.4 `typedef struct RTPquery_api* RTPquery`

Opaque type. Note that the `*_api` type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged.

## 3.12.4 Function Documentation

3.12.4.1 **RTPResult RTPAPI `rtpBufferDescCreate` ( `RTPcontext context`, `RTPbufferformat format`, `RTPbuffertype type`, `void * buffer`, `RTPbufferdesc * desc` )**

Create a buffer descriptor.

This function creates a buffer descriptor with the specified element format and buffertype. A buffer of type `RTP_BUFFER_TYPE_CUDA_LINEAR` is assumed to reside on the current device. The device number can be changed by calling `rtpBufferDescSetCudaDeviceNumber`.

## Parameters

in	<i>context</i>	OptiX Prime context
in	<i>format</i>	Format of the buffer
in	<i>type</i>	Type of the buffer
in	<i>buffer</i>	Pointer to buffer data
out	<i>desc</i>	Pointer to the new buffer descriptor

## Return values

Relevant return values:

- `RTP_SUCCESS`
- `RTP_ERROR_INVALID_VALUE`
- `RTP_ERROR_UNKNOWN`

Example Usage:

```
RTPbufferdesc verticesBD;
rtpBufferDescCreate(context, RTP_BUFFER_FORMAT_VERTEX_FLOAT3
, RTP_BUFFER_TYPE_HOST, vertices, &verticesBD);
```

3.12.4.2 **RTPResult RTPAPI `rtpBufferDescDestroy` ( `RTPbufferdesc desc` )**

Destroys a buffer descriptor.

Buffer descriptors can be destroyed immediately after it is used as a function parameter. The buffer contents associated with a buffer descriptor, however, must remain valid until they are no longer used by any OptiX Prime objects.

## Parameters

<i>in</i>	<i>desc</i>	Buffer descriptor
-----------	-------------	-------------------

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.12.4.3 `RTPResult RTPAPI rtpBufferDescGetContext ( RTPbufferdesc desc, RTPcontext * context )`

Gets the context object associated with the provided buffer descriptor.

## Parameters

<i>in</i>	<i>desc</i>	Buffer descriptor
<i>out</i>	<i>context</i>	Returned context

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.12.4.4 `RTPResult RTPAPI rtpBufferDescSetCudaDeviceNumber ( RTPbufferdesc desc, unsigned deviceNumber )`

Sets the CUDA device number for a buffer.

A buffer of type [RTP\\_BUFFER\\_TYPE\\_CUDA\\_LINEAR](#) is assumed to reside on the device that was current when its buffer descriptor was created unless otherwise specified using this function.

## Parameters

<i>in</i>	<i>desc</i>	Buffer descriptor
<i>in</i>	<i>deviceNumber</i>	CUDA device number

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.12.4.5 `RTPResult RTPAPI rtpBufferDescSetRange ( RTPbufferdesc desc, RTPsize begin, RTPsize end )`

Sets the element range of a buffer to use.

The range is specified in terms of number of elements. By default, the range for a buffer is 0 to the number of elements in the buffer.

## Parameters

in	<i>desc</i>	Buffer descriptor
in	<i>begin</i>	Start index of the range
in	<i>end</i>	End index of the range (exclusive, one past the index of the last element)

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

#### 3.12.4.6 RTPResult RTPAPI `rtpBufferDescSetStride` ( RTPbufferdesc *desc*, unsigned *strideBytes* )

Sets the stride for elements in a buffer.

This function is only valid for buffers of format [RTP\\_BUFFER\\_FORMAT\\_VERTEX\\_FLOAT3](#). This function is useful for vertex buffers that contain interleaved vertex attributes. For buffers that are transferred between the host and a device it is recommended that only buffers with default stride be used to avoid transferring data that will not be used.

## Parameters

in	<i>desc</i>	Buffer descriptor
in	<i>strideBytes</i>	Stride in bytes. The default value of 0 indicates that elements are contiguous in memory.

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

Example Usage:

```
struct Vertex {
    float3 pos, normal, color;
};
...
RTPbufferdesc vertsBD;
rtpBufferDescCreate(context, RTP_BUFFER_FORMAT_VERTEX_FLOAT3
    , RTP_BUFFER_TYPE_HOST, verts, &vertsBD);
rtpBufferDescSetRange(vertsBD, 0, numVerts);
rtpBufferDescSetStride(vertsBD, sizeof(Vertex));
```

#### 3.12.4.7 RTPResult RTPAPI `rtpContextCreate` ( RTPcontexttype *type*, RTPcontext \* *context* )

Creates an OptiX Prime context.

By default, a context created with type [RTP\\_CONTEXT\\_TYPE\\_CUDA](#) will use all available CUDA devices. Specific devices can be selected using [rtpContextSetCudaDeviceNumbers](#). One device will be selected as the *primary device* and will be set as the current device when the function returns. If no available device has compute capability 2.0 or greater the created context will not be able to build acceleration structures.

## Parameters

in	<i>type</i>	The type of context to create
out	<i>context</i>	Pointer to the new OptiX Prime context

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_OBJECT\\_CREATION\\_FAILED](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#)

Example Usage:

```
RTPcontext context;
if(rtpContextCreate( RTP_CONTEXT_TYPE_CUDA, &context ) ==
    RTP_SUCCESS ) {
    int deviceNumbers[] = {0,1};
    rtpContextSetCudaDeviceNumbers( 2, deviceNumbers );
}
else
    rtpContextCreate( RTP_CONTEXT_TYPE_CPU, &context ); // Fallback to
    CPU
```

**3.12.4.8 RTPresult RTPAPI rtpContextDestroy ( RTPcontext context )**

Destroys an OptiX Prime context.

Ongoing work is finished before *context* is destroyed. All OptiX Prime objects associated with *context* are also destroyed when *context* is destroyed.

**Parameters**

in	<i>context</i>	OptiX Prime context to destroy
----	----------------	--------------------------------

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

**3.12.4.9 RTPresult RTPAPI rtpContextGetLastErrorString ( RTPcontext context, const char \*\* return\_string )**

Returns a string describing last error encountered.

This function returns an error string for the last error encountered in *context* that may contain invocation-specific details beyond the simple [RTPresult](#) error code. Note that this function may return errors from previous asynchronous launches or from calls by other threads.

**Parameters**

in	<i>context</i>	OptiX Prime context
out	<i>return_string</i>	String with error details

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)

See also [rtpGetErrorString](#)

#### 3.12.4.10 `RTPresult RTPAPI rtpContextSetCpuThreads ( RTPcontext context, unsigned numThreads )`

Sets the number of CPU threads used by a CPU context.

This function will return an error if the provided *context* is not of type `RTP_CONTEXT_TYPE_CPU`.

By default, one ray tracing thread is created per CPU core.

##### Parameters

in	<i>context</i>	OptiX Prime context
in	<i>numThreads</i>	Number of threads used for the CPU context

##### Return values

Relevant return values:

- `RTP_SUCCESS`
- `RTP_ERROR_INVALID_VALUE`
- `RTP_ERROR_UNKNOWN`

#### 3.12.4.11 `RTPresult RTPAPI rtpContextSetCudaDeviceNumbers ( RTPcontext context, unsigned deviceCount, const unsigned * deviceNumbers )`

Sets the CUDA devices used by a context.

The first device provided in *deviceNumbers* will be used as the *primary device*. Acceleration structures will be built on the primary device and copied to the others. To build the acceleration structures the primary device must be of compute capability 2.0 or greater. The current device will be set to the primary device when this function returns.

If *deviceCount*==0, then the primary device is selected automatically and all available devices are selected for use. *deviceNumbers* is ignored.

##### Parameters

in	<i>context</i>	OptiX Prime context
in	<i>deviceCount</i>	Number of devices supplied in <i>deviceNumbers</i> or 0
in	<i>deviceNumbers</i>	Array of integer device indices, or NULL if <i>deviceCount</i> ==0

This function will return an error if the provided context is not of type `RTP_CONTEXT_TYPE_CUDA`

##### Return values

Relevant return values:

- `RTP_SUCCESS`
- `RTP_ERROR_INVALID_VALUE`
- `RTP_ERROR_UNKNOWN`

#### 3.12.4.12 `RTPresult RTPAPI rtpGetErrorString ( RTPresult errorCode, const char ** errorString )`

Translates an RTPresult error code to a string.

Translates an RTPresult error code to a string describing the error.

##### Parameters

in	<i>errorCode</i>	Error code to be translated
out	<i>errorString</i>	Returned error string

##### Return values

Relevant return values:

- `RTP_SUCCESS`

See also `rtpContextGetLastErrorString`



**3.12.4.13** `RTPResult RTPAPI rtpGetVersion ( unsigned * version )`

Gets OptiX Prime version number.

The encoding for the version number is  $\text{major} \times 1000 + \text{minor} \times 10 + \text{micro}$ . For example, for version 3.5.1 this function would return 3051.

Parameters

<code>out</code>	<code><i>version</i></code>	Returned version
------------------	-----------------------------	------------------

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)

**3.12.4.14** `RTPResult RTPAPI rtpGetVersionString ( const char ** versionString )`

Gets OptiX Prime version string.

Returns OptiX Prime version string and other information in a human-readable format.

Parameters

<code>in</code>	<code><i>versionString</i></code>	Returned version information
-----------------	-----------------------------------	------------------------------

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)

**3.12.4.15** `RTPResult RTPAPI rtpHostBufferLock ( void * buffer, RTPsize size )`

Page-locks a host buffer.

Transfers between the host and device are faster if the host buffers are page-locked. However, page-locked memory is a limited resource and should be used judiciously.

Parameters

<code>in</code>	<code><i>buffer</i></code>	Buffer on the host
<code>in</code>	<code><i>size</i></code>	Size of the buffer

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)

**3.12.4.16** `RTPResult RTPAPI rtpHostBufferUnlock ( void * buffer )`

Unlocks a previously page-locked host buffer.

Transfers between the host and device are faster if the host buffers are page-locked. However, page-locked memory is a limited resource and should be used judiciously. Use this function on buffers previous page-locked with [rtpHostBufferLock](#).

## Parameters

<code>in</code>	<code>buffer</code>	Buffer on the host
-----------------	---------------------	--------------------

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)

3.12.4.17 `RTPResult RTPAPI rtpModelCopy ( RTPmodel model, RTPmodel srcModel )`

Copies one model to another.

This function copies a model from one OptiX Prime context to another for user-managed multi-GPU operation where one context is allocated per device. Only triangle models can be copied, not instance models. Furthermore, when a *srcModel* has the [RTP\\_BUILDER\\_PARAM\\_USE\\_CALLER\\_TRIANGLES](#) build parameter set to 1, and it is intended that the triangle data is automatically transferred to the other context, the destination (*model*) should have the build parameter set to 0 before the copy call. If the destination model also has the build parameter set to 1, its triangles must be set by calling [rtpModelSetTriangles](#) followed by [rtpModelUpdate](#) using `RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET`.

## Parameters

<code>in</code>	<code>model</code>	Destination model
<code>in</code>	<code>srcModel</code>	Source model

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.12.4.18 `RTPResult RTPAPI rtpModelCreate ( RTPcontext context, RTPmodel * model )`

Creates a model.

## Parameters

<code>in</code>	<code>context</code>	OptiX Prime context
<code>out</code>	<code>model</code>	Pointer to the new model

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.12.4.19 `RTPResult RTPAPI rtpModelDestroy ( RTPmodel model )`

Destroys a model.

Any queries created on the model are also destroyed with the model. The queries are allowed to finish before they are destroyed.

## Parameters

<code>in</code>	<code>model</code>	Model
-----------------	--------------------	-------

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.12.4.20 `RTPResult RTPAPI rtpModelFinish ( RTPmodel model )`

Blocks current thread until model update is finished.

This function can be called multiple times. It will return immediately if the previous update has already finished.

## Parameters

<code>in</code>	<code>model</code>	Model
-----------------	--------------------	-------

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.12.4.21 `RTPResult RTPAPI rtpModelGetContext ( RTPmodel model, RTPcontext * context )`

Gets the context object associated with the model.

## Parameters

<code>in</code>	<code>model</code>	Model to obtain the context from
<code>out</code>	<code>context</code>	Returned context

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.12.4.22 `RTPResult RTPAPI rtpModelGetFinished ( RTPmodel model, int * isFinished )`

Polls the status of a model update.

## Parameters

<code>in</code>	<code>model</code>	Model
<code>out</code>	<code>isFinished</code>	Returns finished status

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

#### 3.12.4.23 `RTPresult RTPAPI rtpModelSetBuilderParameter ( RTPmodel model_api, RTPbuilderparam param, RTPsize size, const void * ptr )`

Specifies a builder parameter for a model.

The following builder parameters are supported:

`RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES` : *int*

If the value for `RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES` is set to 0 (default), Prime uses an internal representation for triangles (which requires additional memory) to improve query performance and does not reference the user's vertex buffer during a query. If set to 1, Prime uses the provided triangle data as-is, which may result in slower query performance, but reduces memory usage.

`RTP_BUILDER_PARAM_CHUNK_SIZE` : *RTPsize*

Acceleration structures are built in chunks to reduce the amount of scratch memory needed. The size of the scratch memory chunk is specified in bytes by `RTP_BUILDER_PARAM_CHUNK_SIZE`. If set to -1, the chunk size has no limit. If set to 0 (default) the chunk size is chosen automatically, currently as 10% of the total available video memory for GPU builds and 512MB for CPU builds.

##### Parameters

in	<i>model_api</i>	Model
in	<i>param</i>	Builder parameter to set
in	<i>size</i>	Size in bytes of the parameter being set
in	<i>ptr</i>	Pointer to where the value of the attribute will be copied from. This must point to at least <i>size</i> bytes of memory

##### Return values

Relevant return values:

- `RTP_SUCCESS`
- `RTP_ERROR_INVALID_VALUE`
- `RTP_ERROR_UNKNOWN`

#### 3.12.4.24 `RTPresult RTPAPI rtpModelSetInstances ( RTPmodel model, RTPbufferdesc instances, RTPbufferdesc transforms )`

Sets the instance data for a model.

The *instances* buffer specifies a list of model instances, and the *transforms* buffer holds a transformation matrix for each instance. The instance buffer type must be `RTP_BUFFER_TYPE_HOST`.

Instance buffers must be of format `RTP_BUFFER_FORMAT_INSTANCE_MODEL`, and transform buffers of format `RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x4` or `RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3`. If a stride is specified for the transformations, it must be a multiple of 16 bytes. Furthermore, the matrices must be stored in row-major order. Only affine transformations are supported, and the last row is always assumed to be [0.0, 0.0, 0.0, 1.0].

All instance models in the *instances* buffer must belong to the same context as the model itself. Additionally, the build parameter `RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES` must be the same for all models (if applied). Setting `RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES` for a model which contains instances has no effect.

The buffers are not used until `rtpModelUpdate` is called.

##### Parameters

in	<i>model</i>	Model
----	--------------	-------

<code>in</code>	<i>instances</i>	Buffer descriptor for instances
<code>in</code>	<i>transforms</i>	Buffer descriptor for 4x4 transform matrices

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

#### 3.12.4.25 **RTPResult RTPAPI rtpModelSetTriangles** ( **RTPmodel** *model*, **RTPbufferdesc** *indices*, **RTPbufferdesc** *vertices* )

Sets the triangle data for a model.

The index buffer specifies triplet of vertex indices. If the index buffer descriptor is not specified (e.g. `indices==NULL`), the vertex buffer is considered to be a flat list of triangles, with every three vertices forming a triangle. The buffers are not used until [rtpModelUpdate](#) is called.

**Parameters**

<code>in</code>	<i>model</i>	Model
<code>in</code>	<i>indices</i>	Buffer descriptor for triangle vertex indices, or NULL
<code>in</code>	<i>vertices</i>	Buffer descriptor for triangle vertices

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

#### 3.12.4.26 **RTPResult RTPAPI rtpModelUpdate** ( **RTPmodel** *model*, **unsigned** *hints* )

Updates data, or creates an acceleration structure over triangles or instances.

Depending on the specified hints, [rtpModelUpdate](#) performs different operations:

If the flag [RTP\\_MODEL\\_HINT\\_ASYNC](#) is specified, some or all of the acceleration structure update may run asynchronously and [rtpModelUpdate](#) may return before the update is finished. In the case of [RTP\\_MODEL\\_HINT\\_NONE](#), the acceleration structure build is blocking. It is important that buffers specified in [rtpModelSetTriangles](#) and [rtpModelSetInstances](#) not be modified until the update has finished. [rtpModelFinish](#) blocks the current thread until the update is finished. [rtpModelGetFinished](#) can be used to poll until the update is finished. Once the update has finished the input buffers can be modified.

The acceleration structure build performed by [rtpModelUpdate](#) uses a fast, high quality algorithm, but has the cost of requiring additional working memory. The amount of working memory is controlled by [RTP\\_BUILDER\\_PARAM\\_CHUNK\\_SIZE](#).

The flag [RTP\\_MODEL\\_HINT\\_MASK\\_UPDATE](#) should be used to inform Prime when visibility mask data changed (after calling [rtpModelSetTriangles](#) with the updated values), e.g. when the indices format [RTP\\_BUFFER\\_FORMAT\\_INDICES\\_INT3\\_MASK\\_INT](#) is used. [RTP\\_MODEL\\_HINT\\_MASK\\_UPDATE](#) can be combined with [RTP\\_MODEL\\_HINT\\_ASYNC](#) to perform asynchronous data updates.

Hint [RTP\\_MODEL\\_HINT\\_USER\\_TRIANGLES\\_AFTER\\_COPY\\_SET](#) should be used when a triangle model has been copied (with the user triangle build flag set), and new user triangles have been set (by calling [rtpModelSetTriangles](#) again with the updated values). [RTP\\_MODEL\\_HINT\\_USER\\_TRIANGLES\\_AFTER\\_COPY\\_SET](#) can be combined with [RTP\\_MODEL\\_HINT\\_ASYNC](#) to perform asynchronous data updates.

## Parameters

in	<i>model</i>	Model
in	<i>hints</i>	A combination of flags from <a href="#">RTPmodelhint</a>

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

Example Usage:

```
RTPmodel model;
rtpModelCreate(context, &model);
rtpModelSetTriangles(model, 0, vertsBD);
rtpModelUpdate(model, RTP_MODEL_HINT_ASYNC);

// ... do useful work on CPU while GPU is busy

rtpModelFinish(model);

// It is now safe to modify vertex buffer
```

#### 3.12.4.27 RTPresult RTPAPI rtpQueryCreate ( RTPmodel *model*, RTPQuerytype *queryType*, RTPQuery \* *query* )

Creates a query on a model.

If the model to which a query is bound destroyed with [rtpModelDestroy\(\)](#) the query will be destroyed as well.

## Parameters

in	<i>model</i>	Model to use for this query
in	<i>queryType</i>	Type of the query
out	<i>query</i>	Pointer to the new query

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

#### 3.12.4.28 RTPresult RTPAPI rtpQueryDestroy ( RTPQuery *query* )

Destroys a query.

The query is finished before it is destroyed

## Parameters

in	<i>query</i>	Query to be destroyed
----	--------------	-----------------------

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

**3.12.4.29 RTPresult RTPAPI rtpQueryExecute ( RTPQuery *query*, unsigned *hints* )**

Executes a raytracing query.

If the flag `RTP_QUERY_HINT_ASYNC` is specified, `rtpQueryExecute` may return before the query is actually finished. `rtpQueryFinish` can be called to block the current thread until the query is finished, or `rtpQueryGetFinished` can be used to poll until the query is finished.

**Parameters**

in	<i>query</i>	Query
in	<i>hints</i>	A combination of flags from <a href="#">RTPQueryhint</a>

Once the query has finished all of the hits are guaranteed to have been returned, and it is safe to modify the ray buffer.

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

Example Usage:

```
RTPQuery query;
rtpQueryCreate(model, RTP_QUERY_TYPE_CLOSEST, &query);
rtpQuerySetRays(query, raysBD);
rtpQuerySetHits(hits, hitsBD);
rtpQueryExecute(query, 0);
// safe to modify ray buffer and process hits
```

**3.12.4.30 RTPresult RTPAPI rtpQueryFinish ( RTPQuery *query* )**

Blocks current thread until query is finished.

This function can be called multiple times. It will return immediately if the query has already finished.

**Parameters**

in	<i>query</i>	Query
----	--------------	-------

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

**3.12.4.31 RTPresult RTPAPI rtpQueryGetContext ( RTPQuery *query*, RTPcontext \* *context* )**

Gets the context object associated with a query.

**Parameters**

in	<i>query</i>	Query to obtain the context from
out	<i>context</i>	Returned context

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)

- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

#### 3.12.4.32 `RTPResult RTPAPI rtpQueryGetFinished ( RTPQuery query, int * isFinished )`

Polls the status of a query.

Parameters

in	<i>query</i>	Query
out	<i>isFinished</i>	Returns finished status

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

#### 3.12.4.33 `RTPResult RTPAPI rtpQuerySetCudaStream ( RTPQuery query, cudaStream_t stream )`

Sets a sync stream for a query.

Specify a Cuda stream used for synchronization. If no stream is specified, the default 0-stream is used. A stream can only be specified for contexts with type [RTP\\_CONTEXT\\_TYPE\\_CUDA](#).

Parameters

in	<i>query</i>	Query
in	<i>stream</i>	A cuda stream

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

#### 3.12.4.34 `RTPResult RTPAPI rtpQuerySetHits ( RTPQuery query, RTPbufferdesc hits )`

Sets the hits buffer for a query.

A hit is reported for every ray in the query. Therefore the size of the range in the hit buffer must match that of the ray buffer.

Parameters

in	<i>query</i>	Query
in	<i>hits</i>	Buffer descriptor for hits

**Return values**

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)



#### 3.12.4.35 `RTPresult RTPAPI rtpQuerySetRays ( RTPquery query, RTPbufferdesc rays )`

Sets the rays buffer for a query.

The rays buffer is not accessed until [rtpQueryExecute\(\)](#) is called.

## Parameters

<code>in</code>	<i>query</i>	Query
<code>in</code>	<i>rays</i>	Buffer descriptor for rays

## Return values

Relevant return values:

- [RTP\\_SUCCESS](#)
- [RTP\\_ERROR\\_INVALID\\_VALUE](#)
- [RTP\\_ERROR\\_UNKNOWN](#)

3.13 `optix_prime_declarations.h` File Reference

## 3.13.1 Detailed Description

OptiX Prime public API declarations.

## Author

NVIDIA Corporation OptiX Prime public API declarations

## Enumerations

- enum [RTPresult](#) {  
[RTP\\_SUCCESS](#) = 0,  
[RTP\\_ERROR\\_INVALID\\_VALUE](#) = 1,  
[RTP\\_ERROR\\_OUT\\_OF\\_MEMORY](#) = 2,  
[RTP\\_ERROR\\_INVALID\\_HANDLE](#) = 3,  
[RTP\\_ERROR\\_NOT\\_SUPPORTED](#) = 4,  
[RTP\\_ERROR\\_OBJECT\\_CREATION\\_FAILED](#) = 5,  
[RTP\\_ERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#) = 6,  
[RTP\\_ERROR\\_INVALID\\_CONTEXT](#) = 7,  
[RTP\\_ERROR\\_VALIDATION\\_ERROR](#) = 8,  
[RTP\\_ERROR\\_INVALID\\_OPERATION](#) = 9,  
[RTP\\_ERROR\\_UNKNOWN](#) = 999 }
- enum [RTPcontexttype](#) {  
[RTP\\_CONTEXT\\_TYPE\\_CPU](#) = 0x100,  
[RTP\\_CONTEXT\\_TYPE\\_CUDA](#) = 0x101 }
- enum [RTPbuffertype](#) {  
[RTP\\_BUFFER\\_TYPE\\_HOST](#) = 0x200,  
[RTP\\_BUFFER\\_TYPE\\_CUDA\\_LINEAR](#) = 0x201 }
- enum [RTPbufferformat](#) {

```

RTP_BUFFER_FORMAT_INDICES_INT3 = 0x400,
RTP_BUFFER_FORMAT_INDICES_INT3_MASK_INT = 0x401,
RTP_BUFFER_FORMAT_VERTEX_FLOAT3 = 0x420,
RTP_BUFFER_FORMAT_VERTEX_FLOAT4 = 0x421,
RTP_BUFFER_FORMAT_RAY_ORIGIN_DIRECTION = 0x440,
RTP_BUFFER_FORMAT_RAY_ORIGIN_TMIN_DIRECTION_TMAX = 0x441,
RTP_BUFFER_FORMAT_RAY_ORIGIN_MASK_DIRECTION_TMAX = 0x442,
RTP_BUFFER_FORMAT_HIT_BITMASK = 0x460,
RTP_BUFFER_FORMAT_HIT_T = 0x461,
RTP_BUFFER_FORMAT_HIT_T_TRIID = 0x462,
RTP_BUFFER_FORMAT_HIT_T_TRIID_U_V = 0x463,
RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID = 0x464,
RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID_U_V = 0x465,
RTP_BUFFER_FORMAT_INSTANCE_MODEL = 0x480,
RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x4 = 0x490,
RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3 = 0x491 }

• enum RTPQuerytype {
    RTP_QUERY_TYPE_ANY = 0x1000,
    RTP_QUERY_TYPE_CLOSEST = 0x1001 }

• enum RTPmodelhint {
    RTP_MODEL_HINT_NONE = 0x0000,
    RTP_MODEL_HINT_ASYNC = 0x2001,
    RTP_MODEL_HINT_MASK_UPDATE = 0x2002,
    RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET = 0x2004 }

• enum RTPQueryhint {
    RTP_QUERY_HINT_NONE = 0x0000,
    RTP_QUERY_HINT_ASYNC = 0x4001,
    RTP_QUERY_HINT_WATERTIGHT = 0x4002 }

• enum RTPbuilderparam {
    RTP_BUILDER_PARAM_CHUNK_SIZE = 0x800,
    RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES = 0x801 }

```

### 3.13.2 Enumeration Type Documentation

#### 3.13.2.1 enum RTPbufferformat

Buffer formats

Enumerator

**RTP\_BUFFER\_FORMAT\_INDICES\_INT3** Index buffer with 3 integer vertex indices per triangle

**RTP\_BUFFER\_FORMAT\_INDICES\_INT3\_MASK\_INT** Index buffer with 3 integer vertex indices per triangle, and an integer visibility mask

**RTP\_BUFFER\_FORMAT\_VERTEX\_FLOAT3** Vertex buffer with 3 floats per vertex position

**RTP\_BUFFER\_FORMAT\_VERTEX\_FLOAT4** Vertex buffer with 4 floats per vertex position

**RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_DIRECTION** float3:origin float3:direction

**RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_TMIN\_DIRECTION\_TMAX** float3:origin, float:tmin, float3:direction, float:tmax

**RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_MASK\_DIRECTION\_TMAX** float3:origin, int:mask, float3:direction, float:tmax. If used, buffer format RTP\_BUFFER\_FORMAT\_INDICES\_INT3\_MASK\_INT is required!

**RTP\_BUFFER\_FORMAT\_HIT\_BITMASK** one bit per ray 0=miss, 1=hit

**RTP\_BUFFER\_FORMAT\_HIT\_T** float:ray distance (t < 0 for miss)

**RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID** float:ray distance (t < 0 for miss), int:triangle id

**RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_U\_V** float:ray distance (t < 0 for miss), int:triangle id, float2:barycentric coordinates u,v (w=1-u-v)

***RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_INSTID*** float:ray distance ( $t < 0$  for miss), int:triangle id, int:instance position in list

***RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_INSTID\_U\_V*** float:ray distance ( $t < 0$  for miss), int:triangle id, int:instance position in list, float2:barycentric coordinates  $u,v$  ( $w=1-u-v$ )

***RTP\_BUFFER\_FORMAT\_INSTANCE\_MODEL*** RTPmodel:objects of type RTPmodel

***RTP\_BUFFER\_FORMAT\_TRANSFORM\_FLOAT4x4*** float:row major 4x4 affine matrix (it is assumed that the last row has the entries 0.0f, 0.0f, 0.0f, 1.0f, and will be ignored)

***RTP\_BUFFER\_FORMAT\_TRANSFORM\_FLOAT4x3*** float:row major 4x3 affine matrix

### 3.13.2.2 enum RTPbuffertype

Buffer types

Enumerator

***RTP\_BUFFER\_TYPE\_HOST*** Buffer in host memory

***RTP\_BUFFER\_TYPE\_CUDA\_LINEAR*** Linear buffer in device memory on a cuda device

### 3.13.2.3 enum RTPbuilderparam

Enumerator

***RTP\_BUILDER\_PARAM\_CHUNK\_SIZE*** Number of bytes used for a chunk of the acceleration structure build

***RTP\_BUILDER\_PARAM\_USE\_CALLER\_TRIANGLES*** A hint to specify which data should be used for the intersection test

### 3.13.2.4 enum RTPcontexttype

Context types

Enumerator

***RTP\_CONTEXT\_TYPE\_CPU*** CPU context

***RTP\_CONTEXT\_TYPE\_CUDA*** CUDA context

### 3.13.2.5 enum RTPmodelhint

Model hints

Enumerator

***RTP\_MODEL\_HINT\_NONE*** No hints. Use default settings.

***RTP\_MODEL\_HINT\_ASYNC*** Asynchronous model updating

***RTP\_MODEL\_HINT\_MASK\_UPDATE*** Upload buffer with mask data again

***RTP\_MODEL\_HINT\_USER\_TRIANGLES\_AFTER\_COPY\_SET*** Clear dirty flag of triangles.

### 3.13.2.6 enum RTPqueryhint

Query hints

Enumerator

***RTP\_QUERY\_HINT\_NONE*** No hints. Use default settings.

***RTP\_QUERY\_HINT\_ASYNC*** Asynchronous query execution

***RTP\_QUERY\_HINT\_WATERTIGHT*** Use watertight ray-triangle intersection

3.13.2.7 enum `RTPQuerytype`

Query types

Enumerator

**`RTP_QUERY_TYPE_ANY`** Return any hit along a ray

**`RTP_QUERY_TYPE_CLOSEST`** Return only the closest hit along a ray

3.13.2.8 enum `RTPResult`

Return value for OptiX Prime APIs

Enumerator

**`RTP_SUCCESS`** Success

**`RTP_ERROR_INVALID_VALUE`** An invalid value was provided

**`RTP_ERROR_OUT_OF_MEMORY`** Out of memory

**`RTP_ERROR_INVALID_HANDLE`** An invalid handle was supplied

**`RTP_ERROR_NOT_SUPPORTED`** An unsupported function was requested

**`RTP_ERROR_OBJECT_CREATION_FAILED`** Object creation failed

**`RTP_ERROR_MEMORY_ALLOCATION_FAILED`** Memory allocation failed

**`RTP_ERROR_INVALID_CONTEXT`** An invalid context was provided

**`RTP_ERROR_VALIDATION_ERROR`** A validation error occurred

**`RTP_ERROR_INVALID_OPERATION`** An invalid operation was performed

**`RTP_ERROR_UNKNOWN`** Unknown error

3.14 `optix_primepp.h` File Reference

## 3.14.1 Detailed Description

A C++ wrapper around the OptiX Prime API.

Classes

- class `optix::prime::ContextObj`
- class `optix::prime::BufferDescObj`
- class `optix::prime::ModelObj`
- class `optix::prime::QueryObj`
- class `optix::prime::Exception`

Macros

- `#define CHK(code) checkError( code, getContext()->getRTPcontext() )`

Typedefs

- typedef `Handle< BufferDescObj > optix::prime::BufferDesc`
- typedef `Handle< ContextObj > optix::prime::Context`
- typedef `Handle< ModelObj > optix::prime::Model`
- typedef `Handle< QueryObj > optix::prime::Query`

## Functions

- `std::string optix::prime::getVersionString ()`
- `void optix::prime::checkError (RTPresult code)`
- `void optix::prime::checkError (RTPresult code, RTPcontext context)`

3.15 `optix_world.h` File Reference

## 3.15.1 Detailed Description

OptiX public API C and C++ API.

## Author

NVIDIA Corporation This header is designed to be included by both host and device code providing access to the C-API along with the C++ API found in `optixpp_namespaces.h`. In addition various helper classes and file will also be included when compiling C++ compatible code.

Note that the CUDA vector types will be defined in the `optix::` namespace.

3.16 `optixpp_namespace.h` File Reference

## 3.16.1 Detailed Description

A C++ wrapper around the OptiX API.

## Classes

- class `optix::Handle< T >`
- class `optix::Exception`
- class `optix::APIObj`
- class `optix::DestroyableObj`
- class `optix::ScopedObj`
- class `optix::VariableObj`
- class `optix::ContextObj`
- class `optix::ProgramObj`
- class `optix::GroupObj`
- class `optix::GeometryGroupObj`
- class `optix::TransformObj`
- class `optix::SelectorObj`
- class `optix::AccelerationObj`
- class `optix::GeometryInstanceObj`
- class `optix::GeometryObj`
- class `optix::MaterialObj`
- class `optix::TextureSamplerObj`
- class `optix::BufferObj`
- struct `optix::bufferId< T, Dim >`
- class `optix::callableProgramId< T >`
- class `optix::RemoteDeviceObj`

## Macros

- `#define WIN32_LEAN_AND_MEAN`
- `#define rtBufferId optix::bufferId`
- `#define RT_INTERNAL_CALLABLE_PROGRAM_DEFS()`
- `#define rtCallableProgramId optix::callableProgramId`

## Typedefs

- typedef Handle< AccelerationObj > **optix::Acceleration**
- typedef Handle< BufferObj > **optix::Buffer**
- typedef Handle< ContextObj > **optix::Context**
- typedef Handle< GeometryObj > **optix::Geometry**
- typedef Handle< GeometryGroupObj > **optix::GeometryGroup**
- typedef Handle  
    < GeometryInstanceObj > **optix::GeometryInstance**
- typedef Handle< GroupObj > **optix::Group**
- typedef Handle< MaterialObj > **optix::Material**
- typedef Handle< ProgramObj > **optix::Program**
- typedef Handle< RemoteDeviceObj > **optix::RemoteDevice**
- typedef Handle< SelectorObj > **optix::Selector**
- typedef Handle< TextureSamplerObj > **optix::TextureSampler**
- typedef Handle< TransformObj > **optix::Transform**
- typedef Handle< VariableObj > **optix::Variable**

## Functions

- template<typename ReturnT >  
    class callableProgramId< ReturnT()> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**
- template<typename ReturnT , typename Arg0T >  
    class callableProgramId  
    < ReturnT(Arg0T)> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T >  
    class callableProgramId  
    < ReturnT(Arg0T, Arg1T)> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T >  
    class callableProgramId  
    < ReturnT(Arg0T, Arg1T, Arg2T)> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T >  
    class callableProgramId  
    < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T)> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T >  
    class callableProgramId  
    < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T, Arg4T)> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T >  
    class callableProgramId  
    < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T)> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T >  
    class callableProgramId  
    < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T)> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T >  
    class callableProgramId  
    < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T)> **optix::RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS ()**

- `template<typename ReturnT, typename Arg0T, typename Arg1T, typename Arg2T, typename Arg3T, typename Arg4T, typename Arg5T, typename Arg6T, typename Arg7T, typename Arg8T >`  
`class callableProgramId`  
`< ReturnT(Arg0T, Arg1T, Arg2T,`  
`Arg3T, Arg4T, Arg5T, Arg6T,`  
`Arg7T, Arg8T)> optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()`
- `template<typename ReturnT, typename Arg0T, typename Arg1T, typename Arg2T, typename Arg3T, typename Arg4T, typename Arg5T, typename Arg6T, typename Arg7T, typename Arg8T, typename Arg9T >`  
`class callableProgramId`  
`< ReturnT(Arg0T, Arg1T, Arg2T,`  
`Arg3T, Arg4T, Arg5T, Arg6T,`  
`Arg7T, Arg8T, Arg9T)> optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()`

### 3.16.2 Macro Definition Documentation

#### 3.16.2.1 #define RT\_INTERNAL\_CALLABLE\_PROGRAM\_DEFS( )

Value:

```
{
    public:
        callableProgramId() {}
        callableProgramId(int id) : m_id(id) {}
        int getId() const { return m_id; }
    private:
        int m_id;
}
```

`callableProgramId` is a host version of the device side `callableProgramId`.

Use `callableProgramId` to define types that can be included from both the host and device code. This class provides a container that can be used to transport the program id back and forth between host and device code. The `callableProgramId` class is useful, because it can take a program id obtained from `rtProgramGetId` and provide accessors for calling the program corresponding to the program id.

"`bindless_type.h`" used by both host and device code:

```
#include <optix_world.h>
struct ProgramInfo {
    int val;
    rtProgramId<int(int)> program;
};
```

Host code:

```
#include "bindless_type.h"
ProgramInfo input_program_info;
input_program_info.val = 0;
input_program_info.program = rtCallableProgramId<int(int)>(inputProgram0->getId());
context["input_program_info"]->setUserData(sizeof(ProgramInfo), &input_program_info);
```

Device code:

```
#include "bindless_type.h"
rtBuffer<int,1> result;
rtDeclareVariable(ProgramInfo, input_program_info, ,);

RT_PROGRAM void bindless()
{
    int value = input_program_info.program(input_program_info.val);
    result[0] = value;
}
```

## 3.17 optixu.h File Reference



## 3.17.1 Detailed Description

Simple API for performing raytracing queries using OptiX or the CPU.

## Macros

- `#define RTU_INLINE` static inline
- `#define RTU_CHECK_ERROR(func)`
- `#define RTU_GROUP_ADD_CHILD(_parent, _child, _index)`
- `#define RTU_SELECTOR_ADD_CHILD(_parent, _child, _index)`

## Functions

- `RTresult RTAPI rtuNameForType (RTobjecttype type, char *buffer, RTsize bufferSize)`
- `RTresult RTAPI rtuGetSizeForRTformat (RTformat format, size_t *size)`
- `RTresult RTAPI rtuCUDACompileString (const char *source, const char **preprocessorArguments, unsigned int numPreprocessorArguments, RTsize *resultSize, RTsize *errorSize)`
- `RTresult RTAPI rtuCUDACompileFile (const char *filename, const char **preprocessorArguments, unsigned int numPreprocessorArguments, RTsize *resultSize, RTsize *errorSize)`
- `RTresult RTAPI rtuCUDAGetCompileResult (char *result, char *error)`
- `RTresult RTAPI rtuCreateClusteredMesh (RTcontext context, unsigned int usePTX32InHost64, RTgeometry *mesh, unsigned int num_verts, const float *verts, unsigned int num_tris, const unsigned *indices, const unsigned *mat_indices)`
- `RTresult RTAPI rtuCreateClusteredMeshExt (RTcontext context, unsigned int usePTX32InHost64, RTgeometry *mesh, unsigned int num_verts, const float *verts, unsigned int num_tris, const unsigned *indices, const unsigned *mat_indices, RTbuffer norms, const unsigned *norm_indices, RTbuffer tex_coords, const unsigned *tex_indices)`
- `RTU_INLINE RTresult rtuGroupAddChild (RTgroup group, RTobject child, unsigned int *index)`
- `RTU_INLINE RTresult rtuSelectorAddChild (RTselector selector, RTobject child, unsigned int *index)`
- `RTU_INLINE RTresult rtuGeometryGroupAddChild (RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int *index)`
- `RTU_INLINE RTresult rtuTransformSetChild (RTtransform transform, RTobject child)`
- `RTU_INLINE RTresult rtuTransformGetChild (RTtransform transform, RTobject *type)`
- `RTU_INLINE RTresult rtuTransformGetChildType (RTtransform transform, RTobjecttype *type)`
- `RTU_INLINE RTresult rtuGroupRemoveChild (RTgroup group, RTobject child)`
- `RTU_INLINE RTresult rtuSelectorRemoveChild (RTselector selector, RTobject child)`
- `RTU_INLINE RTresult rtuGeometryGroupRemoveChild (RTgeometrygroup geometrygroup, RTgeometryinstance child)`
- `RTU_INLINE RTresult rtuGroupRemoveChildByIndex (RTgroup group, unsigned int index)`
- `RTU_INLINE RTresult rtuSelectorRemoveChildByIndex (RTselector selector, unsigned int index)`
- `RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex (RTgeometrygroup geometrygroup, unsigned int index)`
- `RTU_INLINE RTresult rtuGroupGetChildIndex (RTgroup group, RTobject child, unsigned int *index)`
- `RTU_INLINE RTresult rtuSelectorGetChildIndex (RTselector selector, RTobject child, unsigned int *index)`
- `RTU_INLINE RTresult rtuGeometryGroupGetChildIndex (RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int *index)`

### 3.17.2 Macro Definition Documentation

#### 3.17.2.1 #define RTU\_CHECK\_ERROR( func )

Value:

```
do {
    RTresult code = func;
    if( code != RT_SUCCESS )
        return code;
} while(0)
```

#### 3.17.2.2 #define RTU\_GROUP\_ADD\_CHILD( \_parent, \_child, \_index )

Value:

```
unsigned int _count;
RTU_CHECK_ERROR( rtGroupGetChildCount( (_parent), &_count ) );
RTU_CHECK_ERROR( rtGroupSetChildCount( (_parent), _count+1 ) );
RTU_CHECK_ERROR( rtGroupSetChild( (_parent), _count, (_child) ) );
if( _index ) *(_index) = _count;
return RT_SUCCESS
```

#### 3.17.2.3 #define RTU\_SELECTOR\_ADD\_CHILD( \_parent, \_child, \_index )

Value:

```
unsigned int _count;
RTU_CHECK_ERROR( rtSelectorGetChildCount( (_parent), &_count ) );
RTU_CHECK_ERROR( rtSelectorSetChildCount( (_parent), _count+1 ) );
RTU_CHECK_ERROR( rtSelectorSetChild( (_parent), _count, (_child) ) );
if( _index ) *(_index) = _count;
return RT_SUCCESS
```

### 3.17.3 Function Documentation

#### 3.17.3.1 RTresult RTAPI rtuCreateClusteredMesh ( RTcontext context, unsigned int usePTX32InHost64, RTgeometry \* mesh, unsigned int num\_verts, const float \* verts, unsigned int num\_tris, const unsigned \* indices, const unsigned \* mat\_indices )

Create clustered triangle mesh for good memory coherence with paging on. Vertex, index and material buffers are created and attached to the mesh. Cluster's bounding box and intersection programs are attached to the mesh. The intersection program has the following attributes:

- `rtDeclareVariable( int, primitive_id, attribute primitive_id, );`
- `rtDeclareVariable(float3, texcoord, attribute texcoord, );` It is always zero
- `rtDeclareVariable(float3, geometric_normal, attribute geometric_normal, );`
- `rtDeclareVariable(float3, shading_normal, attribute shading_normal, );` It is equal to geometric\_normal

Created `RTgeometry` mesh expects there to be placed into a `RTgeometryinstance` where the `mat_indices` specified map into materials attached to the `RTgeometryinstance`

In the event of an error, please query the error string from the `RTcontext`.

Parameters

<i>context</i>	Context
<i>usePTX32In-Host64</i>	Use 32bit PTX bounding box and intersection programs in 64bit application. Takes effect only with 64bit host.
<i>mesh</i>	Output geometry
<i>num_verts</i>	Vertex count
<i>verts</i>	Vertices (num_verts*float*3) [ v1_x, v1_y, v1_z, v2.x, ... ]
<i>num_tris</i>	Triangle count
<i>indices</i>	Vertex indices (num_tris*unsigned*3) [ tri1_index1, tri1_index2, ... ]
<i>mat_indices</i>	Indices of materials (num_tris*unsigned) [ tri1_mat_index, tri2_mat_index, ... ]

**3.17.3.2 RTresult RTAPI rtuCreateClusteredMeshExt ( RTcontext context, unsigned int usePTX32InHost64, RTgeometry \* mesh, unsigned int num\_verts, const float \* verts, unsigned int num\_tris, const unsigned \* indices, const unsigned \* mat\_indices, RTbuffer norms, const unsigned \* norm\_indices, RTbuffer tex\_coords, const unsigned \* tex\_indices )**

Create clustered triangle mesh for good memory coherence with paging on. Buffers for vertices, indices, normals, indices of normals, texture coordinates, indices of texture coordinates and materials are created and attached to the mesh. Cluster's bounding box and intersection programs are attached to the mesh. The intersection program has the following attributes:

- [rtDeclareVariable\( int, primitive\\_id, attribute primitive\\_id, \);](#)
- [rtDeclareVariable\(float3, texcoord, attribute texcoord, \);](#)
- [rtDeclareVariable\(float3, geometric\\_normal, attribute geometric\\_normal, \);](#)
- [rtDeclareVariable\(float3, shading\\_normal, attribute shading\\_normal, \);](#)

Created [RTgeometry](#) mesh expects there to be placed into a [RTgeometryinstance](#) where the mat\_indices specified map into materials attached to the [RTgeometryinstance](#)

Vertex, normal and texture coordinate buffers can be shared between many geometry objects

In the event of an error, please query the error string from the RTcontext.

#### Parameters

<i>context</i>	Context
<i>usePTX32In-Host64</i>	Use 32bit PTX bounding box and intersection programs in 64bit application. Takes effect only with 64bit host.
<i>mesh</i>	Output geometry
<i>num_verts</i>	Vertex count
<i>verts</i>	Vertices (num_verts*float*3) [ v1_x, v1_y, v1_z, v2.x, ... ]
<i>num_tris</i>	Triangle count
<i>indices</i>	Vertex indices (num_tris*unsigned*3) [ tri1_index1, tri1_index2, ... ]
<i>mat_indices</i>	Indices of materials (num_tris*unsigned) [ tri1_mat_index, tri2_mat_index, ... ]
<i>norms</i>	Normals (num_norms*float*3) [ v1_x, v1_y, v1_z, v2.x, ... ]
<i>norm_indices</i>	Indices of vertex normals (num_tris*unsigned*3) [ tri1_norm_index1, tri1_norm_index2 ... ]
<i>tex_coords</i>	Texture uv coords (num_tex_coords*float*2) [ t1_u, t1_v, t2_u ... ]
<i>tex_indices</i>	Indices of texture uv (num_tris*unsigned*3) [ tri1_tex_index1, tri1_tex_index2 ... ]

**3.17.3.3 RTresult RTAPI rtuCUDACompileFile ( const char \* filename, const char \*\* preprocessorArguments, unsigned int numPreprocessorArguments, RTsize \* resultSize, RTsize \* errorSize )**

Compile a cuda source file.

## Parameters

in	<i>filename</i>	source code file name
in	<i>preprocessor-Arguments</i>	list of preprocessor arguments
in	<i>num-Preprocessor-Arguments</i>	number of preprocessor arguments
out	<i>resultSize</i>	size required to hold compiled result string
out	<i>errorSize</i>	size required to hold error string

## Return values

<i>RTresult</i>	Return code
-----------------	-------------

#### 3.17.3.4 RTresult RTAPI rtuCUDACompileString ( const char \* *source*, const char \*\* *preprocessorArguments*, unsigned int *numPreprocessorArguments*, RTsize \* *resultSize*, RTsize \* *errorSize* )

Compile a cuda source string.

## Parameters

in	<i>source</i>	source code string
in	<i>preprocessor-Arguments</i>	list of preprocessor arguments
in	<i>num-Preprocessor-Arguments</i>	number of preprocessor arguments
out	<i>resultSize</i>	size required to hold compiled result string
out	<i>errorSize</i>	size required to hold error string

## Return values

<i>RTresult</i>	Return code
-----------------	-------------

#### 3.17.3.5 RTresult RTAPI rtuCUDAGetCompileResult ( char \* *result*, char \* *error* )

Get the result of the most recent call to one of the above compile functions. The 'result' and 'error' parameters must point to memory large enough to hold the respective strings, as returned by the compile function.

## Parameters

out	<i>result</i>	compiled result string
out	<i>error</i>	error string

## Return values

<i>RTresult</i>	Return code
-----------------	-------------

#### 3.17.3.6 RTU\_INLINE RTresult rtuGeometryGroupAddChild ( RTgeometrygroup *geometrygroup*, RTgeometryinstance *child*, unsigned int \* *index* )

Add an entry to the end of the child array. Fills 'index' with the index of the added child, if the pointer is non-NULL.

#### 3.17.3.7 RTU\_INLINE RTresult rtuGeometryGroupGetChildIndex ( RTgeometrygroup *geometrygroup*, RTgeometryinstance *child*, unsigned int \* *index* )

Use a linear search to find the child in the child array, and return its index. Returns [RT\\_SUCCESS](#) if the child was found, [RT\\_ERROR\\_INVALID\\_VALUE](#) otherwise.

### 3.17.3.8 RTU\_INLINE RTresult rtuGeometryGroupRemoveChild ( RTgeometrygroup *geometrygroup*, RTgeometryinstance *child* )

Find the given child using a linear search in the child array and remove it. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

### 3.17.3.9 RTU\_INLINE RTresult rtuGeometryGroupRemoveChildByIndex ( RTgeometrygroup *geometrygroup*, unsigned int *index* )

Remove the child at the given index in the child array. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

### 3.17.3.10 RTresult RTAPI rtuGetSizeForRTformat ( RTformat *format*, size\_t \* *size* )

Return the size of a given [RTformat](#). [RT\\_FORMAT\\_USER](#) and [RT\\_FORMAT\\_UNKNOWN](#) return 0. Returns [RT\\_ERROR\\_INVALID\\_VALUE](#) if the format isn't recognized, [RT\\_SUCCESS](#) otherwise.

#### Parameters

in	<i>format</i>	OptiX format
out	<i>size</i>	Size of the format

#### Return values

<i>RTresult</i>	Return code
-----------------	-------------

### 3.17.3.11 RTU\_INLINE RTresult rtuGroupAddChild ( RTgroup *group*, RObject *child*, unsigned int \* *index* )

Add an entry to the end of the child array. Fills 'index' with the index of the added child, if the pointer is non-NULL.

### 3.17.3.12 RTU\_INLINE RTresult rtuGroupGetChildIndex ( RTgroup *group*, RObject *child*, unsigned int \* *index* )

Use a linear search to find the child in the child array, and return its index. Returns [RT\\_SUCCESS](#) if the child was found, [RT\\_ERROR\\_INVALID\\_VALUE](#) otherwise.

### 3.17.3.13 RTU\_INLINE RTresult rtuGroupRemoveChild ( RTgroup *group*, RObject *child* )

Find the given child using a linear search in the child array and remove it. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

### 3.17.3.14 RTU\_INLINE RTresult rtuGroupRemoveChildByIndex ( RTgroup *group*, unsigned int *index* )

Remove the child at the given index in the child array. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

### 3.17.3.15 RTresult RTAPI rtuNameForType ( RObjecttype *type*, char \* *buffer*, RTsize *bufferSize* )

Get the name string of a given type. See [RObjecttype](#) for more information.

#### Parameters

in	<i>type</i>	Type requested
out	<i>buffer</i>	Buffer to output the name string
in	<i>bufferSize</i>	Size of the provided buffer

#### Return values

<i>RTresult</i>	Return code
-----------------	-------------

### 3.17.3.16 RTU\_INLINE RTresult rtuSelectorAddChild ( RTselector *selector*, RObject *child*, unsigned int \* *index* )

Add an entry to the end of the child array. Fills 'index' with the index of the added child, if the pointer is non-NULL.

3.17.3.17 `RTU_INLINE RTresult rtuSelectorGetChildIndex ( RTselector selector, RObject child, unsigned int * index )`

Use a linear search to find the child in the child array, and return its index. Returns `RT_SUCCESS` if the child was found, `RT_ERROR_INVALID_VALUE` otherwise.

3.17.3.18 `RTU_INLINE RTresult rtuSelectorRemoveChild ( RTselector selector, RObject child )`

Find the given child using a linear search in the child array and remove it. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

3.17.3.19 `RTU_INLINE RTresult rtuSelectorRemoveChildByIndex ( RTselector selector, unsigned int index )`

Remove the child at the given index in the child array. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

3.17.3.20 `RTU_INLINE RTresult rtuTransformGetChild ( RTtransform transform, RObject * type )`

Wrap `rtTransformGetChild` and `rtTransformGetChildType` in order to provide a type-safe version for C++.

3.17.3.21 `RTU_INLINE RTresult rtuTransformGetChildType ( RTtransform transform, RObjecttype * type )`

Wrap `rtTransformGetChild` and `rtTransformGetChildType` in order to provide a type-safe version for C++.

3.17.3.22 `RTU_INLINE RTresult rtuTransformSetChild ( RTtransform transform, RObject child )`

Wrap `rtTransformSetChild` in order to provide a type-safe version for C++.

## 3.18 optixu\_aabb\_namespace.h File Reference

### 3.18.1 Detailed Description

OptiX public API.

#### Author

NVIDIA Corporation OptiX public API Reference - Public AABB namespace

#### Classes

- class `optix::Aabb`

#### Macros

- `#define RT_AABB_ASSERT assert`
- `#define OPTIXU_INLINE_DEFINED 1`
- `#define OPTIXU_INLINE __forceinline__`

## 3.19 optixu\_math\_namespace.h File Reference

### 3.19.1 Detailed Description

OptiX public API.

## Author

NVIDIA Corporation This file implements common mathematical operations on vector types (float3, float4 etc.) since these are not provided as standard by CUDA.

The syntax is modelled on the Cg standard library.

This file has also been modified from the original cutil\_math.h file. cutil\_math.h is a subset of this file, and you should use this file in place of any cutil\_math.h file you wish to use.

## Classes

- struct [optix::Onb](#)

## Macros

- #define **OPTIXU\_INLINE\_DEFINED** 1
- #define **OPTIXU\_INLINE** \_\_forceinline\_\_
- #define **OPTIXU\_MATH\_DEFINE\_IN\_NAMESPACE**

## Typedefs

- typedef unsigned int **optix::uint**
- typedef unsigned short **optix::ushort**

## Functions

- OPTIXU\_INLINE float **optix::fminf** (const float a, const float b)
- OPTIXU\_INLINE float **optix::fmaxf** (const float a, const float b)
- OPTIXU\_INLINE float **optix::copysignf** (const float dst, const float src)
- OPTIXU\_INLINE int **optix::max** (int a, int b)
- OPTIXU\_INLINE int **optix::min** (int a, int b)
- OPTIXU\_INLINE int **optix::float\_as\_int** (const float f)
- OPTIXU\_INLINE float **optix::int\_as\_float** (int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::lerp** (const float a, const float b, const float t)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::bilerp** (const float x00, const float x10, const float x01, const float x11, const float u, const float v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::clamp** (const float f, const float a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::getByIndex** (const float1 &v, int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (float1 &v, int i, float x)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator-** (const float2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::lerp** (const float2 &a, const float2 &b, const float t)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::bilerp** (const float2 &x00, const float2 &x10, const float2 &x01, const float2 &x11, const float u, const float v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::dot** (const float2 &a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::length** (const float2 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::normalize** (const float2 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::floor** (const float2 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::reflect** (const float2 &i, const float2 &n)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::faceforward** (const float2 &n, const float2 &i, const float2 &nref)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::expf** (const float2 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::getByIndex** (const float2 &v, int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (float2 &v, int i, float x)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator-** (const float3 &a)

- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::lerp** (const float3 &a, const float3 &b, const float t)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::bilerp** (const float3 &x00, const float3 &x10, const float3 &x01, const float3 &x11, const float u, const float v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::dot** (const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::cross** (const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::length** (const float3 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::normalize** (const float3 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::floor** (const float3 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::reflect** (const float3 &i, const float3 &n)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::faceforward** (const float3 &n, const float3 &i, const float3 &nref)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::expf** (const float3 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::getByIndex** (const float3 &v, int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (float3 &v, int i, float x)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator-** (const float4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::lerp** (const float4 &a, const float4 &b, const float t)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::bilerp** (const float4 &x00, const float4 &x10, const float4 &x01, const float4 &x11, const float u, const float v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::dot** (const float4 &a, const float4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::length** (const float4 &r)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::normalize** (const float4 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::floor** (const float4 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::reflect** (const float4 &i, const float4 &n)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::faceforward** (const float4 &n, const float4 &i, const float4 &nref)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::expf** (const float4 &v)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::getByIndex** (const float4 &v, int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (float4 &v, int i, float x)
- OPTIXU\_INLINE RT\_HOSTDEVICE int **optix::clamp** (const int f, const int a, const int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int **optix::getByIndex** (const int1 &v, int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (int1 &v, int i, int x)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::operator-** (const int2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::min** (const int2 &a, const int2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::max** (const int2 &a, const int2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int **optix::getByIndex** (const int2 &v, int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (int2 &v, int i, int x)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator-** (const int3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::min** (const int3 &a, const int3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::max** (const int3 &a, const int3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int **optix::getByIndex** (const int3 &v, int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (int3 &v, int i, int x)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator-** (const int4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::min** (const int4 &a, const int4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::max** (const int4 &a, const int4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int **optix::getByIndex** (const int4 &v, int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (int4 &v, int i, int x)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
unsigned int **optix::clamp** (const unsigned int f, const unsigned int a, const unsigned int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
unsigned int **optix::getByIndex** (const uint1 &v, unsigned int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (uint1 &v, int i, unsigned int x)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::min** (const uint2 &a, const uint2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::max** (const uint2 &a, const uint2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
unsigned int **optix::getByIndex** (const uint2 &v, unsigned int i)



- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (uint2 &v, int i, unsigned int x)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::min** (const uint3 &a, const uint3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::max** (const uint3 &a, const uint3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
unsigned int **optix::getByIndex** (const uint3 &v, unsigned int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (uint3 &v, int i, unsigned int x)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
unsigned int **optix::getByIndex** (const uint4 &v, unsigned int i)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::setByIndex** (uint4 &v, int i, unsigned int x)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::smoothstep** (const float edge0, const float edge1, const float x)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::temperature** (const float t)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::intersect\_triangle\_branchless** (const Ray &ray, const float3 &p0, const float3 &p1, const float3 &p2, float3 &n, float &t, float &beta, float &gamma)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::intersect\_triangle\_earlyexit** (const Ray &ray, const float3 &p0, const float3 &p1, const float3 &p2, float3 &n, float &t, float &beta, float &gamma)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::intersect\_triangle** (const Ray &ray, const float3 &p0, const float3 &p1, const float3 &p2, float3 &n, float &t, float &beta, float &gamma)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::refract** (float3 &r, const float3 &i, const float3 &n, const float ior)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::fresnel\_schlick** (const float cos\_theta, const float exponent=5.0f, const float minimum=0.0f, const float maximum=1.0f)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::fresnel\_schlick** (const float cos\_theta, const float exponent, const float3 &minimum, const float3 &maximum)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::luminance** (const float3 &rgb)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::luminanceCIE** (const float3 &rgb)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::cosine\_sample\_hemisphere** (const float u1, const float u2, float3 &p)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::square\_to\_disk** (const float2 &sample)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::cart\_to\_pol** (const float3 &v)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::make\_float2** (const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::make\_float2** (const int2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::make\_float2** (const uint2 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::fminf** (const float2 &a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::fminf** (const float2 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::fmaxf** (const float2 &a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::fmaxf** (const float2 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator+** (const float2 &a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator+** (const float2 &a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator+** (const float a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (float2 &a, const float2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator-** (const float2 &a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator-** (const float2 &a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator-** (const float a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (float2 &a, const float2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator\*** (const float2 &a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator\*** (const float2 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator\*** (const float s, const float2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (float2 &a, const float2 &s)

- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (float2 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator/** (const float2 &a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator/** (const float2 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::operator/** (const float s, const float2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator/=** (float2 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::clamp** (const float2 &v, const float a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::clamp** (const float2 &v, const float2 &a, const float2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::make\_float3** (const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::make\_float3** (const float2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::make\_float3** (const int3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::make\_float3** (const uint3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::fminf** (const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::fminf** (const float3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::fmaxf** (const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::fmaxf** (const float3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator+** (const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator+** (const float3 &a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator+** (const float a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator-** (const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator-** (const float3 &a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator-** (const float a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator\*** (const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator\*** (const float3 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator\*** (const float s, const float3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (float3 &a, const float3 &s)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (float3 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator/** (const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator/** (const float3 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::operator/** (const float s, const float3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator/=** (float3 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::clamp** (const float3 &v, const float a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::clamp** (const float3 &v, const float3 &a, const float3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const float3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const int4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const uint4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::fminf** (const float4 &a, const float4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::fminf** (const float4 &a)

- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::fmaxf** (const float4 &a, const float4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float **optix::fmaxf** (const float4 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator+** (const float4 &a, const float4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator+** (const float4 &a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator+** (const float a, const float4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (float4 &a, const float4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator-** (const float4 &a, const float4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator-** (const float4 &a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator-** (const float a, const float4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (float4 &a, const float4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator\*** (const float4 &a, const float4 &s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator\*** (const float4 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator\*** (const float s, const float4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (float4 &a, const float4 &s)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (float4 &a, const float s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator/** (const float4 &a, const float4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator/** (const float4 &a, const float s)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::operator/** (const float s, const float4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator/=** (float4 &a, const float s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::clamp** (const float4 &v, const float a, const float b)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::clamp** (const float4 &v, const float4 &a, const float4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::make\_int2** (const int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::make\_int2** (const float2 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::operator+** (const int2 &a, const int2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (int2 &a, const int2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::operator-** (const int2 &a, const int2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::operator-** (const int2 &a, const int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (int2 &a, const int2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::operator\*** (const int2 &a, const int2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::operator\*** (const int2 &a, const int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::operator\*** (const int s, const int2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (int2 &a, const int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::clamp** (const int2 &v, const int a, const int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::clamp** (const int2 &v, const int2 &a, const int2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator==** (const int2 &a, const int2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator!=** (const int2 &a, const int2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::make\_int3** (const int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::make\_int3** (const float3 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator+** (const int3 &a, const int3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (int3 &a, const int3 &b)

- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator-** (const int3 &a, const int3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (int3 &a, const int3 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator\*** (const int3 &a, const int3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator\*** (const int3 &a, const int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator\*** (const int s, const int3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (int3 &a, const int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator/** (const int3 &a, const int3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator/** (const int3 &a, const int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::operator/** (const int s, const int3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator/=** (int3 &a, const int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::clamp** (const int3 &v, const int a, const int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::clamp** (const int3 &v, const int3 &a, const int3 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator==** (const int3 &a, const int3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator!=** (const int3 &a, const int3 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::make\_int4** (const int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::make\_int4** (const float4 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator+** (const int4 &a, const int4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (int4 &a, const int4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator-** (const int4 &a, const int4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (int4 &a, const int4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator\*** (const int4 &a, const int4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator\*** (const int4 &a, const int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator\*** (const int s, const int4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (int4 &a, const int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator/** (const int4 &a, const int4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator/** (const int4 &a, const int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::operator/** (const int s, const int4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator/=** (int4 &a, const int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::clamp** (const int4 &v, const int a, const int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::clamp** (const int4 &v, const int4 &a, const int4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator==** (const int4 &a, const int4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator!=** (const int4 &a, const int4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::make\_uint2** (const unsigned int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::make\_uint2** (const float2 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::operator+** (const uint2 &a, const uint2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (uint2 &a, const uint2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::operator-** (const uint2 &a, const uint2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::operator-** (const uint2 &a, const unsigned int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (uint2 &a, const uint2 &b)

- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::operator\*** (const uint2 &a, const uint2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::operator\*** (const uint2 &a, const unsigned int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::operator\*** (const unsigned int s, const uint2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (uint2 &a, const unsigned int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::clamp** (const uint2 &v, const unsigned int a, const unsigned int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::clamp** (const uint2 &v, const uint2 &a, const uint2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator==** (const uint2 &a, const uint2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator!=** (const uint2 &a, const uint2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::make\_uint3** (const unsigned int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::make\_uint3** (const float3 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::operator+** (const uint3 &a, const uint3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (uint3 &a, const uint3 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::operator-** (const uint3 &a, const uint3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (uint3 &a, const uint3 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::operator\*** (const uint3 &a, const uint3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::operator\*** (const uint3 &a, const unsigned int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::operator\*** (const unsigned int s, const uint3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (uint3 &a, const unsigned int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::operator/** (const uint3 &a, const uint3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::operator/** (const uint3 &a, const unsigned int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::operator/** (const unsigned int s, const uint3 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator/=** (uint3 &a, const unsigned int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::clamp** (const uint3 &v, const unsigned int a, const unsigned int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::clamp** (const uint3 &v, const uint3 &a, const uint3 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator==** (const uint3 &a, const uint3 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator!=** (const uint3 &a, const uint3 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::make\_uint4** (const unsigned int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::make\_uint4** (const float4 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::min** (const uint4 &a, const uint4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::max** (const uint4 &a, const uint4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::operator+** (const uint4 &a, const uint4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator+=** (uint4 &a, const uint4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::operator-** (const uint4 &a, const uint4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator-=** (uint4 &a, const uint4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::operator\*** (const uint4 &a, const uint4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::operator\*** (const uint4 &a, const unsigned int s)

- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::operator\*** (const unsigned int s, const uint4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator\*=** (uint4 &a, const unsigned int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::operator/** (const uint4 &a, const uint4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::operator/** (const uint4 &a, const unsigned int s)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::operator/** (const unsigned int s, const uint4 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void **optix::operator/=** (uint4 &a, const unsigned int s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::clamp** (const uint4 &v, const unsigned int a, const unsigned int b)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::clamp** (const uint4 &v, const uint4 &a, const uint4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator==** (const uint4 &a, const uint4 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE bool **optix::operator!=** (const uint4 &a, const uint4 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::make\_int2** (const int3 &v0)
- OPTIXU\_INLINE RT\_HOSTDEVICE int2 **optix::make\_int2** (const int4 &v0)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::make\_int3** (const int4 &v0)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::make\_uint2** (const uint3 &v0)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint2 **optix::make\_uint2** (const uint4 &v0)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::make\_uint3** (const uint4 &v0)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::make\_float2** (const float3 &v0)
- OPTIXU\_INLINE RT\_HOSTDEVICE float2 **optix::make\_float2** (const float4 &v0)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::make\_float3** (const float4 &v0)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::make\_int3** (const int v0, const int2 &v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE int3 **optix::make\_int3** (const int2 &v0, const int v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::make\_int4** (const int v0, const int v1, const int2 &v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::make\_int4** (const int v0, const int2 &v1, const int v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::make\_int4** (const int2 &v0, const int v1, const int v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::make\_int4** (const int v0, const int3 &v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::make\_int4** (const int3 &v0, const int v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE int4 **optix::make\_int4** (const int2 &v0, const int2 &v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::make\_uint3** (const unsigned int v0, const uint2 &v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint3 **optix::make\_uint3** (const uint2 &v0, const unsigned int v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::make\_uint4** (const unsigned int v0, const unsigned int v1, const uint2 &v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::make\_uint4** (const unsigned int v0, const uint2 &v1, const unsigned int v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::make\_uint4** (const uint2 &v0, const unsigned int v1, const unsigned int v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::make\_uint4** (const unsigned int v0, const uint3 &v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::make\_uint4** (const uint3 &v0, const unsigned int v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE uint4 **optix::make\_uint4** (const uint2 &v0, const uint2 &v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::make\_float3** (const float2 &v0, const float v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE float3 **optix::make\_float3** (const float v0, const float2 &v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const float v0, const float v1, const float2 &v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const float v0, const float2 &v1, const float v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const float2 &v0, const float v1, const float v2)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const float v0, const float3 &v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const float3 &v0, const float v1)
- OPTIXU\_INLINE RT\_HOSTDEVICE float4 **optix::make\_float4** (const float2 &v0, const float2 &v1)

## 3.20 optixu\_math\_stream\_namespace.h File Reference

### 3.20.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation Stream operators for CUDA vector types

### Functions

- `std::ostream & optix::operator<< (std::ostream &os, const optix::Aabb &aabb)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::float4 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::float4 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::float3 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::float3 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::float2 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::float2 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::int4 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::int4 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::int3 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::int3 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::int2 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::int2 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::uint4 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::uint4 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::uint3 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::uint3 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::uint2 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::uint2 &v)`
- `template<unsigned int M, unsigned int N>  
std::ostream & optix::operator<< (std::ostream &os, const optix::Matrix< M, N > &m)`
- `template<unsigned int M, unsigned int N>  
std::istream & optix::operator>> (std::istream &is, optix::Matrix< M, N > &m)`

## 3.21 optixu\_matrix\_namespace.h File Reference

### 3.21.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Public Matrix namespace

### Classes

- struct [optix::VectorDim](#)< DIM >
- struct [optix::VectorDim](#)< 2 >
- struct [optix::VectorDim](#)< 3 >
- struct [optix::VectorDim](#)< 4 >
- class [optix::Matrix](#)< M, N >
- class [optix::Matrix](#)< M, N >

## Macros

- `#define OPTIXU_INLINE_DEFINED 1`
- `#define OPTIXU_INLINE __forceinline__`
- `#define RT_MATRIX_ACCESS(m, i, j) m[i*N+j]`
- `#define RT_MAT_DECL template <unsigned int M, unsigned int N>`

## Typedefs

- `typedef Matrix< 2, 2 > optix::Matrix2x2`
- `typedef Matrix< 2, 3 > optix::Matrix2x3`
- `typedef Matrix< 2, 4 > optix::Matrix2x4`
- `typedef Matrix< 3, 2 > optix::Matrix3x2`
- `typedef Matrix< 3, 3 > optix::Matrix3x3`
- `typedef Matrix< 3, 4 > optix::Matrix3x4`
- `typedef Matrix< 4, 2 > optix::Matrix4x2`
- `typedef Matrix< 4, 3 > optix::Matrix4x3`
- `typedef Matrix< 4, 4 > optix::Matrix4x4`

## Functions

- `template<unsigned int M>`  
`OPTIXU_INLINE RT_HOSTDEVICE`  
`Matrix< M, M > &optix::operator*= (Matrix< M, M > &m1, const Matrix< M, M > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > &optix::operator= (Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > &optix::operator+= (Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > &optix::operator*= (Matrix< M, N > &m1, float f)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > &optix::operator/= (Matrix< M, N > &m1, float f)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > optix::operator- (const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > optix::operator+ (const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > optix::operator/ (const Matrix< M, N > &m, float f)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > optix::operator* (const Matrix< M, N > &m, float f)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N > optix::operator* (float f, const Matrix< M, N > &m)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N >`  
`::floatM optix::operator* (const Matrix< M, N > &m, const typename Matrix< M, N >::floatN &v)`
- `RT_MAT_DECL OPTIXU_INLINE`  
`RT_HOSTDEVICE Matrix< M, N >`  
`::floatN optix::operator* (const typename Matrix< M, N >::floatM &v, const Matrix< M, N > &m)`
- `template<unsigned int M, unsigned int N, unsigned int R>`  
`OPTIXU_INLINE RT_HOSTDEVICE`  
`Matrix< M, R > optix::operator* (const Matrix< M, N > &m1, const Matrix< N, R > &m2)`
- `template<unsigned int N>`  
`OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator* (const Matrix< 2, N > &m, const typename`  
`Matrix< 2, N >::floatN &vec)`



- `template<unsigned int N>`  
`OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator* (const Matrix< 3, N > &m, const typename Matrix< 3, N >::floatN &vec)`
- `template<unsigned int N>`  
`OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator* (const Matrix< 4, N > &m, const typename Matrix< 4, N >::floatN &vec)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator* (const Matrix< 4, 4 > &m, const float4 &vec)`
- `template<unsigned int M, unsigned int N, unsigned int R>`  
`RT_HOSTDEVICE Matrix< M, R > optix::operator* (const Matrix< M, N > &m1, const Matrix< N, R > &m2)`
- `template<unsigned int M>`  
`RT_HOSTDEVICE Matrix< M, M > & optix::operator*= (Matrix< M, M > &m1, const Matrix< M, M > &m2)`
- `OPTIXU_INLINE RT_HOSTDEVICE`  
`Matrix< 3, 3 > optix::make_matrix3x3 (const Matrix< 4, 4 > &matrix)`

## 3.22 optixu\_traversal.h File Reference

### 3.22.1 Detailed Description

Simple API for performing raytracing queries using OptiX or the CPU.

#### Classes

- struct [RTUtraversalresult](#)

#### Typedefs

- `typedef struct RTUtraversal_api * RTUtraversal`

#### Enumerations

- enum [RTUQuerytype](#) {  
`RTU_QUERY_TYPE_ANY_HIT = 0,`  
`RTU_QUERY_TYPE_CLOSEST_HIT,`  
`RTU_QUERY_TYPE_COUNT }`
- enum [RTUrayformat](#) {  
`RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED = 0,`  
`RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED,`  
`RTU_RAYFORMAT_COUNT }`
- enum [RTUtriformat](#) {  
`RTU_TRIFORMAT_MESH = 0,`  
`RTU_TRIFORMAT_TRIANGLE_SOUP,`  
`RTU_TRIFORMAT_COUNT }`
- enum [RTUinitoptions](#) {  
`RTU_INITOPTION_NONE = 0,`  
`RTU_INITOPTION_GPU_ONLY = 1 << 0,`  
`RTU_INITOPTION_CPU_ONLY = 1 << 1,`  
`RTU_INITOPTION_CULL_BACKFACE = 1 << 2 }`
- enum [RTUoutput](#) {  
`RTU_OUTPUT_NONE = 0,`  
`RTU_OUTPUT_NORMAL = 1 << 0,`  
`RTU_OUTPUT_BARYCENTRIC = 1 << 1,`  
`RTU_OUTPUT_BACKFACING = 1 << 2 }`
- enum [RTUoption](#) { `RTU_OPTION_INT_NUM_THREADS = 0 }`

## Functions

- [RTresult](#) RTAPI [rtuTraversalCreate](#) ([RTUtraversal](#) \*traversal, [RTUquerytype](#) query\_type, [RTUrayformat](#) ray\_format, [RTUtriformat](#) tri\_format, unsigned int outputs, unsigned int options, [RTcontext](#) context)
- [RTresult](#) RTAPI [rtuTraversalGetErrorString](#) ([RTUtraversal](#) traversal, [RTresult](#) code, const char \*\*return\_string)
- [RTresult](#) RTAPI [rtuTraversalSetOption](#) ([RTUtraversal](#) traversal, [RTUoption](#) option, void \*value)
- [RTresult](#) RTAPI [rtuTraversalSetMesh](#) ([RTUtraversal](#) traversal, unsigned int num\_verts, const float \*verts, unsigned int num\_tris, const unsigned \*indices)
- [RTresult](#) RTAPI [rtuTraversalSetTriangles](#) ([RTUtraversal](#) traversal, unsigned int num\_tris, const float \*tris)
- [RTresult](#) RTAPI [rtuTraversalSetAccelData](#) ([RTUtraversal](#) traversal, const void \*data, RTsize data\_size)
- [RTresult](#) RTAPI [rtuTraversalGetAccelDataSize](#) ([RTUtraversal](#) traversal, RTsize \*data\_size)
- [RTresult](#) RTAPI [rtuTraversalGetAccelData](#) ([RTUtraversal](#) traversal, void \*data)
- [RTresult](#) RTAPI [rtuTraversalMapRays](#) ([RTUtraversal](#) traversal, unsigned int num\_rays, float \*\*rays)
- [RTresult](#) RTAPI [rtuTraversalUnmapRays](#) ([RTUtraversal](#) traversal)
- [RTresult](#) RTAPI [rtuTraversalPreprocess](#) ([RTUtraversal](#) traversal)
- [RTresult](#) RTAPI [rtuTraversalTraverse](#) ([RTUtraversal](#) traversal)
- [RTresult](#) RTAPI [rtuTraversalMapResults](#) ([RTUtraversal](#) traversal, [RTUtraversalresult](#) \*\*results)
- [RTresult](#) RTAPI [rtuTraversalUnmapResults](#) ([RTUtraversal](#) traversal)
- [RTresult](#) RTAPI [rtuTraversalMapOutput](#) ([RTUtraversal](#) traversal, [RTUoutput](#) which, void \*\*output)
- [RTresult](#) RTAPI [rtuTraversalUnmapOutput](#) ([RTUtraversal](#) traversal, [RTUoutput](#) which)
- [RTresult](#) RTAPI [rtuTraversalDestroy](#) ([RTUtraversal](#) traversal)

## 3.22.2 Typedef Documentation

3.22.2.1 typedef struct [RTUtraversal\\_api](#)\* [RTUtraversal](#)

Opaque type. Note that the \*\_api types should never be used directly. Only the typedef target names will be guaranteed to remain unchanged.

## 3.22.3 Enumeration Type Documentation

3.22.3.1 enum [RTUinitoptions](#)

Initialization options (static across life of traversal object).

The [rtuTraverse](#) API supports both running on the CPU and GPU. When [RTU\\_INITOPTION\\_NONE](#) is specified GPU context creation is attempted. If that fails (such as when there isn't an NVIDIA GPU part present, the CPU code path is automatically chosen. Specifying [RTU\\_INITOPTION\\_GPU\\_ONLY](#) or [RTU\\_INITOPTION\\_CPU\\_ONLY](#) will only use the GPU or CPU modes without automatic transitions from one to the other.

[RTU\\_INITOPTION\\_CULL\\_BACKFACE](#) will enable back face culling during intersection.

## Enumerator

- [RTU\\_INITOPTION\\_NONE](#)** No option
- [RTU\\_INITOPTION\\_GPU\\_ONLY](#)** GPU only
- [RTU\\_INITOPTION\\_CPU\\_ONLY](#)** CPU only
- [RTU\\_INITOPTION\\_CULL\\_BACKFACE](#)** Back face culling

3.22.3.2 enum [RTUoption](#)

Runtime options (can be set multiple times for a given traversal object).

## Enumerator

- [RTU\\_OPTION\\_INT\\_NUM\\_THREADS](#)** Number of threads

## 3.22.3.3 enum RTUoutput

RTUoutput requested.

Enumerator

**RTU\_OUTPUT\_NONE** Output None  
**RTU\_OUTPUT\_NORMAL** float3 [x, y, z]  
**RTU\_OUTPUT\_BARYCENTRIC** float2 [alpha, beta] (gamma implicit)  
**RTU\_OUTPUT\_BACKFACING** char [1 | 0]

## 3.22.3.4 enum RTUquerytype

The type of ray query to be performed.

See OptiX Programming Guide for explanation of any vs. closest hit queries. Note that in the case of [RTU\\_QUERY\\_TYPE\\_ANY\\_HIT](#), the `prim_id` and `t` intersection values in [RTUtraversalresult](#) will correspond to the first successful intersection. These values may not be indicative of the closest intersection, only that there was at least one.

Enumerator

**RTU\_QUERY\_TYPE\_ANY\_HIT** Perform any hit calculation  
**RTU\_QUERY\_TYPE\_CLOSEST\_HIT** Perform closest hit calculation  
**RTU\_QUERY\_TYPE\_COUNT** Query type count

## 3.22.3.5 enum RTUrayformat

The input format of the ray vector.

Enumerator

**RTU\_RAYFORMAT\_ORIGIN\_DIRECTION\_TMIN\_TMAX\_INTERLEAVED** Origin Direction Tmin Tmax interleaved  
**RTU\_RAYFORMAT\_ORIGIN\_DIRECTION\_INTERLEAVED** Origin Direction interleaved  
**RTU\_RAYFORMAT\_COUNT** [Ray](#) format count

## 3.22.3.6 enum RTUtriformat

The input format of the triangles.

TRIANGLE\_SOUP implies future use of [rtuTraversalSetTriangles](#) while MESH implies use of [rtuTraversalSetMesh](#).

Enumerator

**RTU\_TRIFORMAT\_MESH** Triangle format mesh  
**RTU\_TRIFORMAT\_TRIANGLE\_SOUP** Triangle 'soup' format  
**RTU\_TRIFORMAT\_COUNT** Triangle format count

## 3.22.4 Function Documentation

## 3.22.4.1 RTresult RTAPI rtuTraversalCreate ( RTUtraversal \* traversal, RTUquerytype query\_type, RTUrayformat ray\_format, RTUtriformat tri\_format, unsigned int outputs, unsigned int options, RTcontext context )

Create a traversal state and associate a context with it. If context is a null pointer a new context will be created internally. The context should also not be used for any other launch commands from the OptiX host API, nor attached to multiple [RTUtraversal](#) objects at one time.

## Parameters

out	<i>traversal</i>	Return pointer for traverse state handle
	<i>query_type</i>	<a href="#">Ray</a> query type
	<i>ray_format</i>	<a href="#">Ray</a> format
	<i>tri_format</i>	Triangle format
	<i>outputs</i>	OR'ed mask of requested <a href="#">RTUoutput</a>
	<i>options</i>	Bit vector of or'ed RTUinitoptions
	<i>context</i>	RTcontext used for internal object creation

3.22.4.2 RTresult RTAPI rtuTraversalDestroy ( RTUtraversal *traversal* )

Clean up any internal memory associated with *rtuTraversal\** operations. Includes destruction of result buffers returned via [rtuTraversalGetErrorString](#). Invalidates traversal object.

## Parameters

<i>traversal</i>	Traversal state handle
------------------	------------------------

3.22.4.3 RTresult RTAPI rtuTraversalGetAccelData ( RTUtraversal *traversal*, void \* *data* )

Retrieve acceleration data for current geometry. Will force acceleration build if necessary. The data parameter should be preallocated and its length should match return value of [rtuTraversalGetAccelDataSize](#).

## Parameters

	<i>traversal</i>	Traversal state handle
out	<i>data</i>	Acceleration data

3.22.4.4 RTresult RTAPI rtuTraversalGetAccelDataSize ( RTUtraversal *traversal*, RTsize \* *data\_size* )

Retrieve acceleration data size for current geometry. Will force acceleration build if necessary.

## Parameters

	<i>traversal</i>	Traversal state handle
out	<i>data_size</i>	Size of acceleration data

3.22.4.5 RTresult RTAPI rtuTraversalGetErrorString ( RTUtraversal *traversal*, RTresult *code*, const char \*\* *return\_string* )

Returns the string associated with the error code and any additional information from the last error. If traversal is non-NULL *return\_string* only remains valid while traversal is live.

For a list of associated error codes that this function might inspect take a look at [RTresult](#).

## Parameters

out	<i>return_string</i>	Pointer to string with error message in it
	<i>traversal</i>	Traversal state handle. Can be NULL
	<i>code</i>	Error code from last error

3.22.4.6 RTresult RTAPI rtuTraversalMapOutput ( RTUtraversal *traversal*, RTUoutput *which*, void \*\* *output* )

Retrieve user-specified output from last [rtuTraversalTraverse](#) call. Output can be copied from the pointer returned by [rtuTraversalMapOutput](#) and will have length '*num\_rays*' from as prescribed from the previous call to [rtuTraversalMapRays](#). For each [RTUoutput](#), a single [rtuTraversalMapOutput](#) pointers can be outstanding. [rtuTraversalUnmapOutput](#) should be called when finished reading the output.

If requested output type was not turned on with a previous call to [rtuTraversalCreate](#) an error will be returned. See [RTUoutput](#) enum for description of output data formats for various outputs.

## Parameters

	<i>traversal</i>	Traversal state handle
	<i>which</i>	Output type to be specified
out	<i>output</i>	Pointer to output from last traverse

3.22.4.7 RTResult RTAPI rtuTraversalMapRays ( RTUtraversal *traversal*, unsigned int *num\_rays*, float \*\* *rays* )

Specify set of rays to be cast upon next call to [rtuTraversalTraverse](#). [rtuTraversalMapRays](#) obtains a pointer which can be used to copy the ray data into. Rays should be packed in the format described in [rtuTraversalCreate](#) call. When copying is completed [rtuTraversalUnmapRays](#) should be called. Note that this call invalidates any existing results buffers until [rtuTraversalTraverse](#) is called again.

## Parameters

	<i>traversal</i>	Traversal state handle
	<i>num_rays</i>	Number of rays to be traced
	<i>rays</i>	Pointer to ray data

3.22.4.8 RTResult RTAPI rtuTraversalMapResults ( RTUtraversal *traversal*, RTUtraversalresult \*\* *results* )

Retrieve results of last [rtuTraversal](#) call. Results can be copied from the pointer returned by [rtuTraversalMapResults](#) and will have length '*num\_rays*' as prescribed from the previous call to [rtuTraversalMapRays](#). [rtuTraversalUnmapResults](#) should be called when finished reading the results. Returned primitive ID of -1 indicates a ray miss.

## Parameters

	<i>traversal</i>	Traversal state handle
out	<i>results</i>	Pointer to results of last traverse

3.22.4.9 RTResult RTAPI rtuTraversalPreprocess ( RTUtraversal *traversal* )

Perform any necessary preprocessing (eg, acceleration structure building, optix context compilation). It is not necessary to call this function as [rtuTraversalTraverse](#) will call this internally as necessary.

## Parameters

	<i>traversal</i>	Traversal state handle
--	------------------	------------------------

3.22.4.10 RTResult RTAPI rtuTraversalSetAccelData ( RTUtraversal *traversal*, const void \* *data*, RTsize *data\_size* )

Specify acceleration data for current geometry. Input acceleration data should be result of [rtuTraversalGetAccelData](#) or [rtAccelerationGetData](#) call.

## Parameters

	<i>traversal</i>	Traversal state handle
	<i>data</i>	Acceleration data
	<i>data_size</i>	Size of acceleration data

3.22.4.11 RTResult RTAPI rtuTraversalSetMesh ( RTUtraversal *traversal*, unsigned int *num\_verts*, const float \* *verts*, unsigned int *num\_tris*, const unsigned \* *indices* )

Specify triangle mesh to be intersected by the next call to [rtuTraversalTraverse](#). Only one geometry set may be active at a time. Subsequent calls to [rtuTraversalSetTriangles](#) or [rtuTraversalSetMesh](#) will override any previously specified geometry. No internal copies of the mesh data are made. The user should ensure that the mesh data remains valid until after [rtuTraversalTraverse](#) has been called. Counter-clockwise winding is assumed for normal and backfacing computations.

## Parameters

<i>traversal</i>	Traversal state handle
<i>num_verts</i>	Vertex count
<i>verts</i>	Vertices [ v1_x, v1_y, v1_z, v2.x, ... ]
<i>num_tris</i>	Triangle count
<i>indices</i>	Indices [ tri1_index1, tri1_index2, ... ]

3.22.4.12 RTresult RTAPI rtuTraversalSetOption ( RTUtraversal *traversal*, RTUoption *option*, void \* *value* )

Set a runtime option. Unlike initialization options, these options may be set more than once for a given [RTUtraversal](#) instance.

## Parameters

<i>traversal</i>	Traversal state handle
<i>option</i>	The option to be set
<i>value</i>	Value of the option

3.22.4.13 RTresult RTAPI rtuTraversalSetTriangles ( RTUtraversal *traversal*, unsigned int *num\_tris*, const float \* *tris* )

Specify triangle soup to be intersected by the next call to `rtuTraversalLaunch`. Only one geometry set may be active at a time. Subsequent calls to [rtuTraversalSetTriangles](#) or [rtuTraversalSetMesh](#) will override any previously specified geometry. No internal copies of the triangle data are made. The user should ensure that the triangle data remains valid until after [rtuTraversalTraverse](#) has been called. Counter-clockwise winding is assumed for normal and backfacing computations.

## Parameters

<i>traversal</i>	Traversal state handle
<i>num_tris</i>	Triangle count
<i>tris</i>	Triangles [ tri1_v1.x, tri1_v1.y, tri1_v1.z, tri1_v2.x, ... ]

3.22.4.14 RTresult RTAPI rtuTraversalTraverse ( RTUtraversal *traversal* )

Perform any necessary preprocessing (eg, acceleration structure building and kernel compilation ) and cast current rays against current geometry.

## Parameters

<i>traversal</i>	Traversal state handle
------------------	------------------------

3.22.4.15 RTresult RTAPI rtuTraversalUnmapOutput ( RTUtraversal *traversal*, RTUoutput *which* )

See [rtuTraversalMapOutput](#) .

3.22.4.16 RTresult RTAPI rtuTraversalUnmapRays ( RTUtraversal *traversal* )

See [rtuTraversalMapRays](#) .

3.22.4.17 RTresult RTAPI rtuTraversalUnmapResults ( RTUtraversal *traversal* )

See [rtuTraversalMapResults](#) .

## Index

- ~Exception
  - optix::Exception, [24](#)
- \_OPTIX\_SEMANTIC\_TYPE\_CurrentRay
  - optix\_defines.h, [89](#)
- \_OPTIX\_SEMANTIC\_TYPE\_IntersectionDistance
  - optix\_defines.h, [89](#)
- \_OPTIX\_SEMANTIC\_TYPE\_LaunchIndex
  - optix\_defines.h, [89](#)
- Aabb
  - optix::Aabb, [5](#)
- area
  - optix::Aabb, [5](#)
- bindProgressiveStream
  - optix::BufferObj, [13](#)
- center
  - optix::Aabb, [5](#)
- checkError
  - optix::APIObj, [9](#)
  - optix::ContextObj, [19](#)
- contains
  - optix::Aabb, [5](#)
- create1DLayeredBuffer
  - optix::ContextObj, [19](#)
- create2DLayeredBuffer
  - optix::ContextObj, [19](#)
- createAcceleration
  - optix::ContextObj, [19](#)
- createBuffer
  - optix::ContextObj, [19](#)
- createBufferForCUDA
  - optix::ContextObj, [19](#)
- createCubeBuffer
  - optix::ContextObj, [19](#)
- createCubeLayeredBuffer
  - optix::ContextObj, [20](#)
- createGeometryGroup
  - optix::ContextObj, [20](#)
- createGeometryInstance
  - optix::ContextObj, [20](#)
- createGroup
  - optix::ContextObj, [20](#)
- createMipmappedBuffer
  - optix::ContextObj, [20](#)
- DXGI Texture Formats, [3](#)
- declareVariable
  - optix::ContextObj, [20](#)
  - optix::GeometryInstanceObj, [26](#)
  - optix::GeometryObj, [27](#)
  - optix::MaterialObj, [32](#)
  - optix::ProgramObj, [38](#)
  - optix::ScopedObj, [44](#)
- det
  - optix::Matrix, [34](#)
- Direct3D Texture Formats, [2](#)
- direction
  - Ray, [40](#)
- distance
  - optix::Aabb, [5](#)
- distance2
  - optix::Aabb, [5](#)
- enlarge
  - optix::Aabb, [5](#)
- extent
  - optix::Aabb, [5](#)
- getBufferFromId
  - optix::ContextObj, [20](#)
- getCol
  - optix::Matrix, [34](#)
- getContext
  - optix::ContextObj, [20](#)
- getData
  - optix::Matrix, [34](#)
- getDataSize
  - optix::AccelerationObj, [8](#)
- getDevicePointer
  - optix::BufferObj, [13](#)
- getEnabledDeviceCount
  - optix::ContextObj, [21](#)
- getGLBOld
  - optix::BufferObj, [14](#)
- getId
  - optix::BufferObj, [14](#)
  - optix::ProgramObj, [38](#)
  - optix::TextureSamplerObj, [47](#)
- getMaxTextureCount
  - optix::ContextObj, [21](#)
- getPrimitiveCount
  - optix::GeometryObj, [27](#)
- getPrimitiveIndexOffset
  - optix::GeometryObj, [28](#)
- getProgramFromId
  - optix::ContextObj, [21](#)
- getProperty
  - optix::AccelerationObj, [8](#)
- getRow
  - optix::Matrix, [34](#)
- getTextureSamplerFromId
  - optix::ContextObj, [21](#)
- getVariableCount
  - optix::ContextObj, [21](#)
  - optix::GeometryInstanceObj, [26](#)
  - optix::GeometryObj, [28](#)
  - optix::MaterialObj, [32](#)
  - optix::ProgramObj, [38](#)
  - optix::ScopedObj, [44](#)

- halfArea
  - optix::Aabb, [5](#)
- IDirect3DDevice9
  - optix\_d3d9\_interop.h, [71](#)
- IDirect3DResource9
  - optix\_d3d9\_interop.h, [71](#)
- identity
  - optix::Matrix, [34](#)
- include
  - optix::Aabb, [5](#), [6](#)
- intersection
  - optix::Aabb, [6](#)
- intersects
  - optix::Aabb, [6](#)
- invalidate
  - optix::Aabb, [6](#)
- inverse
  - optix::Matrix, [34](#)
- isFlat
  - optix::Aabb, [6](#)
- launch
  - optix::ContextObj, [21](#)
- launchProgressive
  - optix::ContextObj, [21](#)
- longestAxis
  - optix::Aabb, [6](#)
- m\_max
  - optix::Aabb, [7](#)
- m\_min
  - optix::Aabb, [7](#)
- makeException
  - optix::Exception, [24](#)
- map
  - optix::BufferObj, [14](#)
- markDirty
  - optix::AccelerationObj, [8](#)
  - optix::GeometryObj, [28](#)
- Matrix
  - optix::Matrix, [33](#), [34](#)
- maxExtent
  - optix::Aabb, [6](#)
- OPTIX\_VERSION
  - optix.h, [55](#)
- OpenGL Texture Formats, [1](#)
- operator<
  - optix::Matrix, [34](#)
- operator=
  - optix::Matrix, [34](#)
- operator==
  - optix::Aabb, [6](#)
- optix.h, [55](#)
  - OPTIX\_VERSION, [55](#)
- optix\_declarations.h
  - RT\_BUFFER\_ATTRIBUTE\_STREAM\_BITRATE, [82](#)
  - RT\_BUFFER\_ATTRIBUTE\_STREAM\_FORMAT, [82](#)
  - RT\_BUFFER\_ATTRIBUTE\_STREAM\_FPS, [82](#)
  - RT\_BUFFER\_ATTRIBUTE\_STREAM\_GAMMA, [82](#)
  - RT\_BUFFER\_COPY\_ON\_DIRTY, [82](#)
  - RT\_BUFFER\_CUBEMAP, [82](#)
  - RT\_BUFFER\_GPU\_LOCAL, [82](#)
  - RT\_BUFFER\_ID\_NULL, [82](#)
  - RT\_BUFFER\_INPUT, [83](#)
  - RT\_BUFFER\_INPUT\_OUTPUT, [83](#)
  - RT\_BUFFER\_LAYERED, [82](#)
  - RT\_BUFFER\_MAP\_READ, [83](#)
  - RT\_BUFFER\_MAP\_READ\_WRITE, [83](#)
  - RT\_BUFFER\_MAP\_WRITE, [83](#)
  - RT\_BUFFER\_MAP\_WRITE\_DISCARD, [83](#)
  - RT\_BUFFER\_OUTPUT, [83](#)
  - RT\_BUFFER\_PROGRESSIVE\_STREAM, [83](#)
  - RT\_CONTEXT\_ATTRIBUTE\_AVAILABLE\_DEVICE\_MEMORY, [83](#)
  - RT\_CONTEXT\_ATTRIBUTE\_CPU\_NUM\_THREADS, [83](#)
  - RT\_CONTEXT\_ATTRIBUTE\_GPU\_PAGING\_ACTIVE, [83](#)
  - RT\_CONTEXT\_ATTRIBUTE\_GPU\_PAGING\_FORCED\_OFF, [83](#)
  - RT\_CONTEXT\_ATTRIBUTE\_MAX\_TEXTURE\_COUNT, [83](#)
  - RT\_CONTEXT\_ATTRIBUTE\_USED\_HOST\_MEMORY, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_CLOCK\_RATE, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_COMPUTE\_CAPABILITY, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_CUDA\_DEVICE\_ORDINAL, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_EXECUTION\_TIMEOUT\_ENABLED, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_MAX\_HARDWARE\_TEXTURE\_COUNT, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_MAX\_THREADS\_PER\_BLOCK, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_MULTIPROCESSOR\_COUNT, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_NAME, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_TCC\_DRIVER, [83](#)
  - RT\_DEVICE\_ATTRIBUTE\_TOTAL\_MEMORY, [83](#)
  - RT\_ERROR\_ALREADY\_MAPPED, [87](#)
  - RT\_ERROR\_AUTHENTICATION\_FAILED, [87](#)
  - RT\_ERROR\_CLUSTER\_ALREADY\_RUNNING, [87](#)
  - RT\_ERROR\_CLUSTER\_NOT\_RUNNING, [87](#)
  - RT\_ERROR\_CONNECTION\_ALREADY\_EXISTS, [87](#)
  - RT\_ERROR\_CONNECTION\_FAILED, [87](#)
  - RT\_ERROR\_CONTEXT\_CREATION\_FAILED, [87](#)
  - RT\_ERROR\_FILE\_NOT\_FOUND, [87](#)
  - RT\_ERROR\_ILLEGAL\_SYMBOL, [87](#)
  - RT\_ERROR\_INSUFFICIENT\_FREE\_NODES, [87](#)



RT\_ERROR\_INVALID\_CONTEXT, 87  
RT\_ERROR\_INVALID\_DEVICE, 87  
RT\_ERROR\_INVALID\_DRIVER\_VERSION, 87  
RT\_ERROR\_INVALID\_IMAGE, 87  
RT\_ERROR\_INVALID\_SOURCE, 87  
RT\_ERROR\_INVALID\_VALUE, 87  
RT\_ERROR\_LAUNCH\_FAILED, 87  
RT\_ERROR\_MEMORY\_ALLOCATION\_FAILED, 87  
RT\_ERROR\_NETWORK\_INIT\_FAILED, 87  
RT\_ERROR\_NETWORK\_LOAD\_FAILED, 87  
RT\_ERROR\_NO\_DEVICE, 87  
RT\_ERROR\_NOT\_SUPPORTED, 87  
RT\_ERROR\_OBJECT\_CREATION\_FAILED, 87  
RT\_ERROR\_RESOURCE\_ALREADY\_REGISTERED, 87  
RT\_ERROR\_RESOURCE\_NOT\_REGISTERED, 87  
RT\_ERROR\_TYPE\_MISMATCH, 87  
RT\_ERROR\_UNKNOWN, 87  
RT\_ERROR\_VARIABLE\_NOT\_FOUND, 87  
RT\_ERROR\_VARIABLE\_REDECLARED, 87  
RT\_ERROR\_VERSION\_MISMATCH, 87  
RT\_EXCEPTION\_ALL, 84  
RT\_EXCEPTION\_BUFFER\_ID\_INVALID, 84  
RT\_EXCEPTION\_BUFFER\_INDEX\_OUT\_OF\_BOUNDS, 84  
RT\_EXCEPTION\_INDEX\_OUT\_OF\_BOUNDS, 84  
RT\_EXCEPTION\_INTERNAL\_ERROR, 84  
RT\_EXCEPTION\_INVALID\_RAY, 84  
RT\_EXCEPTION\_PROGRAM\_ID\_INVALID, 84  
RT\_EXCEPTION\_STACK\_OVERFLOW, 84  
RT\_EXCEPTION\_TEXTURE\_ID\_INVALID, 84  
RT\_EXCEPTION\_USER, 84  
RT\_FILTER\_LINEAR, 84  
RT\_FILTER\_NEAREST, 84  
RT\_FILTER\_NONE, 84  
RT\_FORMAT\_BUFFER\_ID, 85  
RT\_FORMAT\_BYTE, 84  
RT\_FORMAT\_BYTE2, 84  
RT\_FORMAT\_BYTE3, 84  
RT\_FORMAT\_BYTE4, 84  
RT\_FORMAT\_FLOAT, 84  
RT\_FORMAT\_FLOAT2, 84  
RT\_FORMAT\_FLOAT3, 84  
RT\_FORMAT\_FLOAT4, 84  
RT\_FORMAT\_HALF, 85  
RT\_FORMAT\_HALF2, 85  
RT\_FORMAT\_HALF3, 85  
RT\_FORMAT\_HALF4, 85  
RT\_FORMAT\_INT, 85  
RT\_FORMAT\_INT2, 85  
RT\_FORMAT\_INT3, 85  
RT\_FORMAT\_INT4, 85  
RT\_FORMAT\_PROGRAM\_ID, 85  
RT\_FORMAT\_SHORT, 84  
RT\_FORMAT\_SHORT2, 84  
RT\_FORMAT\_SHORT3, 84  
RT\_FORMAT\_SHORT4, 84  
RT\_FORMAT\_UNKNOWN, 84  
RT\_FORMAT\_UNSIGNED\_BYTE, 84  
RT\_FORMAT\_UNSIGNED\_BYTE2, 84  
RT\_FORMAT\_UNSIGNED\_BYTE3, 84  
RT\_FORMAT\_UNSIGNED\_BYTE4, 84  
RT\_FORMAT\_UNSIGNED\_INT, 85  
RT\_FORMAT\_UNSIGNED\_INT2, 85  
RT\_FORMAT\_UNSIGNED\_INT3, 85  
RT\_FORMAT\_UNSIGNED\_INT4, 85  
RT\_FORMAT\_UNSIGNED\_SHORT, 84  
RT\_FORMAT\_UNSIGNED\_SHORT2, 84  
RT\_FORMAT\_UNSIGNED\_SHORT3, 85  
RT\_FORMAT\_UNSIGNED\_SHORT4, 85  
RT\_FORMAT\_USER, 85  
RT\_OBJECTTYPE\_BUFFER, 85  
RT\_OBJECTTYPE\_FLOAT, 86  
RT\_OBJECTTYPE\_FLOAT2, 86  
RT\_OBJECTTYPE\_FLOAT3, 86  
RT\_OBJECTTYPE\_FLOAT4, 86  
RT\_OBJECTTYPE\_GEOMETRY\_GROUP, 85  
RT\_OBJECTTYPE\_GEOMETRY\_INSTANCE, 85  
RT\_OBJECTTYPE\_GROUP, 85  
RT\_OBJECTTYPE\_INT, 86  
RT\_OBJECTTYPE\_INT2, 86  
RT\_OBJECTTYPE\_INT3, 86  
RT\_OBJECTTYPE\_INT4, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT2x2, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT2x3, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT2x4, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT3x2, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT3x3, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT3x4, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT4x2, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT4x3, 86  
RT\_OBJECTTYPE\_MATRIX\_FLOAT4x4, 86  
RT\_OBJECTTYPE\_OBJECT, 85  
RT\_OBJECTTYPE\_PROGRAM, 86  
RT\_OBJECTTYPE\_SELECTOR, 85  
RT\_OBJECTTYPE\_TEXTURE\_SAMPLER, 85  
RT\_OBJECTTYPE\_TRANSFORM, 85  
RT\_OBJECTTYPE\_UNKNOWN, 85  
RT\_OBJECTTYPE\_UNSIGNED\_INT, 86  
RT\_OBJECTTYPE\_UNSIGNED\_INT2, 86  
RT\_OBJECTTYPE\_UNSIGNED\_INT3, 86  
RT\_OBJECTTYPE\_UNSIGNED\_INT4, 86  
RT\_OBJECTTYPE\_USER, 86  
RT\_PROGRAM\_ID\_NULL, 86  
RT\_REMOTEDEVICE\_ATTRIBUTE\_CLUSTER\_URL, 86  
RT\_REMOTEDEVICE\_ATTRIBUTE\_CONFIGURATIONS, 86  
RT\_REMOTEDEVICE\_ATTRIBUTE\_GPU\_TOTAL\_MEMORY, 86  
RT\_REMOTEDEVICE\_ATTRIBUTE\_HEAD\_NODE\_URL, 86  
RT\_REMOTEDEVICE\_ATTRIBUTE\_NAME, 86

- RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_CONFIGURATIONS, [86](#)
- RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_FREE\_NODES, [86](#)
- RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_GPUS, [86](#)
- RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_RESERVED\_NODES, [86](#)
- RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_TOTAL\_NODES, [86](#)
- RT\_REMOTEDEVICE\_ATTRIBUTE\_STATUS, [86](#)
- RT\_REMOTEDEVICE\_STATUS\_CONNECTED, [87](#)
- RT\_REMOTEDEVICE\_STATUS\_DISCONNECTED, [87](#)
- RT\_REMOTEDEVICE\_STATUS\_READY, [87](#)
- RT\_REMOTEDEVICE\_STATUS\_RESERVED, [87](#)
- RT\_SUCCESS, [87](#)
- RT\_TARGET\_GL\_RENDER\_BUFFER, [85](#)
- RT\_TARGET\_GL\_TEXTURE\_1D, [85](#)
- RT\_TARGET\_GL\_TEXTURE\_1D\_ARRAY, [85](#)
- RT\_TARGET\_GL\_TEXTURE\_2D, [85](#)
- RT\_TARGET\_GL\_TEXTURE\_2D\_ARRAY, [85](#)
- RT\_TARGET\_GL\_TEXTURE\_3D, [85](#)
- RT\_TARGET\_GL\_TEXTURE\_CUBE\_MAP, [85](#)
- RT\_TARGET\_GL\_TEXTURE\_CUBE\_MAP\_ARRAY, [85](#)
- RT\_TARGET\_GL\_TEXTURE\_RECTANGLE, [85](#)
- RT\_TEXTURE\_ID\_NULL, [88](#)
- RT\_TEXTURE\_INDEX\_ARRAY\_INDEX, [88](#)
- RT\_TEXTURE\_INDEX\_NORMALIZED\_COORDINATES, [88](#)
- RT\_TEXTURE\_READ\_ELEMENT\_TYPE, [88](#)
- RT\_TEXTURE\_READ\_ELEMENT\_TYPE\_SRGB, [88](#)
- RT\_TEXTURE\_READ\_NORMALIZED\_FLOAT, [88](#)
- RT\_TEXTURE\_READ\_NORMALIZED\_FLOAT\_SRGB, [88](#)
- RT\_TIMEOUT\_CALLBACK, [87](#)
- RT\_WRAP\_CLAMP\_TO\_BORDER, [88](#)
- RT\_WRAP\_CLAMP\_TO\_EDGE, [88](#)
- RT\_WRAP\_MIRROR, [88](#)
- RT\_WRAP\_REPEAT, [88](#)
- optix\_defines.h
  - \_OPTIX\_SEMANTIC\_TYPE\_CurrentRay, [89](#)
  - \_OPTIX\_SEMANTIC\_TYPE\_IntersectionDistance, [89](#)
  - \_OPTIX\_SEMANTIC\_TYPE\_LaunchIndex, [89](#)
  - RT\_INTERNAL\_INVERSE\_TRANSPOSE, [89](#)
  - RT\_OBJECT\_TO\_WORLD, [90](#)
  - RT\_WORLD\_TO\_OBJECT, [90](#)
- optix\_prime\_declarations.h
  - RTP\_BUFFER\_FORMAT\_HIT\_BITMASK, [282](#)
  - RTP\_BUFFER\_FORMAT\_HIT\_T, [282](#)
  - RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID, [282](#)
  - RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_INSTID, [282](#)
  - RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_INSTID\_U\_V, [283](#)
  - RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_U\_V, [282](#)
  - RTP\_BUFFER\_FORMAT\_INDICES\_INT3, [282](#)
  - RTP\_BUFFER\_FORMAT\_INDICES\_INT3\_MASK\_INT, [282](#)
  - RTP\_BUFFER\_FORMAT\_INSTANCE\_MODEL, [283](#)
  - RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_DIRECTION, [282](#)
  - RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_MASK\_DIRECTION\_TMAX, [282](#)
  - RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_TMIN\_DIRECTION\_TMAX, [282](#)
  - RTP\_BUFFER\_FORMAT\_TRANSFORM\_FLOAT4x3, [283](#)
  - RTP\_BUFFER\_FORMAT\_TRANSFORM\_FLOAT4x4, [283](#)
  - RTP\_BUFFER\_FORMAT\_VERTEX\_FLOAT3, [282](#)
  - RTP\_BUFFER\_FORMAT\_VERTEX\_FLOAT4, [282](#)
  - RTP\_BUFFER\_TYPE\_CUDA\_LINEAR, [283](#)
  - RTP\_BUFFER\_TYPE\_HOST, [283](#)
  - RTP\_BUILDER\_PARAM\_CHUNK\_SIZE, [283](#)
  - RTP\_BUILDER\_PARAM\_USE\_CALLER\_TRIANGLES, [283](#)
  - RTP\_CONTEXT\_TYPE\_CPU, [283](#)
  - RTP\_CONTEXT\_TYPE\_CUDA, [283](#)
  - RTP\_ERROR\_INVALID\_CONTEXT, [284](#)
  - RTP\_ERROR\_INVALID\_HANDLE, [284](#)
  - RTP\_ERROR\_INVALID\_OPERATION, [284](#)
  - RTP\_ERROR\_INVALID\_VALUE, [284](#)
  - RTP\_ERROR\_MEMORY\_ALLOCATION\_FAILED, [284](#)
  - RTP\_ERROR\_NOT\_SUPPORTED, [284](#)
  - RTP\_ERROR\_OBJECT\_CREATION\_FAILED, [284](#)
  - RTP\_ERROR\_OUT\_OF\_MEMORY, [284](#)
  - RTP\_ERROR\_UNKNOWN, [284](#)
  - RTP\_ERROR\_VALIDATION\_ERROR, [284](#)
  - RTP\_MODEL\_HINT\_ASYNC, [283](#)
  - RTP\_MODEL\_HINT\_MASK\_UPDATE, [283](#)
  - RTP\_MODEL\_HINT\_NONE, [283](#)
  - RTP\_MODEL\_HINT\_USER\_TRIANGLES\_AFTER\_COPY\_SET, [283](#)
  - RTP\_QUERY\_HINT\_ASYNC, [283](#)
  - RTP\_QUERY\_HINT\_NONE, [283](#)
  - RTP\_QUERY\_HINT\_WATERTIGHT, [283](#)
  - RTP\_QUERY\_TYPE\_ANY, [284](#)
  - RTP\_QUERY\_TYPE\_CLOSEST, [284](#)
  - RTP\_SUCCESS, [284](#)
- optix::APIObj, [8](#)
  - checkError, [9](#)
- optix::Aabb, [4](#)
  - Aabb, [5](#)
  - area, [5](#)
  - center, [5](#)
  - contains, [5](#)
  - distance, [5](#)

- distance2, 5
- enlarge, 5
- extent, 5
- halfArea, 5
- include, 5, 6
- intersection, 6
- intersects, 6
- invalidate, 6
- isFlat, 6
- longestAxis, 6
- m\_max, 7
- m\_min, 7
- maxExtent, 6
- operator==, 6
- set, 6
- signedDistance, 6
- valid, 6
- volume, 7
- optix::AccelerationObj, 7
  - getDataSize, 8
  - getProperty, 8
  - markDirty, 8
  - setProperty, 8
- optix::BufferObj, 12
  - bindProgressiveStream, 13
  - getDevicePointer, 13
  - getGLBOld, 14
  - getIdx, 14
  - map, 14
  - registerD3D9Buffer, 14
  - setAttribute, 14
  - setFormat, 14
  - setSize, 14
- optix::ContextObj, 16
  - checkError, 19
  - create1DLayeredBuffer, 19
  - create2DLayeredBuffer, 19
  - createAcceleration, 19
  - createBuffer, 19
  - createBufferForCUDA, 19
  - createCubeBuffer, 19
  - createCubeLayeredBuffer, 20
  - createGeometryGroup, 20
  - createGeometryInstance, 20
  - createGroup, 20
  - createMipmappedBuffer, 20
  - declareVariable, 20
  - getBufferFromIdx, 20
  - getContext, 20
  - getEnabledDeviceCount, 21
  - getMaxTextureCount, 21
  - getProgramFromIdx, 21
  - getTextureSamplerFromIdx, 21
  - getVariableCount, 21
  - launch, 21
  - launchProgressive, 21
  - queryVariable, 21
  - setCPUNumThreads, 21
  - setDevices, 21
  - setPrintEnabled, 21
  - setRayGenerationProgram, 21
  - setStackSize, 21
  - setTimeoutCallback, 21
- optix::DestroyableObj, 22
- optix::Exception, 23
  - ~Exception, 24
  - makeException, 24
- optix::GeometryGroupObj, 24
  - removeChild, 25
  - setAcceleration, 25
  - setChildCount, 25
- optix::GeometryInstanceObj, 25
  - declareVariable, 26
  - getVariableCount, 26
  - queryVariable, 26
  - setGeometry, 26
- optix::GeometryObj, 26
  - declareVariable, 27
  - getPrimitiveCount, 27
  - getPrimitiveIndexOffset, 28
  - getVariableCount, 28
  - markDirty, 28
  - queryVariable, 28
  - setBoundingBoxProgram, 28
  - setPrimitiveCount, 28
  - setPrimitiveIndexOffset, 28
- optix::GroupObj, 28
  - removeChild, 29
  - setAcceleration, 29
  - setChildCount, 29
- optix::Handle
  - take, 31
- optix::Handle< T >, 30
- optix::MaterialObj, 31
  - declareVariable, 32
  - getVariableCount, 32
  - queryVariable, 32
  - setClosestHitProgram, 32
- optix::Matrix
  - det, 34
  - getCol, 34
  - getData, 34
  - getRow, 34
  - identity, 34
  - inverse, 34
  - Matrix, 33, 34
  - operator<, 34
  - operator=, 34
  - rotate, 35
  - scale, 35
  - setCol, 35
  - setRow, 35
  - translate, 35
  - transpose, 35
- optix::Matrix< M, N >, 32
- optix::Onb, 37

optix::ProgramObj, 37  
     declareVariable, 38  
     getId, 38  
     getVariableCount, 38  
     queryVariable, 38  
 optix::RemoteDeviceObj, 40  
 optix::ScopedObj, 44  
     declareVariable, 44  
     getVariableCount, 44  
     queryVariable, 44  
 optix::SelectorObj, 45  
     removeChild, 46  
     setChildCount, 46  
     setVisitProgram, 46  
 optix::TextureSamplerObj, 46  
     getId, 47  
     registerD3D9Texture, 47  
     registerGLTexture, 47  
     setBuffer, 47  
     setMipLevelCount, 48  
 optix::TransformObj, 48  
     setChild, 48  
     setMatrix, 48  
 optix::VariableObj, 49  
 optix::VectorDim< 2 >, 51  
 optix::VectorDim< 3 >, 52  
 optix::VectorDim< 4 >, 52  
 optix::VectorDim< DIM >, 51  
 optix::VectorTypes< float, 1 >, 52  
 optix::VectorTypes< float, 2 >, 52  
 optix::VectorTypes< float, 3 >, 52  
 optix::VectorTypes< float, 4 >, 53  
 optix::VectorTypes< int, 1 >, 53  
 optix::VectorTypes< int, 2 >, 53  
 optix::VectorTypes< int, 3 >, 53  
 optix::VectorTypes< int, 4 >, 53  
 optix::VectorTypes< T, Dim >, 52  
 optix::VectorTypes< unsigned int, 1 >, 54  
 optix::VectorTypes< unsigned int, 2 >, 54  
 optix::VectorTypes< unsigned int, 3 >, 54  
 optix::VectorTypes< unsigned int, 4 >, 54  
 optix::boundCallableProgramId< T >, 9  
 optix::buffer< T, Dim >, 9  
 optix::buffer< T, Dim >::type< T2 >, 49  
 optix::bufferId< T, Dim >, 11  
 optix::callableProgramId< T >, 14  
 optix::prime::BufferDescObj, 10  
 optix::prime::ContextObj, 15  
 optix::prime::Exception, 23  
 optix::prime::ModelObj, 35  
     setBuilderParameter, 36  
     setInstances, 36  
     setTriangles, 36, 37  
 optix::prime::QueryObj, 38  
 optix::rt\_print\_t< double >, 41  
 optix::rt\_print\_t< float >, 41  
 optix::rt\_print\_t< long long >, 41  
 optix::rt\_print\_t< T >, 41  
 optix::rt\_print\_t< unsigned long long >, 41  
 optix\_cuda\_interop.h, 55  
     rtBufferCreateForCUDA, 56  
     rtBufferGetDevicePointer, 57  
     rtBufferMarkDirty, 57  
     rtBufferSetDevicePointer, 58  
 optix\_d3d10\_interop.h, 58  
     rtBufferCreateFromD3D10Resource, 59  
     rtBufferD3D10Register, 60  
     rtBufferD3D10Unregister, 60  
     rtBufferGetD3D10Resource, 61  
     rtContextSetD3D10Device, 61  
     rtDeviceGetD3D10Device, 62  
     rtTextureSamplerCreateFromD3D10Resource, 62  
     rtTextureSamplerD3D10Register, 63  
     rtTextureSamplerD3D10Unregister, 64  
     rtTextureSamplerGetD3D10Resource, 64  
 optix\_d3d11\_interop.h, 65  
     rtBufferCreateFromD3D11Resource, 65  
     rtBufferD3D11Register, 66  
     rtBufferD3D11Unregister, 66  
     rtBufferGetD3D11Resource, 67  
     rtContextSetD3D11Device, 67  
     rtDeviceGetD3D11Device, 68  
     rtTextureSamplerCreateFromD3D11Resource, 68  
     rtTextureSamplerD3D11Register, 69  
     rtTextureSamplerD3D11Unregister, 70  
     rtTextureSamplerGetD3D11Resource, 70  
 optix\_d3d9\_interop.h, 71  
     IDirect3DDevice9, 71  
     IDirect3DResource9, 71  
     rtBufferCreateFromD3D9Resource, 71  
     rtBufferD3D9Register, 72  
     rtBufferD3D9Unregister, 73  
     rtBufferGetD3D9Resource, 73  
     rtContextSetD3D9Device, 74  
     rtDeviceGetD3D9Device, 74  
     rtTextureSamplerCreateFromD3D9Resource, 75  
     rtTextureSamplerD3D9Register, 75  
     rtTextureSamplerD3D9Unregister, 76  
     rtTextureSamplerGetD3D9Resource, 76  
 optix\_datatypes.h, 77  
     RT\_DEFAULT\_MAX, 77  
 optix\_declarations.h, 78  
     RTbufferattribute, 82  
     RTbufferflag, 82  
     RTbufferidnull, 82  
     RTbuffermapflag, 82  
     RTbuffertype, 83  
     RTcontextattribute, 83  
     RTdeviceattribute, 83  
     RTexception, 83  
     RTfiltermode, 84  
     RTformat, 84  
     RTgltarget, 85  
     RTobjecttype, 85  
     RTprogramidnull, 86  
     RTremotedeviceattribute, 86

- RTremotedevicestatus, [86](#)
- RTresult, [87](#)
- RTtextureidnull, [87](#)
- RTtextureindexmode, [88](#)
- RTtexturereadmode, [88](#)
- RTwrapmode, [88](#)
- optix\_defines.h, [88](#)
  - RTtransformflags, [89](#)
  - RTtransformkind, [89](#)
  - rtSemanticTypes, [89](#)
- optix\_device.h, [90](#)
  - RT\_PROGRAM, [97](#)
  - rtBuffer, [97](#)
  - rtBufferId, [98](#)
  - rtCallableProgram, [98](#)
  - rtCallableProgramId, [98](#)
  - rtCallableProgramX, [99](#)
  - rtDeclareAnnotation, [99](#)
  - rtDeclareVariable, [100](#)
  - rtGetExceptionCode, [101](#)
  - rtGetTransform, [102](#)
  - rtIgnoreIntersection, [102](#)
  - rtIntersectChild, [102](#)
  - rtPotentialIntersection, [103](#)
  - rtPrintExceptionDetails, [103](#)
  - rtPrintf, [105–108](#)
  - rtReportIntersection, [109](#)
  - rtTerminateRay, [109](#)
  - rtTextureSampler, [101](#)
  - rtThrow, [109](#)
  - rtTrace, [110](#)
  - rtTransformNormal, [110](#)
  - rtTransformPoint, [111](#)
  - rtTransformVector, [111](#)
- optix\_gl\_interop.h, [112](#)
  - rtBufferCreateFromGLBO, [112](#)
  - rtBufferGLRegister, [114](#)
  - rtBufferGLUnregister, [114](#)
  - rtBufferGetGLBOId, [113](#)
  - rtDeviceGetWGLDevice, [115](#)
  - rtTextureSamplerCreateFromGLImage, [115](#)
  - rtTextureSamplerGLRegister, [117](#)
  - rtTextureSamplerGLUnregister, [117](#)
  - rtTextureSamplerGetGLImageId, [116](#)
- optix\_host.h, [118](#)
  - RTacceleration, [125](#)
  - RTbuffer, [125](#)
  - RTcontext, [125](#)
  - RTgeometry, [125](#)
  - RTgeometrygroup, [126](#)
  - RTgeometryinstance, [126](#)
  - RTgroup, [126](#)
  - RTmaterial, [126](#)
  - RTobject, [126](#)
  - RTprogram, [126](#)
  - RTremotedevice, [126](#)
  - RTselector, [126](#)
  - RTtexturesampler, [126](#)
  - RTtimeoutcallback, [126](#)
  - RTtransform, [126](#)
  - RTvariable, [126](#)
  - rtAccelerationCreate, [127](#)
  - rtAccelerationDestroy, [127](#)
  - rtAccelerationGetBuilder, [128](#)
  - rtAccelerationGetContext, [128](#)
  - rtAccelerationGetData, [128](#)
  - rtAccelerationGetDataSize, [130](#)
  - rtAccelerationGetProperty, [130](#)
  - rtAccelerationGetTraverser, [131](#)
  - rtAccelerationIsDirty, [131](#)
  - rtAccelerationMarkDirty, [132](#)
  - rtAccelerationSetBuilder, [132](#)
  - rtAccelerationSetData, [133](#)
  - rtAccelerationSetProperty, [134](#)
  - rtAccelerationSetTraverser, [135](#)
  - rtAccelerationValidate, [136](#)
  - rtBufferBindProgressiveStream, [136](#)
  - rtBufferCreate, [137](#)
  - rtBufferDestroy, [138](#)
  - rtBufferGetAttribute, [138](#)
  - rtBufferGetContext, [139](#)
  - rtBufferGetDimensionality, [139](#)
  - rtBufferGetElementSize, [140](#)
  - rtBufferGetFormat, [140](#)
  - rtBufferGetId, [141](#)
  - rtBufferGetMipLevelCount, [141](#)
  - rtBufferGetMipLevelSize1D, [142](#)
  - rtBufferGetMipLevelSize2D, [142](#)
  - rtBufferGetMipLevelSize3D, [143](#)
  - rtBufferGetProgressiveUpdateReady, [143](#)
  - rtBufferGetSize1D, [144](#)
  - rtBufferGetSize2D, [144](#)
  - rtBufferGetSize3D, [145](#)
  - rtBufferGetSizev, [145](#)
  - rtBufferMap, [146](#)
  - rtBufferMapEx, [146](#)
  - rtBufferSetAttribute, [147](#)
  - rtBufferSetElementSize, [148](#)
  - rtBufferSetFormat, [148](#)
  - rtBufferSetMipLevelCount, [150](#)
  - rtBufferSetSize1D, [150](#)
  - rtBufferSetSize2D, [151](#)
  - rtBufferSetSize3D, [151](#)
  - rtBufferSetSizev, [152](#)
  - rtBufferUnmap, [153](#)
  - rtBufferUnmapEx, [153](#)
  - rtBufferValidate, [154](#)
  - rtContextCompile, [154](#)
  - rtContextCreate, [155](#)
  - rtContextDeclareVariable, [155](#)
  - rtContextDestroy, [156](#)
  - rtContextGetAttribute, [156](#)
  - rtContextGetBufferFromId, [157](#)
  - rtContextGetDeviceCount, [158](#)
  - rtContextGetDevices, [158](#)
  - rtContextGetEntryPointCount, [159](#)

rtContextGetErrorString, 159  
rtContextGetExceptionEnabled, 159  
rtContextGetExceptionProgram, 160  
rtContextGetMissProgram, 160  
rtContextGetPrintBufferSize, 161  
rtContextGetPrintEnabled, 161  
rtContextGetPrintLaunchIndex, 162  
rtContextGetProgramFromId, 162  
rtContextGetRayGenerationProgram, 163  
rtContextGetRayTypeCount, 163  
rtContextGetRunningState, 163  
rtContextGetStackSize, 164  
rtContextGetTextureSamplerFromId, 164  
rtContextGetVariable, 165  
rtContextGetVariableCount, 165  
rtContextLaunch1D, 165  
rtContextLaunch2D, 166  
rtContextLaunch3D, 166  
rtContextLaunchProgressive2D, 166  
rtContextQueryVariable, 167  
rtContextRemoveVariable, 168  
rtContextSetAttribute, 168  
rtContextSetDevices, 169  
rtContextSetEntryPointCount, 169  
rtContextSetExceptionEnabled, 170  
rtContextSetExceptionProgram, 171  
rtContextSetMissProgram, 171  
rtContextSetPrintBufferSize, 172  
rtContextSetPrintEnabled, 172  
rtContextSetPrintLaunchIndex, 173  
rtContextSetRayGenerationProgram, 173  
rtContextSetRayTypeCount, 174  
rtContextSetRemoteDevice, 174  
rtContextSetStackSize, 175  
rtContextSetTimeoutCallback, 175  
rtContextStopProgressive, 176  
rtContextValidate, 176  
rtDeviceGetAttribute, 177  
rtDeviceGetDeviceCount, 178  
rtGeometryCreate, 178  
rtGeometryDeclareVariable, 179  
rtGeometryDestroy, 179  
rtGeometryGetBoundingBoxProgram, 180  
rtGeometryGetContext, 180  
rtGeometryGetIntersectionProgram, 181  
rtGeometryGetPrimitiveCount, 181  
rtGeometryGetPrimitiveIndexOffset, 181  
rtGeometryGetVariable, 182  
rtGeometryGetVariableCount, 182  
rtGeometryGroupCreate, 183  
rtGeometryGroupDestroy, 183  
rtGeometryGroupGetAcceleration, 184  
rtGeometryGroupGetChild, 184  
rtGeometryGroupGetChildCount, 185  
rtGeometryGroupGetContext, 185  
rtGeometryGroupSetAcceleration, 186  
rtGeometryGroupSetChild, 186  
rtGeometryGroupSetChildCount, 188  
rtGeometryGroupValidate, 188  
rtGeometryInstanceCreate, 189  
rtGeometryInstanceDeclareVariable, 189  
rtGeometryInstanceDestroy, 191  
rtGeometryInstanceGetContext, 191  
rtGeometryInstanceGetGeometry, 192  
rtGeometryInstanceGetMaterial, 192  
rtGeometryInstanceGetMaterialCount, 193  
rtGeometryInstanceGetVariable, 193  
rtGeometryInstanceGetVariableCount, 194  
rtGeometryInstanceQueryVariable, 194  
rtGeometryInstanceRemoveVariable, 195  
rtGeometryInstanceSetGeometry, 196  
rtGeometryInstanceSetMaterial, 196  
rtGeometryInstanceSetMaterialCount, 197  
rtGeometryInstanceValidate, 197  
rtGeometryIsDirty, 198  
rtGeometryMarkDirty, 198  
rtGeometryQueryVariable, 198  
rtGeometryRemoveVariable, 199  
rtGeometrySetBoundingBoxProgram, 200  
rtGeometrySetIntersectionProgram, 200  
rtGeometrySetPrimitiveCount, 201  
rtGeometrySetPrimitiveIndexOffset, 201  
rtGeometryValidate, 201  
rtGetVersion, 202  
rtGroupCreate, 202  
rtGroupDestroy, 203  
rtGroupGetAcceleration, 203  
rtGroupGetChild, 204  
rtGroupGetChildCount, 204  
rtGroupGetChildType, 204  
rtGroupGetContext, 205  
rtGroupSetAcceleration, 205  
rtGroupSetChild, 206  
rtGroupSetChildCount, 206  
rtGroupValidate, 207  
rtMaterialCreate, 207  
rtMaterialDeclareVariable, 208  
rtMaterialDestroy, 208  
rtMaterialGetAnyHitProgram, 209  
rtMaterialGetClosestHitProgram, 209  
rtMaterialGetContext, 210  
rtMaterialGetVariable, 210  
rtMaterialGetVariableCount, 210  
rtMaterialQueryVariable, 211  
rtMaterialRemoveVariable, 211  
rtMaterialSetAnyHitProgram, 212  
rtMaterialSetClosestHitProgram, 213  
rtMaterialValidate, 213  
rtProgramCreateFromPTXFile, 214  
rtProgramCreateFromPTXString, 214  
rtProgramDeclareVariable, 215  
rtProgramDestroy, 215  
rtProgramGetContext, 216  
rtProgramGetId, 216  
rtProgramGetVariable, 217  
rtProgramGetVariableCount, 217



rtProgramQueryVariable, 218  
rtProgramRemoveVariable, 218  
rtProgramValidate, 219  
rtRemoteDeviceCreate, 219  
rtRemoteDeviceDestroy, 221  
rtRemoteDeviceGetAttribute, 221  
rtRemoteDeviceRelease, 223  
rtRemoteDeviceReserve, 223  
rtSelectorCreate, 224  
rtSelectorDeclareVariable, 225  
rtSelectorDestroy, 225  
rtSelectorGetChild, 226  
rtSelectorGetChildCount, 226  
rtSelectorGetChildType, 227  
rtSelectorGetContext, 227  
rtSelectorGetVariable, 228  
rtSelectorGetVariableCount, 228  
rtSelectorGetVisitProgram, 229  
rtSelectorQueryVariable, 229  
rtSelectorRemoveVariable, 229  
rtSelectorSetChild, 230  
rtSelectorSetChildCount, 230  
rtSelectorSetVisitProgram, 231  
rtSelectorValidate, 231  
rtTextureSamplerCreate, 232  
rtTextureSamplerDestroy, 232  
rtTextureSamplerGetArraySize, 233  
rtTextureSamplerGetBuffer, 233  
rtTextureSamplerGetContext, 234  
rtTextureSamplerGetFilteringModes, 234  
rtTextureSamplerGetId, 235  
rtTextureSamplerGetIndexingMode, 235  
rtTextureSamplerGetMaxAnisotropy, 236  
rtTextureSamplerGetMipLevelBias, 236  
rtTextureSamplerGetMipLevelClamp, 237  
rtTextureSamplerGetMipLevelCount, 237  
rtTextureSamplerGetReadMode, 237  
rtTextureSamplerGetWrapMode, 238  
rtTextureSamplerSetArraySize, 238  
rtTextureSamplerSetBuffer, 239  
rtTextureSamplerSetFilteringModes, 239  
rtTextureSamplerSetIndexingMode, 240  
rtTextureSamplerSetMaxAnisotropy, 241  
rtTextureSamplerSetMipLevelBias, 241  
rtTextureSamplerSetMipLevelClamp, 241  
rtTextureSamplerSetMipLevelCount, 242  
rtTextureSamplerSetReadMode, 242  
rtTextureSamplerSetWrapMode, 243  
rtTextureSamplerValidate, 244  
rtTransformCreate, 244  
rtTransformDestroy, 245  
rtTransformGetChild, 245  
rtTransformGetChildType, 246  
rtTransformGetContext, 246  
rtTransformGetMatrix, 247  
rtTransformSetChild, 247  
rtTransformSetMatrix, 248  
rtTransformValidate, 249  
rtVariableGet1f, 249  
rtVariableGet1fv, 250  
rtVariableGet1i, 250  
rtVariableGet1iv, 250  
rtVariableGet1ui, 250  
rtVariableGet1uiv, 250  
rtVariableGet2f, 250  
rtVariableGet2fv, 251  
rtVariableGet2i, 251  
rtVariableGet2iv, 251  
rtVariableGet2ui, 251  
rtVariableGet2uiv, 251  
rtVariableGet3f, 251  
rtVariableGet3fv, 251  
rtVariableGet3i, 252  
rtVariableGet3iv, 252  
rtVariableGet3ui, 252  
rtVariableGet3uiv, 252  
rtVariableGet4f, 252  
rtVariableGet4fv, 252  
rtVariableGet4i, 252  
rtVariableGet4iv, 253  
rtVariableGet4ui, 253  
rtVariableGet4uiv, 253  
rtVariableGetAnnotation, 253  
rtVariableGetContext, 254  
rtVariableGetMatrix2x2fv, 254  
rtVariableGetMatrix2x3fv, 254  
rtVariableGetMatrix2x4fv, 254  
rtVariableGetMatrix3x2fv, 254  
rtVariableGetMatrix3x3fv, 255  
rtVariableGetMatrix3x4fv, 255  
rtVariableGetMatrix4x2fv, 255  
rtVariableGetMatrix4x3fv, 255  
rtVariableGetMatrix4x4fv, 255  
rtVariableGetName, 255  
rtVariableGetObject, 256  
rtVariableGetSize, 256  
rtVariableGetType, 257  
rtVariableGetUserData, 258  
rtVariableSet1f, 259  
rtVariableSet1fv, 259  
rtVariableSet1i, 260  
rtVariableSet1iv, 260  
rtVariableSet1ui, 260  
rtVariableSet1uiv, 260  
rtVariableSet2f, 260  
rtVariableSet2fv, 260  
rtVariableSet2i, 260  
rtVariableSet2iv, 260  
rtVariableSet2ui, 261  
rtVariableSet2uiv, 261  
rtVariableSet3f, 261  
rtVariableSet3fv, 261  
rtVariableSet3i, 261  
rtVariableSet3iv, 261  
rtVariableSet3ui, 261  
rtVariableSet3uiv, 262

- rtVariableSet4f, [262](#)
- rtVariableSet4fv, [262](#)
- rtVariableSet4i, [262](#)
- rtVariableSet4iv, [262](#)
- rtVariableSet4ui, [262](#)
- rtVariableSet4uiv, [263](#)
- rtVariableSetMatrix2x2fv, [263](#)
- rtVariableSetMatrix2x3fv, [263](#)
- rtVariableSetMatrix2x4fv, [263](#)
- rtVariableSetMatrix3x2fv, [263](#)
- rtVariableSetMatrix3x3fv, [263](#)
- rtVariableSetMatrix3x4fv, [263](#)
- rtVariableSetMatrix4x2fv, [264](#)
- rtVariableSetMatrix4x3fv, [264](#)
- rtVariableSetMatrix4x4fv, [264](#)
- rtVariableSetObject, [264](#)
- rtVariableSetUserData, [264](#)
- optix\_prime.h, [265](#)
  - RTPbufferdesc, [267](#)
  - RTPcontext, [267](#)
  - RTPmodel, [267](#)
  - RTPquery, [267](#)
  - rtpBufferDescCreate, [267](#)
  - rtpBufferDescDestroy, [267](#)
  - rtpBufferDescGetContext, [268](#)
  - rtpBufferDescSetCudaDeviceNumber, [268](#)
  - rtpBufferDescSetRange, [268](#)
  - rtpBufferDescSetStride, [269](#)
  - rtpContextCreate, [269](#)
  - rtpContextDestroy, [270](#)
  - rtpContextGetLastErrorString, [270](#)
  - rtpContextSetCpuThreads, [270](#)
  - rtpContextSetCudaDeviceNumbers, [271](#)
  - rtpGetErrorString, [271](#)
  - rtpGetVersion, [271](#)
  - rtpGetVersionString, [272](#)
  - rtpHostBufferLock, [272](#)
  - rtpHostBufferUnlock, [272](#)
  - rtpModelCopy, [273](#)
  - rtpModelCreate, [273](#)
  - rtpModelDestroy, [273](#)
  - rtpModelFinish, [274](#)
  - rtpModelGetContext, [274](#)
  - rtpModelGetFinished, [274](#)
  - rtpModelSetBuilderParameter, [274](#)
  - rtpModelSetInstances, [275](#)
  - rtpModelSetTriangles, [276](#)
  - rtpModelUpdate, [276](#)
  - rtpQueryCreate, [277](#)
  - rtpQueryDestroy, [277](#)
  - rtpQueryExecute, [277](#)
  - rtpQueryFinish, [278](#)
  - rtpQueryGetContext, [278](#)
  - rtpQueryGetFinished, [279](#)
  - rtpQuerySetCudaStream, [279](#)
  - rtpQuerySetHits, [279](#)
  - rtpQuerySetRays, [279](#)
- optix\_prime\_declarations.h, [281](#)
- RTPbufferformat, [282](#)
- RTPbuffertype, [283](#)
- RTPbuilderparam, [283](#)
- RTPcontexttype, [283](#)
- RTPmodelhint, [283](#)
- RTPqueryhint, [283](#)
- RTPquerytype, [283](#)
- RTPresult, [284](#)
- optix\_primepp.h, [284](#)
- optix\_world.h, [285](#)
- optixpp\_namespace.h, [285](#)
- optixu.h, [287](#)
  - RTU\_CHECK\_ERROR, [289](#)
  - rtuCUDACompileFile, [290](#)
  - rtuCUDACompileString, [291](#)
  - rtuCUDAGetCompileResult, [291](#)
  - rtuCreateClusteredMesh, [289](#)
  - rtuCreateClusteredMeshExt, [290](#)
  - rtuGeometryGroupAddChild, [291](#)
  - rtuGeometryGroupGetChildIndex, [291](#)
  - rtuGeometryGroupRemoveChild, [291](#)
  - rtuGeometryGroupRemoveChildByIndex, [292](#)
  - rtuGetSizeForRTformat, [292](#)
  - rtuGroupAddChild, [292](#)
  - rtuGroupGetChildIndex, [292](#)
  - rtuGroupRemoveChild, [292](#)
  - rtuGroupRemoveChildByIndex, [292](#)
  - rtuNameForType, [292](#)
  - rtuSelectorAddChild, [292](#)
  - rtuSelectorGetChildIndex, [292](#)
  - rtuSelectorRemoveChild, [293](#)
  - rtuSelectorRemoveChildByIndex, [293](#)
  - rtuTransformGetChild, [293](#)
  - rtuTransformGetChildType, [293](#)
  - rtuTransformSetChild, [293](#)
- optixu\_traversal.h
  - RTU\_INITOPTION\_CPU\_ONLY, [305](#)
  - RTU\_INITOPTION\_CULL\_BACKFACE, [305](#)
  - RTU\_INITOPTION\_GPU\_ONLY, [305](#)
  - RTU\_INITOPTION\_NONE, [305](#)
  - RTU\_OPTION\_INT\_NUM\_THREADS, [305](#)
  - RTU\_OUTPUT\_BACKFACING, [306](#)
  - RTU\_OUTPUT\_BARYCENTRIC, [306](#)
  - RTU\_OUTPUT\_NONE, [306](#)
  - RTU\_OUTPUT\_NORMAL, [306](#)
  - RTU\_QUERY\_TYPE\_ANY\_HIT, [306](#)
  - RTU\_QUERY\_TYPE\_CLOSEST\_HIT, [306](#)
  - RTU\_QUERY\_TYPE\_COUNT, [306](#)
  - RTU\_RAYFORMAT\_COUNT, [306](#)
  - RTU\_RAYFORMAT\_ORIGIN\_DIRECTION\_INTERLEAVED, [306](#)
  - RTU\_RAYFORMAT\_ORIGIN\_DIRECTION\_TMIN\_TMAX\_INTERLEAVED, [306](#)
  - RTU\_TRIFORMAT\_COUNT, [306](#)
  - RTU\_TRIFORMAT\_MESH, [306](#)
  - RTU\_TRIFORMAT\_TRIANGLE\_SOUP, [306](#)
- optixu\_aabb\_namespace.h, [293](#)
- optixu\_math\_namespace.h, [293](#)



- optixu\_math\_stream\_namespace.h, 302
- optixu\_matrix\_namespace.h, 302
- optixu\_traversal.h, 304
  - RTUinitoptions, 305
  - RTUoption, 305
  - RTUoutput, 305
  - RTUquerytype, 306
  - RTUrayformat, 306
  - RTUtraversal, 305
  - RTUtriformat, 306
  - rtuTraversalCreate, 306
  - rtuTraversalDestroy, 307
  - rtuTraversalGetAccelData, 307
  - rtuTraversalGetAccelDataSize, 307
  - rtuTraversalGetErrorString, 307
  - rtuTraversalMapOutput, 307
  - rtuTraversalMapRays, 308
  - rtuTraversalMapResults, 308
  - rtuTraversalPreprocess, 308
  - rtuTraversalSetAccelData, 308
  - rtuTraversalSetMesh, 308
  - rtuTraversalSetOption, 309
  - rtuTraversalSetTriangles, 309
  - rtuTraversalTraverse, 309
  - rtuTraversalUnmapOutput, 309
  - rtuTraversalUnmapRays, 309
  - rtuTraversalUnmapResults, 309
- origin
  - Ray, 40
- prim\_id
  - RTUtraversalresult, 43
- queryVariable
  - optix::ContextObj, 21
  - optix::GeometryInstanceObj, 26
  - optix::GeometryObj, 28
  - optix::MaterialObj, 32
  - optix::ProgramObj, 38
  - optix::ScopedObj, 44
- RT\_BUFFER\_ATTRIBUTE\_STREAM\_BITRATE
  - optix\_declarations.h, 82
- RT\_BUFFER\_ATTRIBUTE\_STREAM\_FORMAT
  - optix\_declarations.h, 82
- RT\_BUFFER\_ATTRIBUTE\_STREAM\_FPS
  - optix\_declarations.h, 82
- RT\_BUFFER\_ATTRIBUTE\_STREAM\_GAMMA
  - optix\_declarations.h, 82
- RT\_BUFFER\_COPY\_ON\_DIRTY
  - optix\_declarations.h, 82
- RT\_BUFFER\_CUBEMAP
  - optix\_declarations.h, 82
- RT\_BUFFER\_GPU\_LOCAL
  - optix\_declarations.h, 82
- RT\_BUFFER\_ID\_NULL
  - optix\_declarations.h, 82
- RT\_BUFFER\_INPUT
  - optix\_declarations.h, 83
- RT\_BUFFER\_INPUT\_OUTPUT
  - optix\_declarations.h, 83
- RT\_BUFFER\_LAYERED
  - optix\_declarations.h, 82
- RT\_BUFFER\_MAP\_READ
  - optix\_declarations.h, 83
- RT\_BUFFER\_MAP\_READ\_WRITE
  - optix\_declarations.h, 83
- RT\_BUFFER\_MAP\_WRITE
  - optix\_declarations.h, 83
- RT\_BUFFER\_MAP\_WRITE\_DISCARD
  - optix\_declarations.h, 83
- RT\_BUFFER\_OUTPUT
  - optix\_declarations.h, 83
- RT\_BUFFER\_PROGRESSIVE\_STREAM
  - optix\_declarations.h, 83
- RT\_CONTEXT\_ATTRIBUTE\_AVAILABLE\_DEVICE\_M-  
EMORY
  - optix\_declarations.h, 83
- RT\_CONTEXT\_ATTRIBUTE\_CPU\_NUM\_THREADS
  - optix\_declarations.h, 83
- RT\_CONTEXT\_ATTRIBUTE\_GPU\_PAGING\_ACTIVE
  - optix\_declarations.h, 83
- RT\_CONTEXT\_ATTRIBUTE\_GPU\_PAGING\_FORCE-  
D\_OFF
  - optix\_declarations.h, 83
- RT\_CONTEXT\_ATTRIBUTE\_MAX\_TEXTURE\_COUN-  
T
  - optix\_declarations.h, 83
- RT\_CONTEXT\_ATTRIBUTE\_USED\_HOST\_MEMORY
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_CLOCK\_RATE
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_COMPUTE\_CAPABILITY
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_CUDA\_DEVICE\_ORDINAL
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_EXECUTION\_TIMEOUT\_E-  
NABLED
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_MAX\_HARDWARE\_TEXT-  
URE\_COUNT
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_MAX\_THREADS\_PER\_BL-  
OCK
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_MULTIPROCESSOR\_CO-  
UNT
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_NAME
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_TCC\_DRIVER
  - optix\_declarations.h, 83
- RT\_DEVICE\_ATTRIBUTE\_TOTAL\_MEMORY
  - optix\_declarations.h, 83
- RT\_ERROR\_ALREADY\_MAPPED
  - optix\_declarations.h, 87
- RT\_ERROR\_AUTHENTICATION\_FAILED

- optix\_declarations.h, [87](#)
- RT\_ERROR\_CLUSTER\_ALREADY\_RUNNING
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_CLUSTER\_NOT\_RUNNING
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_CONNECTION\_ALREADY\_EXISTS
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_CONNECTION\_FAILED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_CONTEXT\_CREATION\_FAILED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_FILE\_NOT\_FOUND
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_ILLEGAL\_SYMBOL
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_INSUFFICIENT\_FREE\_NODES
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_INVALID\_CONTEXT
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_INVALID\_DEVICE
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_INVALID\_DRIVER\_VERSION
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_INVALID\_IMAGE
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_INVALID\_SOURCE
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_INVALID\_VALUE
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_LAUNCH\_FAILED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_MEMORY\_ALLOCATION\_FAILED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_NETWORK\_INIT\_FAILED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_NETWORK\_LOAD\_FAILED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_NO\_DEVICE
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_NOT\_SUPPORTED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_OBJECT\_CREATION\_FAILED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_RESOURCE\_ALREADY\_REGISTERED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_RESOURCE\_NOT\_REGISTERED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_TYPE\_MISMATCH
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_UNKNOWN
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_VARIABLE\_NOT\_FOUND
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_VARIABLE\_REDECLARED
  - optix\_declarations.h, [87](#)
- RT\_ERROR\_VERSION\_MISMATCH
  - optix\_declarations.h, [87](#)
- RT\_EXCEPTION\_ALL
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_BUFFER\_ID\_INVALID
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_BUFFER\_INDEX\_OUT\_OF\_BOUNDS
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_INDEX\_OUT\_OF\_BOUNDS
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_INTERNAL\_ERROR
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_INVALID\_RAY
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_PROGRAM\_ID\_INVALID
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_STACK\_OVERFLOW
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_TEXTURE\_ID\_INVALID
  - optix\_declarations.h, [84](#)
- RT\_EXCEPTION\_USER
  - optix\_declarations.h, [84](#)
- RT\_FILTER\_LINEAR
  - optix\_declarations.h, [84](#)
- RT\_FILTER\_NEAREST
  - optix\_declarations.h, [84](#)
- RT\_FILTER\_NONE
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_BUFFER\_ID
  - optix\_declarations.h, [85](#)
- RT\_FORMAT\_BYTE
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_BYTE2
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_BYTE3
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_BYTE4
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_FLOAT
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_FLOAT2
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_FLOAT3
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_FLOAT4
  - optix\_declarations.h, [84](#)
- RT\_FORMAT\_HALF
  - optix\_declarations.h, [85](#)
- RT\_FORMAT\_HALF2
  - optix\_declarations.h, [85](#)
- RT\_FORMAT\_HALF3
  - optix\_declarations.h, [85](#)
- RT\_FORMAT\_HALF4
  - optix\_declarations.h, [85](#)
- RT\_FORMAT\_INT
  - optix\_declarations.h, [85](#)
- RT\_FORMAT\_INT2
  - optix\_declarations.h, [85](#)
- RT\_FORMAT\_INT3
  - optix\_declarations.h, [85](#)

- RT\_FORMAT\_INT4
  - [optix\\_declarations.h, 85](#)
- RT\_FORMAT\_PROGRAM\_ID
  - [optix\\_declarations.h, 85](#)
- RT\_FORMAT\_SHORT
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_SHORT2
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_SHORT3
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_SHORT4
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_UNKNOWN
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_UNSIGNED\_BYTE
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_UNSIGNED\_BYTE2
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_UNSIGNED\_BYTE3
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_UNSIGNED\_BYTE4
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_UNSIGNED\_INT
  - [optix\\_declarations.h, 85](#)
- RT\_FORMAT\_UNSIGNED\_INT2
  - [optix\\_declarations.h, 85](#)
- RT\_FORMAT\_UNSIGNED\_INT3
  - [optix\\_declarations.h, 85](#)
- RT\_FORMAT\_UNSIGNED\_INT4
  - [optix\\_declarations.h, 85](#)
- RT\_FORMAT\_UNSIGNED\_SHORT
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_UNSIGNED\_SHORT2
  - [optix\\_declarations.h, 84](#)
- RT\_FORMAT\_UNSIGNED\_SHORT3
  - [optix\\_declarations.h, 85](#)
- RT\_FORMAT\_UNSIGNED\_SHORT4
  - [optix\\_declarations.h, 85](#)
- RT\_FORMAT\_USER
  - [optix\\_declarations.h, 85](#)
- RT\_INTERNAL\_INVERSE\_TRANSPOSE
  - [optix\\_defines.h, 89](#)
- RT\_OBJECT\_TO\_WORLD
  - [optix\\_defines.h, 90](#)
- RT\_OBJECTTYPE\_BUFFER
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_FLOAT
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_FLOAT2
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_FLOAT3
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_FLOAT4
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_GEOMETRY\_GROUP
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_GEOMETRY\_INSTANCE
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_GROUP
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_INT
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_INT2
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_INT3
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_INT4
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT2x2
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT2x3
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT2x4
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT3x2
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT3x3
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT3x4
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT4x2
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT4x3
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_MATRIX\_FLOAT4x4
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_OBJECT
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_PROGRAM
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_SELECTOR
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_TEXTURE\_SAMPLER
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_TRANSFORM
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_UNKNOWN
  - [optix\\_declarations.h, 85](#)
- RT\_OBJECTTYPE\_UNSIGNED\_INT
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_UNSIGNED\_INT2
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_UNSIGNED\_INT3
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_UNSIGNED\_INT4
  - [optix\\_declarations.h, 86](#)
- RT\_OBJECTTYPE\_USER
  - [optix\\_declarations.h, 86](#)
- RT\_PROGRAM\_ID\_NULL
  - [optix\\_declarations.h, 86](#)
- RT\_REMOTEDEVICE\_ATTRIBUTE\_CLUSTER\_URL
  - [optix\\_declarations.h, 86](#)
- RT\_REMOTEDEVICE\_ATTRIBUTE\_CONFIGURATION
  - [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_GPU\_TOTAL\_MEMORY  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_HEAD\_NODE\_URL  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_NAME  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_CONFIGURATIONS  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_FREE\_NODES  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_GPUS  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_RESERVED\_NODES  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_NUM\_TOTAL\_NODES  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_ATTRIBUTE\_STATUS  
    [optix\\_declarations.h, 86](#)

RT\_REMOTEDEVICE\_STATUS\_CONNECTED  
    [optix\\_declarations.h, 87](#)

RT\_REMOTEDEVICE\_STATUS\_DISCONNECTED  
    [optix\\_declarations.h, 87](#)

RT\_REMOTEDEVICE\_STATUS\_READY  
    [optix\\_declarations.h, 87](#)

RT\_REMOTEDEVICE\_STATUS\_RESERVED  
    [optix\\_declarations.h, 87](#)

RT\_SUCCESS  
    [optix\\_declarations.h, 87](#)

RT\_TARGET\_GL\_RENDER\_BUFFER  
    [optix\\_declarations.h, 85](#)

RT\_TARGET\_GL\_TEXTURE\_1D  
    [optix\\_declarations.h, 85](#)

RT\_TARGET\_GL\_TEXTURE\_1D\_ARRAY  
    [optix\\_declarations.h, 85](#)

RT\_TARGET\_GL\_TEXTURE\_2D  
    [optix\\_declarations.h, 85](#)

RT\_TARGET\_GL\_TEXTURE\_2D\_ARRAY  
    [optix\\_declarations.h, 85](#)

RT\_TARGET\_GL\_TEXTURE\_3D  
    [optix\\_declarations.h, 85](#)

RT\_TARGET\_GL\_TEXTURE\_CUBE\_MAP  
    [optix\\_declarations.h, 85](#)

RT\_TARGET\_GL\_TEXTURE\_CUBE\_MAP\_ARRAY  
    [optix\\_declarations.h, 85](#)

RT\_TARGET\_GL\_TEXTURE\_RECTANGLE  
    [optix\\_declarations.h, 85](#)

RT\_TEXTURE\_ID\_NULL  
    [optix\\_declarations.h, 88](#)

RT\_TEXTURE\_INDEX\_ARRAY\_INDEX  
    [optix\\_declarations.h, 88](#)

RT\_TEXTURE\_INDEX\_NORMALIZED\_COORDINATES  
    [optix\\_declarations.h, 88](#)

RT\_TEXTURE\_READ\_ELEMENT\_TYPE  
    [optix\\_declarations.h, 88](#)

RT\_TEXTURE\_READ\_ELEMENT\_TYPE\_SRGB  
    [optix\\_declarations.h, 88](#)

RT\_TEXTURE\_READ\_NORMALIZED\_FLOAT  
    [optix\\_declarations.h, 88](#)

RT\_TEXTURE\_READ\_NORMALIZED\_FLOAT\_SRGB  
    [optix\\_declarations.h, 88](#)

RT\_TIMEOUT\_CALLBACK  
    [optix\\_declarations.h, 87](#)

RT\_WORLD\_TO\_OBJECT  
    [optix\\_defines.h, 90](#)

RT\_WRAP\_CLAMP\_TO\_BORDER  
    [optix\\_declarations.h, 88](#)

RT\_WRAP\_CLAMP\_TO\_EDGE  
    [optix\\_declarations.h, 88](#)

RT\_WRAP\_MIRROR  
    [optix\\_declarations.h, 88](#)

RT\_WRAP\_REPEAT  
    [optix\\_declarations.h, 88](#)

RTP\_BUFFER\_FORMAT\_HIT\_BITMASK  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_HIT\_T  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_INSTID  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_INSTID\_U\_V  
    [optix\\_prime\\_declarations.h, 283](#)

RTP\_BUFFER\_FORMAT\_HIT\_T\_TRIID\_U\_V  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_INDICES\_INT3  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_INDICES\_INT3\_MASK\_INT  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_INSTANCE\_MODEL  
    [optix\\_prime\\_declarations.h, 283](#)

RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_DIRECTION  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_MASK\_DIRECTION\_TMAX  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_RAY\_ORIGIN\_TMIN\_DIRECTION\_TMAX  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_TRANSFORM\_FLOAT4x3  
    [optix\\_prime\\_declarations.h, 283](#)

RTP\_BUFFER\_FORMAT\_TRANSFORM\_FLOAT4x4  
    [optix\\_prime\\_declarations.h, 283](#)

RTP\_BUFFER\_FORMAT\_VERTEX\_FLOAT3  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_FORMAT\_VERTEX\_FLOAT4  
    [optix\\_prime\\_declarations.h, 282](#)

RTP\_BUFFER\_TYPE\_CUDA\_LINEAR  
    [optix\\_prime\\_declarations.h, 283](#)

RTP\_BUFFER\_TYPE\_HOST

- optix\_prime\_declarations.h, [283](#)
- RTP\_BUILDER\_PARAM\_CHUNK\_SIZE
  - optix\_prime\_declarations.h, [283](#)
- RTP\_BUILDER\_PARAM\_USE\_CALLER\_TRIANGLES
  - optix\_prime\_declarations.h, [283](#)
- RTP\_CONTEXT\_TYPE\_CPU
  - optix\_prime\_declarations.h, [283](#)
- RTP\_CONTEXT\_TYPE\_CUDA
  - optix\_prime\_declarations.h, [283](#)
- RTP\_ERROR\_INVALID\_CONTEXT
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_INVALID\_HANDLE
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_INVALID\_OPERATION
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_INVALID\_VALUE
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_MEMORY\_ALLOCATION\_FAILED
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_NOT\_SUPPORTED
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_OBJECT\_CREATION\_FAILED
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_OUT\_OF\_MEMORY
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_UNKNOWN
  - optix\_prime\_declarations.h, [284](#)
- RTP\_ERROR\_VALIDATION\_ERROR
  - optix\_prime\_declarations.h, [284](#)
- RTP\_MODEL\_HINT\_ASYNC
  - optix\_prime\_declarations.h, [283](#)
- RTP\_MODEL\_HINT\_MASK\_UPDATE
  - optix\_prime\_declarations.h, [283](#)
- RTP\_MODEL\_HINT\_NONE
  - optix\_prime\_declarations.h, [283](#)
- RTP\_MODEL\_HINT\_USER\_TRIANGLES\_AFTER\_COPY\_SET
  - optix\_prime\_declarations.h, [283](#)
- RTP\_QUERY\_HINT\_ASYNC
  - optix\_prime\_declarations.h, [283](#)
- RTP\_QUERY\_HINT\_NONE
  - optix\_prime\_declarations.h, [283](#)
- RTP\_QUERY\_HINT\_WATERTIGHT
  - optix\_prime\_declarations.h, [283](#)
- RTP\_QUERY\_TYPE\_ANY
  - optix\_prime\_declarations.h, [284](#)
- RTP\_QUERY\_TYPE\_CLOSEST
  - optix\_prime\_declarations.h, [284](#)
- RTP\_SUCCESS
  - optix\_prime\_declarations.h, [284](#)
- RTU\_INITOPTION\_CPU\_ONLY
  - optixu\_traversal.h, [305](#)
- RTU\_INITOPTION\_CULL\_BACKFACE
  - optixu\_traversal.h, [305](#)
- RTU\_INITOPTION\_GPU\_ONLY
  - optixu\_traversal.h, [305](#)
- RTU\_INITOPTION\_NONE
  - optixu\_traversal.h, [305](#)
- RTU\_OPTION\_INT\_NUM\_THREADS
  - optixu\_traversal.h, [305](#)
- RTU\_OUTPUT\_BACKFACING
  - optixu\_traversal.h, [306](#)
- RTU\_OUTPUT\_BARYCENTRIC
  - optixu\_traversal.h, [306](#)
- RTU\_OUTPUT\_NONE
  - optixu\_traversal.h, [306](#)
- RTU\_OUTPUT\_NORMAL
  - optixu\_traversal.h, [306](#)
- RTU\_QUERY\_TYPE\_ANY\_HIT
  - optixu\_traversal.h, [306](#)
- RTU\_QUERY\_TYPE\_CLOSEST\_HIT
  - optixu\_traversal.h, [306](#)
- RTU\_QUERY\_TYPE\_COUNT
  - optixu\_traversal.h, [306](#)
- RTU\_RAYFORMAT\_COUNT
  - optixu\_traversal.h, [306](#)
- RTU\_RAYFORMAT\_ORIGIN\_DIRECTION\_INTERLEAVED
  - optixu\_traversal.h, [306](#)
- RTU\_RAYFORMAT\_ORIGIN\_DIRECTION\_TMIN\_TMAX\_INTERLEAVED
  - optixu\_traversal.h, [306](#)
- RTU\_TRIFORMAT\_COUNT
  - optixu\_traversal.h, [306](#)
- RTU\_TRIFORMAT\_MESH
  - optixu\_traversal.h, [306](#)
- RTU\_TRIFORMAT\_TRIANGLE\_SOUP
  - optixu\_traversal.h, [306](#)
- RT\_DEFAULT\_MAX
  - optix\_datatypes.h, [77](#)
- RT\_PROGRAM
  - optix\_device.h, [97](#)
- RTPbufferdesc
  - optix\_prime.h, [267](#)
- RTPbufferformat
  - optix\_prime\_declarations.h, [282](#)
- RTPbuffertype
  - optix\_prime\_declarations.h, [283](#)
- RTPbuilderparam
  - optix\_prime\_declarations.h, [283](#)
- RTPcontext
  - optix\_prime.h, [267](#)
- RTPcontexttype
  - optix\_prime\_declarations.h, [283](#)
- RTPinternals\_3070, [43](#)
- RTPinternals\_3070::BvhNode, [14](#)
- RTPinternals\_3070::WoopTriangle, [55](#)
- RTPmodel
  - optix\_prime.h, [267](#)
- RTPmodelhint
  - optix\_prime\_declarations.h, [283](#)
- RTPquery
  - optix\_prime.h, [267](#)
- RTPqueryhint
  - optix\_prime\_declarations.h, [283](#)
- RTPquerytype

- optix\_prime\_declarations.h, 283
- RTPresult
  - optix\_prime\_declarations.h, 284
- RTU\_CHECK\_ERROR
  - optixu.h, 289
- RTU\_GROUP\_ADD\_CHILD
  - optixu.h, 289
- RTUinitoptions
  - optixu\_traversal.h, 305
- RTUoption
  - optixu\_traversal.h, 305
- RTUoutput
  - optixu\_traversal.h, 305
- RTUquerytype
  - optixu\_traversal.h, 306
- RTUrayformat
  - optixu\_traversal.h, 306
- RTUtraversal
  - optixu\_traversal.h, 305
- RTUtraversalresult, 43
  - prim\_id, 43
  - t, 43
- RTUtriformat
  - optixu\_traversal.h, 306
- RTacceleration
  - optix\_host.h, 125
- RTbuffer
  - optix\_host.h, 125
- RTbufferattribute
  - optix\_declarations.h, 82
- RTbufferflag
  - optix\_declarations.h, 82
- RTbufferidnull
  - optix\_declarations.h, 82
- RTbuffermapflag
  - optix\_declarations.h, 82
- RTbuffertype
  - optix\_declarations.h, 83
- RTcontext
  - optix\_host.h, 125
- RTcontextattribute
  - optix\_declarations.h, 83
- RTdeviceattribute
  - optix\_declarations.h, 83
- RTexception
  - optix\_declarations.h, 83
- RTfiltermode
  - optix\_declarations.h, 84
- RTformat
  - optix\_declarations.h, 84
- RTgeometry
  - optix\_host.h, 125
- RTgeometrygroup
  - optix\_host.h, 126
- RTgeometryinstance
  - optix\_host.h, 126
- RTgltarget
  - optix\_declarations.h, 85
- RTgroup
  - optix\_host.h, 126
- RTmaterial
  - optix\_host.h, 126
- RTobject
  - optix\_host.h, 126
- RTobjecttype
  - optix\_declarations.h, 85
- RTprogram
  - optix\_host.h, 126
- RTprogramidnull
  - optix\_declarations.h, 86
- RTremotedevice
  - optix\_host.h, 126
- RTremotedeviceattribute
  - optix\_declarations.h, 86
- RTremotedevicestatus
  - optix\_declarations.h, 86
- RTresult
  - optix\_declarations.h, 87
- RTselector
  - optix\_host.h, 126
- RTtextureidnull
  - optix\_declarations.h, 87
- RTtextureindexmode
  - optix\_declarations.h, 88
- RTtexturereadmode
  - optix\_declarations.h, 88
- RTtexturesampler
  - optix\_host.h, 126
- RTtimeoutcallback
  - optix\_host.h, 126
- RTtransform
  - optix\_host.h, 126
- RTtransformflags
  - optix\_defines.h, 89
- RTtransformkind
  - optix\_defines.h, 89
- RTvariable
  - optix\_host.h, 126
- RTwrapmode
  - optix\_declarations.h, 88
- Ray, 39
  - direction, 40
  - origin, 40
  - ray\_type, 40
  - tmax, 40
  - tmin, 40
- ray\_type
  - Ray, 40
- registerD3D9Buffer
  - optix::BufferObj, 14
- registerD3D9Texture
  - optix::TextureSamplerObj, 47
- registerGLTexture
  - optix::TextureSamplerObj, 47
- removeChild
  - optix::GeometryGroupObj, 25



- optix::GroupObj, 29
- optix::SelectorObj, 46
- rotate
  - optix::Matrix, 35
- rtAccelerationCreate
  - optix\_host.h, 127
- rtAccelerationDestroy
  - optix\_host.h, 127
- rtAccelerationGetBuilder
  - optix\_host.h, 128
- rtAccelerationGetContext
  - optix\_host.h, 128
- rtAccelerationGetData
  - optix\_host.h, 128
- rtAccelerationGetDataSize
  - optix\_host.h, 130
- rtAccelerationGetProperty
  - optix\_host.h, 130
- rtAccelerationGetTraverser
  - optix\_host.h, 131
- rtAccelerationIsDirty
  - optix\_host.h, 131
- rtAccelerationMarkDirty
  - optix\_host.h, 132
- rtAccelerationSetBuilder
  - optix\_host.h, 132
- rtAccelerationSetData
  - optix\_host.h, 133
- rtAccelerationSetProperty
  - optix\_host.h, 134
- rtAccelerationSetTraverser
  - optix\_host.h, 135
- rtAccelerationValidate
  - optix\_host.h, 136
- rtBuffer
  - optix\_device.h, 97
- rtBufferBindProgressiveStream
  - optix\_host.h, 136
- rtBufferCreate
  - optix\_host.h, 137
- rtBufferCreateForCUDA
  - optix\_cuda\_interop.h, 56
- rtBufferCreateFromD3D10Resource
  - optix\_d3d10\_interop.h, 59
- rtBufferCreateFromD3D11Resource
  - optix\_d3d11\_interop.h, 65
- rtBufferCreateFromD3D9Resource
  - optix\_d3d9\_interop.h, 71
- rtBufferCreateFromGLBO
  - optix\_gl\_interop.h, 112
- rtBufferD3D10Register
  - optix\_d3d10\_interop.h, 60
- rtBufferD3D10Unregister
  - optix\_d3d10\_interop.h, 60
- rtBufferD3D11Register
  - optix\_d3d11\_interop.h, 66
- rtBufferD3D11Unregister
  - optix\_d3d11\_interop.h, 66
- rtBufferD3D9Register
  - optix\_d3d9\_interop.h, 72
- rtBufferD3D9Unregister
  - optix\_d3d9\_interop.h, 73
- rtBufferDestroy
  - optix\_host.h, 138
- rtBufferGLRegister
  - optix\_gl\_interop.h, 114
- rtBufferGLUnregister
  - optix\_gl\_interop.h, 114
- rtBufferGetAttribute
  - optix\_host.h, 138
- rtBufferGetContext
  - optix\_host.h, 139
- rtBufferGetD3D10Resource
  - optix\_d3d10\_interop.h, 61
- rtBufferGetD3D11Resource
  - optix\_d3d11\_interop.h, 67
- rtBufferGetD3D9Resource
  - optix\_d3d9\_interop.h, 73
- rtBufferGetDevicePointer
  - optix\_cuda\_interop.h, 57
- rtBufferGetDimensionality
  - optix\_host.h, 139
- rtBufferGetElementSize
  - optix\_host.h, 140
- rtBufferGetFormat
  - optix\_host.h, 140
- rtBufferGetGLBOld
  - optix\_gl\_interop.h, 113
- rtBufferGetId
  - optix\_host.h, 141
- rtBufferGetMipLevelCount
  - optix\_host.h, 141
- rtBufferGetMipLevelSize1D
  - optix\_host.h, 142
- rtBufferGetMipLevelSize2D
  - optix\_host.h, 142
- rtBufferGetMipLevelSize3D
  - optix\_host.h, 143
- rtBufferGetProgressiveUpdateReady
  - optix\_host.h, 143
- rtBufferGetSize1D
  - optix\_host.h, 144
- rtBufferGetSize2D
  - optix\_host.h, 144
- rtBufferGetSize3D
  - optix\_host.h, 145
- rtBufferGetSizev
  - optix\_host.h, 145
- rtBufferId
  - optix\_device.h, 98
- rtBufferMap
  - optix\_host.h, 146
- rtBufferMapEx
  - optix\_host.h, 146
- rtBufferMarkDirty
  - optix\_cuda\_interop.h, 57

rtBufferSetAttribute  
    optix\_host.h, 147

rtBufferSetDevicePointer  
    optix\_cuda\_interop.h, 58

rtBufferSetElementSize  
    optix\_host.h, 148

rtBufferSetFormat  
    optix\_host.h, 148

rtBufferSetMipLevelCount  
    optix\_host.h, 150

rtBufferSetSize1D  
    optix\_host.h, 150

rtBufferSetSize2D  
    optix\_host.h, 151

rtBufferSetSize3D  
    optix\_host.h, 151

rtBufferSetSizev  
    optix\_host.h, 152

rtBufferUnmap  
    optix\_host.h, 153

rtBufferUnmapEx  
    optix\_host.h, 153

rtBufferValidate  
    optix\_host.h, 154

rtCallableProgram  
    optix\_device.h, 98

rtCallableProgramId  
    optix\_device.h, 98

rtCallableProgramSizeofWrapper< T >, 41

rtCallableProgramSizeofWrapper< void >, 41

rtCallableProgramX  
    optix\_device.h, 99

rtContextCompile  
    optix\_host.h, 154

rtContextCreate  
    optix\_host.h, 155

rtContextDeclareVariable  
    optix\_host.h, 155

rtContextDestroy  
    optix\_host.h, 156

rtContextGetAttribute  
    optix\_host.h, 156

rtContextGetBufferFromId  
    optix\_host.h, 157

rtContextGetDeviceCount  
    optix\_host.h, 158

rtContextGetDevices  
    optix\_host.h, 158

rtContextGetEntryPointCount  
    optix\_host.h, 159

rtContextGetErrorString  
    optix\_host.h, 159

rtContextGetExceptionEnabled  
    optix\_host.h, 159

rtContextGetExceptionProgram  
    optix\_host.h, 160

rtContextGetMissProgram  
    optix\_host.h, 160

rtContextGetPrintBufferSize  
    optix\_host.h, 161

rtContextGetPrintEnabled  
    optix\_host.h, 161

rtContextGetPrintLaunchIndex  
    optix\_host.h, 162

rtContextGetProgramFromId  
    optix\_host.h, 162

rtContextGetRayGenerationProgram  
    optix\_host.h, 163

rtContextGetRayTypeCount  
    optix\_host.h, 163

rtContextGetRunningState  
    optix\_host.h, 163

rtContextGetStackSize  
    optix\_host.h, 164

rtContextGetTextureSamplerFromId  
    optix\_host.h, 164

rtContextGetVariable  
    optix\_host.h, 165

rtContextGetVariableCount  
    optix\_host.h, 165

rtContextLaunch1D  
    optix\_host.h, 165

rtContextLaunch2D  
    optix\_host.h, 166

rtContextLaunch3D  
    optix\_host.h, 166

rtContextLaunchProgressive2D  
    optix\_host.h, 166

rtContextQueryVariable  
    optix\_host.h, 167

rtContextRemoveVariable  
    optix\_host.h, 168

rtContextSetAttribute  
    optix\_host.h, 168

rtContextSetD3D10Device  
    optix\_d3d10\_interop.h, 61

rtContextSetD3D11Device  
    optix\_d3d11\_interop.h, 67

rtContextSetD3D9Device  
    optix\_d3d9\_interop.h, 74

rtContextSetDevices  
    optix\_host.h, 169

rtContextSetEntryPointCount  
    optix\_host.h, 169

rtContextSetExceptionEnabled  
    optix\_host.h, 170

rtContextSetExceptionProgram  
    optix\_host.h, 171

rtContextSetMissProgram  
    optix\_host.h, 171

rtContextSetPrintBufferSize  
    optix\_host.h, 172

rtContextSetPrintEnabled  
    optix\_host.h, 172

rtContextSetPrintLaunchIndex  
    optix\_host.h, 173



rtContextSetRayGenerationProgram  
optix\_host.h, 173

rtContextSetRayTypeCount  
optix\_host.h, 174

rtContextSetRemoteDevice  
optix\_host.h, 174

rtContextSetStackSize  
optix\_host.h, 175

rtContextSetTimeoutCallback  
optix\_host.h, 175

rtContextStopProgressive  
optix\_host.h, 176

rtContextValidate  
optix\_host.h, 176

rtDeclareAnnotation  
optix\_device.h, 99

rtDeclareVariable  
optix\_device.h, 100

rtDeviceGetAttribute  
optix\_host.h, 177

rtDeviceGetD3D10Device  
optix\_d3d10\_interop.h, 62

rtDeviceGetD3D11Device  
optix\_d3d11\_interop.h, 68

rtDeviceGetD3D9Device  
optix\_d3d9\_interop.h, 74

rtDeviceGetDeviceCount  
optix\_host.h, 178

rtDeviceGetWGLDevice  
optix\_gl\_interop.h, 115

rtGeometryCreate  
optix\_host.h, 178

rtGeometryDeclareVariable  
optix\_host.h, 179

rtGeometryDestroy  
optix\_host.h, 179

rtGeometryGetBoundingBoxProgram  
optix\_host.h, 180

rtGeometryGetContext  
optix\_host.h, 180

rtGeometryGetIntersectionProgram  
optix\_host.h, 181

rtGeometryGetPrimitiveCount  
optix\_host.h, 181

rtGeometryGetPrimitiveIndexOffset  
optix\_host.h, 181

rtGeometryGetVariable  
optix\_host.h, 182

rtGeometryGetVariableCount  
optix\_host.h, 182

rtGeometryGroupCreate  
optix\_host.h, 183

rtGeometryGroupDestroy  
optix\_host.h, 183

rtGeometryGroupGetAcceleration  
optix\_host.h, 184

rtGeometryGroupGetChild  
optix\_host.h, 184

rtGeometryGroupGetChildCount  
optix\_host.h, 185

rtGeometryGroupGetContext  
optix\_host.h, 185

rtGeometryGroupSetAcceleration  
optix\_host.h, 186

rtGeometryGroupSetChild  
optix\_host.h, 186

rtGeometryGroupSetChildCount  
optix\_host.h, 188

rtGeometryGroupValidate  
optix\_host.h, 188

rtGeometryInstanceCreate  
optix\_host.h, 189

rtGeometryInstanceDeclareVariable  
optix\_host.h, 189

rtGeometryInstanceDestroy  
optix\_host.h, 191

rtGeometryInstanceGetContext  
optix\_host.h, 191

rtGeometryInstanceGetGeometry  
optix\_host.h, 192

rtGeometryInstanceGetMaterial  
optix\_host.h, 192

rtGeometryInstanceGetMaterialCount  
optix\_host.h, 193

rtGeometryInstanceGetVariable  
optix\_host.h, 193

rtGeometryInstanceGetVariableCount  
optix\_host.h, 194

rtGeometryInstanceQueryVariable  
optix\_host.h, 194

rtGeometryInstanceRemoveVariable  
optix\_host.h, 195

rtGeometryInstanceSetGeometry  
optix\_host.h, 196

rtGeometryInstanceSetMaterial  
optix\_host.h, 196

rtGeometryInstanceSetMaterialCount  
optix\_host.h, 197

rtGeometryInstanceValidate  
optix\_host.h, 197

rtGeometryIsDirty  
optix\_host.h, 198

rtGeometryMarkDirty  
optix\_host.h, 198

rtGeometryQueryVariable  
optix\_host.h, 198

rtGeometryRemoveVariable  
optix\_host.h, 199

rtGeometrySetBoundingBoxProgram  
optix\_host.h, 200

rtGeometrySetIntersectionProgram  
optix\_host.h, 200

rtGeometrySetPrimitiveCount  
optix\_host.h, 201

rtGeometrySetPrimitiveIndexOffset  
optix\_host.h, 201

`rtGeometryValidate`  
    [optix\\_host.h, 201](#)  
`rtGetExceptionCode`  
    [optix\\_device.h, 101](#)  
`rtGetTransform`  
    [optix\\_device.h, 102](#)  
`rtGetVersion`  
    [optix\\_host.h, 202](#)  
`rtGroupCreate`  
    [optix\\_host.h, 202](#)  
`rtGroupDestroy`  
    [optix\\_host.h, 203](#)  
`rtGroupGetAcceleration`  
    [optix\\_host.h, 203](#)  
`rtGroupGetChild`  
    [optix\\_host.h, 204](#)  
`rtGroupGetChildCount`  
    [optix\\_host.h, 204](#)  
`rtGroupGetChildType`  
    [optix\\_host.h, 204](#)  
`rtGroupGetContext`  
    [optix\\_host.h, 205](#)  
`rtGroupSetAcceleration`  
    [optix\\_host.h, 205](#)  
`rtGroupSetChild`  
    [optix\\_host.h, 206](#)  
`rtGroupSetChildCount`  
    [optix\\_host.h, 206](#)  
`rtGroupValidate`  
    [optix\\_host.h, 207](#)  
`rtIgnoreIntersection`  
    [optix\\_device.h, 102](#)  
`rtIntersectChild`  
    [optix\\_device.h, 102](#)  
`rtMaterialCreate`  
    [optix\\_host.h, 207](#)  
`rtMaterialDeclareVariable`  
    [optix\\_host.h, 208](#)  
`rtMaterialDestroy`  
    [optix\\_host.h, 208](#)  
`rtMaterialGetAnyHitProgram`  
    [optix\\_host.h, 209](#)  
`rtMaterialGetClosestHitProgram`  
    [optix\\_host.h, 209](#)  
`rtMaterialGetContext`  
    [optix\\_host.h, 210](#)  
`rtMaterialGetVariable`  
    [optix\\_host.h, 210](#)  
`rtMaterialGetVariableCount`  
    [optix\\_host.h, 210](#)  
`rtMaterialQueryVariable`  
    [optix\\_host.h, 211](#)  
`rtMaterialRemoveVariable`  
    [optix\\_host.h, 211](#)  
`rtMaterialSetAnyHitProgram`  
    [optix\\_host.h, 212](#)  
`rtMaterialSetClosestHitProgram`  
    [optix\\_host.h, 213](#)  
  
`rtMaterialValidate`  
    [optix\\_host.h, 213](#)  
`rtObject`, [42](#)  
`rtPotentialIntersection`  
    [optix\\_device.h, 103](#)  
`rtPrintExceptionDetails`  
    [optix\\_device.h, 103](#)  
`rtPrintf`  
    [optix\\_device.h, 105–108](#)  
`rtProgramCreateFromPTXFile`  
    [optix\\_host.h, 214](#)  
`rtProgramCreateFromPTXString`  
    [optix\\_host.h, 214](#)  
`rtProgramDeclareVariable`  
    [optix\\_host.h, 215](#)  
`rtProgramDestroy`  
    [optix\\_host.h, 215](#)  
`rtProgramGetContext`  
    [optix\\_host.h, 216](#)  
`rtProgramGetId`  
    [optix\\_host.h, 216](#)  
`rtProgramGetVariable`  
    [optix\\_host.h, 217](#)  
`rtProgramGetVariableCount`  
    [optix\\_host.h, 217](#)  
`rtProgramQueryVariable`  
    [optix\\_host.h, 218](#)  
`rtProgramRemoveVariable`  
    [optix\\_host.h, 218](#)  
`rtProgramValidate`  
    [optix\\_host.h, 219](#)  
`rtRemoteDeviceCreate`  
    [optix\\_host.h, 219](#)  
`rtRemoteDeviceDestroy`  
    [optix\\_host.h, 221](#)  
`rtRemoteDeviceGetAttribute`  
    [optix\\_host.h, 221](#)  
`rtRemoteDeviceRelease`  
    [optix\\_host.h, 223](#)  
`rtRemoteDeviceReserve`  
    [optix\\_host.h, 223](#)  
`rtReportIntersection`  
    [optix\\_device.h, 109](#)  
`rtSelectorCreate`  
    [optix\\_host.h, 224](#)  
`rtSelectorDeclareVariable`  
    [optix\\_host.h, 225](#)  
`rtSelectorDestroy`  
    [optix\\_host.h, 225](#)  
`rtSelectorGetChild`  
    [optix\\_host.h, 226](#)  
`rtSelectorGetChildCount`  
    [optix\\_host.h, 226](#)  
`rtSelectorGetChildType`  
    [optix\\_host.h, 227](#)  
`rtSelectorGetContext`  
    [optix\\_host.h, 227](#)  
`rtSelectorGetVariable`

optix\_host.h, [228](#)  
rtSelectorGetVariableCount  
optix\_host.h, [228](#)  
rtSelectorGetVisitProgram  
optix\_host.h, [229](#)  
rtSelectorQueryVariable  
optix\_host.h, [229](#)  
rtSelectorRemoveVariable  
optix\_host.h, [229](#)  
rtSelectorSetChild  
optix\_host.h, [230](#)  
rtSelectorSetChildCount  
optix\_host.h, [230](#)  
rtSelectorSetVisitProgram  
optix\_host.h, [231](#)  
rtSelectorValidate  
optix\_host.h, [231](#)  
rtSemanticTypes  
optix\_defines.h, [89](#)  
rtTerminateRay  
optix\_device.h, [109](#)  
rtTextureSampler  
optix\_device.h, [101](#)  
rtTextureSamplerCreate  
optix\_host.h, [232](#)  
rtTextureSamplerCreateFromD3D10Resource  
optix\_d3d10\_interop.h, [62](#)  
rtTextureSamplerCreateFromD3D11Resource  
optix\_d3d11\_interop.h, [68](#)  
rtTextureSamplerCreateFromD3D9Resource  
optix\_d3d9\_interop.h, [75](#)  
rtTextureSamplerCreateFromGLImage  
optix\_gl\_interop.h, [115](#)  
rtTextureSamplerD3D10Register  
optix\_d3d10\_interop.h, [63](#)  
rtTextureSamplerD3D10Unregister  
optix\_d3d10\_interop.h, [64](#)  
rtTextureSamplerD3D11Register  
optix\_d3d11\_interop.h, [69](#)  
rtTextureSamplerD3D11Unregister  
optix\_d3d11\_interop.h, [70](#)  
rtTextureSamplerD3D9Register  
optix\_d3d9\_interop.h, [75](#)  
rtTextureSamplerD3D9Unregister  
optix\_d3d9\_interop.h, [76](#)  
rtTextureSamplerDestroy  
optix\_host.h, [232](#)  
rtTextureSamplerGLRegister  
optix\_gl\_interop.h, [117](#)  
rtTextureSamplerGLUnregister  
optix\_gl\_interop.h, [117](#)  
rtTextureSamplerGetArraySize  
optix\_host.h, [233](#)  
rtTextureSamplerGetBuffer  
optix\_host.h, [233](#)  
rtTextureSamplerGetContext  
optix\_host.h, [234](#)  
rtTextureSamplerGetD3D10Resource  
optix\_d3d10\_interop.h, [64](#)  
rtTextureSamplerGetD3D11Resource  
optix\_d3d11\_interop.h, [70](#)  
rtTextureSamplerGetD3D9Resource  
optix\_d3d9\_interop.h, [76](#)  
rtTextureSamplerGetFilteringModes  
optix\_host.h, [234](#)  
rtTextureSamplerGetGLImageId  
optix\_gl\_interop.h, [116](#)  
rtTextureSamplerGetId  
optix\_host.h, [235](#)  
rtTextureSamplerGetIndexingMode  
optix\_host.h, [235](#)  
rtTextureSamplerGetMaxAnisotropy  
optix\_host.h, [236](#)  
rtTextureSamplerGetMipLevelBias  
optix\_host.h, [236](#)  
rtTextureSamplerGetMipLevelClamp  
optix\_host.h, [237](#)  
rtTextureSamplerGetMipLevelCount  
optix\_host.h, [237](#)  
rtTextureSamplerGetReadMode  
optix\_host.h, [237](#)  
rtTextureSamplerGetWrapMode  
optix\_host.h, [238](#)  
rtTextureSamplerSetArraySize  
optix\_host.h, [238](#)  
rtTextureSamplerSetBuffer  
optix\_host.h, [239](#)  
rtTextureSamplerSetFilteringModes  
optix\_host.h, [239](#)  
rtTextureSamplerSetIndexingMode  
optix\_host.h, [240](#)  
rtTextureSamplerSetMaxAnisotropy  
optix\_host.h, [241](#)  
rtTextureSamplerSetMipLevelBias  
optix\_host.h, [241](#)  
rtTextureSamplerSetMipLevelClamp  
optix\_host.h, [241](#)  
rtTextureSamplerSetMipLevelCount  
optix\_host.h, [242](#)  
rtTextureSamplerSetReadMode  
optix\_host.h, [242](#)  
rtTextureSamplerSetWrapMode  
optix\_host.h, [243](#)  
rtTextureSamplerValidate  
optix\_host.h, [244](#)  
rtThrow  
optix\_device.h, [109](#)  
rtTrace  
optix\_device.h, [110](#)  
rtTransformCreate  
optix\_host.h, [244](#)  
rtTransformDestroy  
optix\_host.h, [245](#)  
rtTransformGetChild  
optix\_host.h, [245](#)  
rtTransformGetChildType

optix\_host.h, 246  
rtTransformGetContext  
  optix\_host.h, 246  
rtTransformGetMatrix  
  optix\_host.h, 247  
rtTransformNormal  
  optix\_device.h, 110  
rtTransformPoint  
  optix\_device.h, 111  
rtTransformSetChild  
  optix\_host.h, 247  
rtTransformSetMatrix  
  optix\_host.h, 248  
rtTransformValidate  
  optix\_host.h, 249  
rtTransformVector  
  optix\_device.h, 111  
rtVariableGet1f  
  optix\_host.h, 249  
rtVariableGet1fv  
  optix\_host.h, 250  
rtVariableGet1i  
  optix\_host.h, 250  
rtVariableGet1iv  
  optix\_host.h, 250  
rtVariableGet1ui  
  optix\_host.h, 250  
rtVariableGet1uiv  
  optix\_host.h, 250  
rtVariableGet2f  
  optix\_host.h, 250  
rtVariableGet2fv  
  optix\_host.h, 251  
rtVariableGet2i  
  optix\_host.h, 251  
rtVariableGet2iv  
  optix\_host.h, 251  
rtVariableGet2ui  
  optix\_host.h, 251  
rtVariableGet2uiv  
  optix\_host.h, 251  
rtVariableGet3f  
  optix\_host.h, 251  
rtVariableGet3fv  
  optix\_host.h, 251  
rtVariableGet3i  
  optix\_host.h, 252  
rtVariableGet3iv  
  optix\_host.h, 252  
rtVariableGet3ui  
  optix\_host.h, 252  
rtVariableGet3uiv  
  optix\_host.h, 252  
rtVariableGet4f  
  optix\_host.h, 252  
rtVariableGet4fv  
  optix\_host.h, 252  
rtVariableGet4i  
  optix\_host.h, 252  
rtVariableGet4iv  
  optix\_host.h, 252  
rtVariableGet4ui  
  optix\_host.h, 253  
rtVariableGet4uiv  
  optix\_host.h, 253  
rtVariableGetAnnotation  
  optix\_host.h, 253  
rtVariableGetContext  
  optix\_host.h, 254  
rtVariableGetMatrix2x2fv  
  optix\_host.h, 254  
rtVariableGetMatrix2x3fv  
  optix\_host.h, 254  
rtVariableGetMatrix2x4fv  
  optix\_host.h, 254  
rtVariableGetMatrix3x2fv  
  optix\_host.h, 254  
rtVariableGetMatrix3x3fv  
  optix\_host.h, 255  
rtVariableGetMatrix3x4fv  
  optix\_host.h, 255  
rtVariableGetMatrix4x2fv  
  optix\_host.h, 255  
rtVariableGetMatrix4x3fv  
  optix\_host.h, 255  
rtVariableGetMatrix4x4fv  
  optix\_host.h, 255  
rtVariableGetName  
  optix\_host.h, 255  
rtVariableGetObject  
  optix\_host.h, 256  
rtVariableGetSize  
  optix\_host.h, 256  
rtVariableGetType  
  optix\_host.h, 257  
rtVariableGetUserData  
  optix\_host.h, 258  
rtVariableSet1f  
  optix\_host.h, 259  
rtVariableSet1fv  
  optix\_host.h, 259  
rtVariableSet1i  
  optix\_host.h, 260  
rtVariableSet1iv  
  optix\_host.h, 260  
rtVariableSet1ui  
  optix\_host.h, 260  
rtVariableSet1uiv  
  optix\_host.h, 260  
rtVariableSet2f  
  optix\_host.h, 260  
rtVariableSet2fv  
  optix\_host.h, 260  
rtVariableSet2i  
  optix\_host.h, 260  
rtVariableSet2iv

- optix\_host.h, 260
- rtVariableSet2ui
  - optix\_host.h, 261
- rtVariableSet2uiv
  - optix\_host.h, 261
- rtVariableSet3f
  - optix\_host.h, 261
- rtVariableSet3fv
  - optix\_host.h, 261
- rtVariableSet3i
  - optix\_host.h, 261
- rtVariableSet3iv
  - optix\_host.h, 261
- rtVariableSet3ui
  - optix\_host.h, 261
- rtVariableSet3uiv
  - optix\_host.h, 262
- rtVariableSet4f
  - optix\_host.h, 262
- rtVariableSet4fv
  - optix\_host.h, 262
- rtVariableSet4i
  - optix\_host.h, 262
- rtVariableSet4iv
  - optix\_host.h, 262
- rtVariableSet4ui
  - optix\_host.h, 262
- rtVariableSet4uiv
  - optix\_host.h, 263
- rtVariableSetMatrix2x2fv
  - optix\_host.h, 263
- rtVariableSetMatrix2x3fv
  - optix\_host.h, 263
- rtVariableSetMatrix2x4fv
  - optix\_host.h, 263
- rtVariableSetMatrix3x2fv
  - optix\_host.h, 263
- rtVariableSetMatrix3x3fv
  - optix\_host.h, 263
- rtVariableSetMatrix3x4fv
  - optix\_host.h, 263
- rtVariableSetMatrix4x2fv
  - optix\_host.h, 264
- rtVariableSetMatrix4x3fv
  - optix\_host.h, 264
- rtVariableSetMatrix4x4fv
  - optix\_host.h, 264
- rtVariableSetObject
  - optix\_host.h, 264
- rtVariableSetUserData
  - optix\_host.h, 264
- rti\_internal\_callableprogram::CPArgVoid, 22
- rti\_internal\_callableprogram::check\_is\_CPArgVoid<
  - Condition, Dummy >, 15
- rti\_internal\_callableprogram::check\_is\_CPArgVoid<
  - false, IntentionalError >, 15
- rti\_internal\_callableprogram::is\_CPArgVoid< CPArg-
  - Void >, 31
- rti\_internal\_callableprogram::is\_CPArgVoid< T1 >, 31
- rti\_internal\_typeinfo::rti\_typeenum< optix::bound-
  - CallableProgramId< T > >, 42
- rti\_internal\_typeinfo::rti\_typeenum< optix::callable-
  - ProgramId< T > >, 42
- rti\_internal\_typeinfo::rti\_typeenum< T >, 42
- rti\_internal\_typeinfo::rti\_typeinfo, 42
- rtpBufferDescCreate
  - optix\_prime.h, 267
- rtpBufferDescDestroy
  - optix\_prime.h, 267
- rtpBufferDescGetContext
  - optix\_prime.h, 268
- rtpBufferDescSetCudaDeviceNumber
  - optix\_prime.h, 268
- rtpBufferDescSetRange
  - optix\_prime.h, 268
- rtpBufferDescSetStride
  - optix\_prime.h, 269
- rtpContextCreate
  - optix\_prime.h, 269
- rtpContextDestroy
  - optix\_prime.h, 270
- rtpContextGetLastErrorString
  - optix\_prime.h, 270
- rtpContextSetCpuThreads
  - optix\_prime.h, 270
- rtpContextSetCudaDeviceNumbers
  - optix\_prime.h, 271
- rtpGetErrorString
  - optix\_prime.h, 271
- rtpGetVersion
  - optix\_prime.h, 271
- rtpGetVersionString
  - optix\_prime.h, 272
- rtpHostBufferLock
  - optix\_prime.h, 272
- rtpHostBufferUnlock
  - optix\_prime.h, 272
- rtpModelCopy
  - optix\_prime.h, 273
- rtpModelCreate
  - optix\_prime.h, 273
- rtpModelDestroy
  - optix\_prime.h, 273
- rtpModelFinish
  - optix\_prime.h, 274
- rtpModelGetContext
  - optix\_prime.h, 274
- rtpModelGetFinished
  - optix\_prime.h, 274
- rtpModelSetBuilderParameter
  - optix\_prime.h, 274
- rtpModelSetInstances
  - optix\_prime.h, 275
- rtpModelSetTriangles
  - optix\_prime.h, 276
- rtpModelUpdate

- optix\_prime.h, 276
- rtpQueryCreate
  - optix\_prime.h, 277
- rtpQueryDestroy
  - optix\_prime.h, 277
- rtpQueryExecute
  - optix\_prime.h, 277
- rtpQueryFinish
  - optix\_prime.h, 278
- rtpQueryGetContext
  - optix\_prime.h, 278
- rtpQueryGetFinished
  - optix\_prime.h, 279
- rtpQuerySetCudaStream
  - optix\_prime.h, 279
- rtpQuerySetHits
  - optix\_prime.h, 279
- rtpQuerySetRays
  - optix\_prime.h, 279
- rtuCUDACompileFile
  - optixu.h, 290
- rtuCUDACompileString
  - optixu.h, 291
- rtuCUDAGetCompileResult
  - optixu.h, 291
- rtuCreateClusteredMesh
  - optixu.h, 289
- rtuCreateClusteredMeshExt
  - optixu.h, 290
- rtuGeometryGroupAddChild
  - optixu.h, 291
- rtuGeometryGroupGetChildIndex
  - optixu.h, 291
- rtuGeometryGroupRemoveChild
  - optixu.h, 291
- rtuGeometryGroupRemoveChildByIndex
  - optixu.h, 292
- rtuGetSizeForRTformat
  - optixu.h, 292
- rtuGroupAddChild
  - optixu.h, 292
- rtuGroupGetChildIndex
  - optixu.h, 292
- rtuGroupRemoveChild
  - optixu.h, 292
- rtuGroupRemoveChildByIndex
  - optixu.h, 292
- rtuNameForType
  - optixu.h, 292
- rtuSelectorAddChild
  - optixu.h, 292
- rtuSelectorGetChildIndex
  - optixu.h, 292
- rtuSelectorRemoveChild
  - optixu.h, 293
- rtuSelectorRemoveChildByIndex
  - optixu.h, 293
- rtuTransformGetChild
  - optixu.h, 293
- rtuTransformGetChildType
  - optixu.h, 293
- rtuTransformSetChild
  - optixu.h, 293
- rtuTraversalCreate
  - optixu\_traversal.h, 306
- rtuTraversalDestroy
  - optixu\_traversal.h, 307
- rtuTraversalGetAccelData
  - optixu\_traversal.h, 307
- rtuTraversalGetAccelDataSize
  - optixu\_traversal.h, 307
- rtuTraversalGetErrorString
  - optixu\_traversal.h, 307
- rtuTraversalMapOutput
  - optixu\_traversal.h, 307
- rtuTraversalMapRays
  - optixu\_traversal.h, 308
- rtuTraversalMapResults
  - optixu\_traversal.h, 308
- rtuTraversalPreprocess
  - optixu\_traversal.h, 308
- rtuTraversalSetAccelData
  - optixu\_traversal.h, 308
- rtuTraversalSetMesh
  - optixu\_traversal.h, 308
- rtuTraversalSetOption
  - optixu\_traversal.h, 309
- rtuTraversalSetTriangles
  - optixu\_traversal.h, 309
- rtuTraversalTraverse
  - optixu\_traversal.h, 309
- rtuTraversalUnmapOutput
  - optixu\_traversal.h, 309
- rtuTraversalUnmapRays
  - optixu\_traversal.h, 309
- rtuTraversalUnmapResults
  - optixu\_traversal.h, 309
- scale
  - optix::Matrix, 35
- set
  - optix::Aabb, 6
- setAcceleration
  - optix::GeometryGroupObj, 25
  - optix::GroupObj, 29
- setAttribute
  - optix::BufferObj, 14
- setBoundingBoxProgram
  - optix::GeometryObj, 28
- setBuffer
  - optix::TextureSamplerObj, 47
- setBuilderParameter
  - optix::prime::ModelObj, 36
- setCPUNumThreads
  - optix::ContextObj, 21
- setChild
  - optix::TransformObj, 48

- setChildCount
  - optix::GeometryGroupObj, 25
  - optix::GroupObj, 29
  - optix::SelectorObj, 46
- setClosestHitProgram
  - optix::MaterialObj, 32
- setCol
  - optix::Matrix, 35
- setDevices
  - optix::ContextObj, 21
- setFormat
  - optix::BufferObj, 14
- setGeometry
  - optix::GeometryInstanceObj, 26
- setInstances
  - optix::prime::ModelObj, 36
- setMatrix
  - optix::TransformObj, 48
- setMipLevelCount
  - optix::TextureSamplerObj, 48
- setPrimitiveCount
  - optix::GeometryObj, 28
- setPrimitiveIndexOffset
  - optix::GeometryObj, 28
- setPrintEnabled
  - optix::ContextObj, 21
- setProperty
  - optix::AccelerationObj, 8
- setRayGenerationProgram
  - optix::ContextObj, 21
- setRow
  - optix::Matrix, 35
- setSize
  - optix::BufferObj, 14
- setStackSize
  - optix::ContextObj, 21
- setTimeoutCallback
  - optix::ContextObj, 21
- setTriangles
  - optix::prime::ModelObj, 36, 37
- setVisitProgram
  - optix::SelectorObj, 46
- signedDistance
  - optix::Aabb, 6
- t
  - RTUtraversalresult, 43
- take
  - optix::Handle, 31
- tmax
  - Ray, 40
- tmin
  - Ray, 40
- translate
  - optix::Matrix, 35
- transpose
  - optix::Matrix, 35
- valid
  - optix::Aabb, 6
  - volume
    - optix::Aabb, 7