

roctracer  
4.1.60201

Generated by Doxygen 1.8.11

Tue Sep 17 2024 00:10:10



# Contents

<b>1</b>	<b>ROC Tracer API Specification</b>	<b>1</b>
1.1	Introduction	1
1.2	Known Limitations and Restrictions	1
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Module Documentation</b>	<b>5</b>
3.1	Symbol Versions	5
3.1.1	Detailed Description	5
3.1.2	Macro Definition Documentation	5
3.1.2.1	ROCTRACER_VERSION_4_1	5
3.2	Versioning	6
3.2.1	Detailed Description	6
3.2.2	Macro Definition Documentation	6
3.2.2.1	ROCTRACER_VERSION_MAJOR	6
3.2.2.2	ROCTRACER_VERSION_MINOR	6
3.2.3	Function Documentation	6
3.2.3.1	roctracer_version_major() ROCTRACER_VERSION_4_1	6
3.2.3.2	roctracer_version_minor() ROCTRACER_VERSION_4_1	6
3.3	Status Codes	7
3.3.1	Detailed Description	7
3.3.2	Enumeration Type Documentation	7
3.3.2.1	roctracer_status_t	7
3.3.3	Function Documentation	8
3.3.3.1	roctracer_error_string() ROCTRACER_VERSION_4_1	8
3.4	Traced Runtime Domains	9
3.4.1	Detailed Description	9
3.4.2	Typedef Documentation	9
3.4.2.1	roctracer_domain_t	9
3.4.3	Function Documentation	9
3.4.3.1	roctracer_op_code(uint32_t domain, const char *str, uint32_t *op, uint32_t *kind) ROCTRACER_VERSION_4_1	9
3.4.3.2	roctracer_op_string(uint32_t domain, uint32_t op, uint32_t kind) ROCTRACER_VERSION_4_1	10
3.4.3.3	roctracer_set_properties(roctracer_domain_t domain, void *properties) ROCTRACER_VERSION_4_1	10
3.5	Callback API	11
3.5.1	Detailed Description	11
3.5.2	Typedef Documentation	11
3.5.2.1	roctracer_rtapi_callback_t	11
3.5.3	Function Documentation	11

3.5.3.1	roctracer_disable_domain_callback(activity_domain_t domain) ROCTRACER_VERSION_4_1	11
3.5.3.2	roctracer_disable_op_callback(activity_domain_t domain, uint32_t op) ROCTRACER_VERSION_4_1	12
3.5.3.3	roctracer_enable_domain_callback(activity_domain_t domain, activity_rtapi_callback_t callback, void *arg) ROCTRACER_VERSION_4_1	12
3.5.3.4	roctracer_enable_op_callback(activity_domain_t domain, uint32_t op, activity_rtapi_callback_t callback, void *arg) ROCTRACER_VERSION_4_1	13
3.6	Activity API	14
3.6.1	Detailed Description	15
3.6.2	Typedef Documentation	15
3.6.2.1	roctracer_allocator_t	15
3.6.2.2	roctracer_buffer_callback_t	16
3.6.2.3	roctracer_pool_t	16
3.6.2.4	roctracer_record_t	16
3.6.3	Function Documentation	16
3.6.3.1	roctracer_close_pool() ROCTRACER_VERSION_4_1	16
3.6.3.2	roctracer_close_pool_expl(roctracer_pool_t *pool) ROCTRACER_VERSION_4_1	16
3.6.3.3	roctracer_default_pool() ROCTRACER_VERSION_4_1	17
3.6.3.4	roctracer_default_pool_expl(roctracer_pool_t *pool) ROCTRACER_VERSION_4_1	17
3.6.3.5	roctracer_disable_domain_activity(activity_domain_t domain) ROCTRACER_VERSION_4_1	17
3.6.3.6	roctracer_disable_op_activity(activity_domain_t domain, uint32_t op) ROCTRACER_VERSION_4_1	18
3.6.3.7	roctracer_enable_domain_activity(activity_domain_t domain) ROCTRACER_VERSION_4_1	18
3.6.3.8	roctracer_enable_domain_activity_expl(activity_domain_t domain, roctracer_pool_t *pool) ROCTRACER_VERSION_4_1	18
3.6.3.9	roctracer_enable_op_activity(activity_domain_t domain, uint32_t op) ROCTRACER_VERSION_4_1	19
3.6.3.10	roctracer_enable_op_activity_expl(activity_domain_t domain, uint32_t op, roctracer_pool_t *pool) ROCTRACER_VERSION_4_1	19
3.6.3.11	roctracer_flush_activity() ROCTRACER_VERSION_4_1	19
3.6.3.12	roctracer_flush_activity_expl(roctracer_pool_t *pool) ROCTRACER_VERSION_4_1	20
3.6.3.13	roctracer_next_record(const activity_record_t *record, const activity_record_t **next) ROCTRACER_VERSION_4_1	20
3.6.3.14	roctracer_open_pool(const roctracer_properties_t *properties) ROCTRACER_VERSION_4_1	20
3.6.3.15	roctracer_open_pool_expl(const roctracer_properties_t *properties, roctracer_pool_t **pool) ROCTRACER_VERSION_4_1	21
3.7	Timestamp Operations	22
3.7.1	Detailed Description	22
3.7.2	Function Documentation	22
3.7.2.1	roctracer_get_timestamp(roctracer_timestamp_t *timestamp) ROCTRACER_VERSION_4_1	22
3.8	Initialization and Finalization	23
3.8.1	Detailed Description	23
3.8.2	Function Documentation	23
3.8.2.1	roctracer_plugin_finalize()	23
3.8.2.2	roctracer_plugin_initialize(uint32_t roctracer_major_version, uint32_t roctracer_minor_version)	23
3.9	Trace data reporting	25
3.9.1	Detailed Description	25
3.9.2	Function Documentation	25

3.9.2.1	roctracer_plugin_write_activity_records(const roctracer_record_t *begin, const roctracer_record_t *end)	25
3.9.2.2	roctracer_plugin_write_callback_record(const roctracer_record_t *record, const void *callback_data)	25
<b>4</b>	<b>Data Structure Documentation</b>	<b>27</b>
4.1	roctracer_properties_t Struct Reference	27
4.1.1	Detailed Description	27
4.1.2	Field Documentation	28
4.1.2.1	alloc_arg	28
4.1.2.2	alloc_fun	28
4.1.2.3	buffer_callback_arg	28
4.1.2.4	buffer_callback_fun	28
4.1.2.5	buffer_size	28
4.1.2.6	mode	28
<b>5</b>	<b>File Documentation</b>	<b>29</b>
5.1	/long_pathname_so_that_rpms_can_package_the_debug_info/src/roctracer/inc/roctracer.h File Reference	29
5.1.1	Detailed Description	33
5.1.2	Macro Definition Documentation	33
5.1.2.1	ROCTRACER_API	33
5.1.2.2	ROCTRACER_CALL	33
5.1.2.3	ROCTRACER_EXPORT	33
5.1.2.4	ROCTRACER_IMPORT	33
5.2	/long_pathname_so_that_rpms_can_package_the_debug_info/src/roctracer/inc/roctracer_plugin.h File Reference	33
5.2.1	Detailed Description	34
5.2.2	ROctracer Plugin API	34
	<b>Index</b>	<b>35</b>



## Chapter 1

# ROC Tracer API Specification

### 1.1 Introduction

ROCracer library, Runtimes Generic Callback/Activity APIs.

The goal of the implementation is to provide a generic independent from specific runtime profiler to trace API and asynchronous activity.

The API provides functionality for registering the runtimes API callbacks and asynchronous activity records pool support.

### 1.2 Known Limitations and Restrictions

The ROCtracer API library implementation currently has the following restrictions. Future releases aim to address these restrictions.

1. The ACTIVITY\_DOMAIN\_HSA\_OPS operations HSA\_OP\_ID\_DISPATCH, HSA\_OP\_ID\_BARRIER, and HSA\_OP\_ID\_RESERVED1 are not currently implemented.





## Chapter 2

## Todo List

Global `roctracer_op_string` (uint32\_t domain, uint32\_t op, uint32\_t kind) ROCTRACER\_VERSION\_4\_1  
Define kind.



## Chapter 3

# Module Documentation

### 3.1 Symbol Versions

The names used for the shared library versioned symbols.

#### Macros

- `#define ROCTRACER_VERSION_4_1`

*The function was introduced in version 4.1 of the interface and has the symbol version string of "ROCTRACER\_4.1".*

#### 3.1.1 Detailed Description

The names used for the shared library versioned symbols.

Every function is annotated with one of the version macros defined in this section. Each macro specifies a corresponding symbol version string. After dynamically loading the shared library with `dlopen`, the address of each function can be obtained using `dlvsym` with the name of the function and its corresponding symbol version string. An error will be reported by `dlvsym` if the installed library does not support the version for the function specified in this version of the interface.

#### 3.1.2 Macro Definition Documentation

##### 3.1.2.1 `#define ROCTRACER_VERSION_4_1`

The function was introduced in version 4.1 of the interface and has the symbol version string of "ROCTRACER\_4.1".

## 3.2 Versioning

Version information about the interface and the associated installed library.

### Macros

- `#define ROCTRACER_VERSION_MAJOR 4`  
*The major version of the interface as a macro so it can be used by the preprocessor.*
- `#define ROCTRACER_VERSION_MINOR 1`  
*The minor version of the interface as a macro so it can be used by the preprocessor.*

### Functions

- `ROCTRACER_API uint32_t roctracer_version_major () ROCTRACER_VERSION_4_1`  
*Query the major version of the installed library.*
- `ROCTRACER_API uint32_t roctracer_version_minor () ROCTRACER_VERSION_4_1`  
*Query the minor version of the installed library.*

#### 3.2.1 Detailed Description

Version information about the interface and the associated installed library.

The semantic version of the interface following semver.org rules. A client that uses this interface is only compatible with the installed library if the major version numbers match and the interface minor version number is less than or equal to the installed library minor version number.

#### 3.2.2 Macro Definition Documentation

##### 3.2.2.1 `#define ROCTRACER_VERSION_MAJOR 4`

The major version of the interface as a macro so it can be used by the preprocessor.

##### 3.2.2.2 `#define ROCTRACER_VERSION_MINOR 1`

The minor version of the interface as a macro so it can be used by the preprocessor.

#### 3.2.3 Function Documentation

##### 3.2.3.1 `ROCTRACER_API uint32_t roctracer_version_major ( )`

Query the major version of the installed library.

Return the major version of the installed library. This can be used to check if it is compatible with this interface version. This function can be used even when the library is not initialized.

##### 3.2.3.2 `ROCTRACER_API uint32_t roctracer_version_minor ( )`

Query the minor version of the installed library.

Return the minor version of the installed library. This can be used to check if it is compatible with this interface version. This function can be used even when the library is not initialized.

## 3.3 Status Codes

Most operations return a status code to indicate success or error.

### Enumerations

- enum `roctracer_status_t` {  
`ROCTRACER_STATUS_SUCCESS` = 0, `ROCTRACER_STATUS_ERROR` = -1, `ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID` = -2, `ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT` = -3,  
`ROCTRACER_STATUS_ERROR_DEFAULT_POOL_UNDEFINED` = -4, `ROCTRACER_STATUS_ERROR_DEFAULT_POOL_ALREADY_DEFINED` = -5, `ROCTRACER_STATUS_ERROR_MEMORY_ALLOCATION` = -6,  
`ROCTRACER_STATUS_ERROR_MISMATCHED_EXTERNAL_CORRELATION_ID` = -7,  
`ROCTRACER_STATUS_ERROR_NOT_IMPLEMENTED` = -8, `ROCTRACER_STATUS_UNINIT` = 2, `ROCTRACER_STATUS_BREAK` = 3, `ROCTRACER_STATUS_BAD_DOMAIN` = `ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID`,  
`ROCTRACER_STATUS_BAD_PARAMETER` = `ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT`, `ROCTRACER_STATUS_HIP_API_ERR` = 6, `ROCTRACER_STATUS_HIP_OPS_ERR` = 7, `ROCTRACER_STATUS_HCC_OPS_ERR` = `ROCTRACER_STATUS_HIP_OPS_ERR`,  
`ROCTRACER_STATUS_HSA_ERR` = 7, `ROCTRACER_STATUS_ROCTX_ERR` = 8 }

*ROC Tracer API status codes.*

### Functions

- `ROCTRACER_API` const char \* `roctracer_error_string()` `ROCTRACER_VERSION_4_1`

*Query the textual description of the last error for the current thread.*

#### 3.3.1 Detailed Description

Most operations return a status code to indicate success or error.

#### 3.3.2 Enumeration Type Documentation

##### 3.3.2.1 enum roctracer\_status\_t

ROC Tracer API status codes.

##### Enumerator

**`ROCTRACER_STATUS_SUCCESS`** The function has executed successfully.

**`ROCTRACER_STATUS_ERROR`** A generic error has occurred.

**`ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID`** The domain ID is invalid.

**`ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT`** An invalid argument was given to the function.

**`ROCTRACER_STATUS_ERROR_DEFAULT_POOL_UNDEFINED`** No default pool is defined.

**ROCTRACER\_STATUS\_ERROR\_DEFAULT\_POOL\_ALREADY\_DEFINED** The default pool is already defined.

**ROCTRACER\_STATUS\_ERROR\_MEMORY\_ALLOCATION** Memory allocation error.

**ROCTRACER\_STATUS\_ERROR\_MISMATCHED\_EXTERNAL\_CORRELATION\_ID** External correlation ID pop mismatch.

**ROCTRACER\_STATUS\_ERROR\_NOT\_IMPLEMENTED** The operation is not currently implemented. This error may be reported by any function. Check the [Known Limitations and Restrictions](#) section to determine the status of the library implementation of the interface.

**ROCTRACER\_STATUS\_UNINIT** Deprecated error code.

**ROCTRACER\_STATUS\_BREAK** Deprecated error code.

**ROCTRACER\_STATUS\_BAD\_DOMAIN** Deprecated error code.

**ROCTRACER\_STATUS\_BAD\_PARAMETER** Deprecated error code.

**ROCTRACER\_STATUS\_HIP\_API\_ERR** Deprecated error code.

**ROCTRACER\_STATUS\_HIP\_OPS\_ERR** Deprecated error code.

**ROCTRACER\_STATUS\_HCC\_OPS\_ERR** Deprecated error code.

**ROCTRACER\_STATUS\_HSA\_ERR** Deprecated error code.

**ROCTRACER\_STATUS\_ROCTX\_ERR** Deprecated error code.

### 3.3.3 Function Documentation

#### 3.3.3.1 **ROCTRACER\_API** `const char* roctracer_error_string ( )`

Query the textual description of the last error for the current thread.

Returns a NUL terminated string describing the error of the last ROC Tracer API call by the calling thread that did not return success. The empty string is returned if there is no previous error. The last error is not cleared.

#### Returns

Return the error string. The caller owns the returned string and should use `free()` to deallocate it.

## 3.4 Traced Runtime Domains

The ROC Tracer API can trace multiple runtime libraries.

### Typedefs

- typedef activity\_domain\_t [roctracer\\_domain\\_t](#)  
*Enumeration of domains that can be traced.*

### Functions

- [ROCTRACER\\_API](#) const char \* [roctracer\\_op\\_string](#) (uint32\_t domain, uint32\_t op, uint32\_t kind) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query textual name of an operation of a domain.*
- [ROCTRACER\\_API](#) roctracer\_status\_t [roctracer\\_op\\_code](#) (uint32\_t domain, const char \*str, uint32\_t \*op, uint32\_t \*kind) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query the operation code given a domain and the name of an operation.*
- [ROCTRACER\\_API](#) roctracer\_status\_t [roctracer\\_set\\_properties](#) ([roctracer\\_domain\\_t](#) domain, void \*properties) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Set the properties of a domain.*

### 3.4.1 Detailed Description

The ROC Tracer API can trace multiple runtime libraries.

Each library can have API operations and asynchronous operations that can be traced.

### 3.4.2 Typedef Documentation

#### 3.4.2.1 typedef activity\_domain\_t roctracer\_domain\_t

Enumeration of domains that can be traced.

### 3.4.3 Function Documentation

#### 3.4.3.1 [ROCTRACER\\_API](#) roctracer\_status\_t [roctracer\\_op\\_code](#) ( uint32\_t *domain*, const char \* *str*, uint32\_t \* *op*, uint32\_t \* *kind* )

Query the operation code given a domain and the name of an operation.

## Parameters

in	<i>domain</i>	The domain being queried.
in	<i>str</i>	The NUL terminated name of the operation name being queried.
out	<i>op</i>	The operation code.
out	<i>kind</i>	If not NULL then the operation kind code.

## Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully. <i>op</i> and <i>kind</i> have been updated.
<a href="#"><i>ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT</i></a>	The <i>op</i> is invalid for <i>domain</i> .
<a href="#"><i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i></a>	The domain is invalid or not supported.

3.4.3.2 **ROCTRACER\_API** const char\* roctracer\_op\_string ( uint32\_t *domain*, uint32\_t *op*, uint32\_t *kind* )

Query textual name of an operation of a domain.

## Parameters

in	<i>domain</i>	Domain being queried.
in	<i>op</i>	Operation within domain.
in	<i>kind</i>	

3.4.3.3 **ROCTRACER\_API** roctracer\_status\_t roctracer\_set\_properties ( roctracer\_domain\_t *domain*, void \* *properties* )

Set the properties of a domain.

## Parameters

in	<i>domain</i>	The domain.
in	<i>properties</i>	The properties. Each domain defines its own type for the properties. Some domains require the properties to be set before they can be enabled.

## Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
---	--



## 3.5 Callback API

ROC tracer provides support for runtime API callbacks and activity records logging.

### Typedefs

- `typedef activity_rtapi_callback_t roctracer_rtapi_callback_t`  
*Runtime API callback type.*

### Functions

- `ROCTRACER_API roctracer_status_t roctracer_enable_op_callback (activity_domain_t domain, uint32_t op, activity_rtapi_callback_t callback, void *arg) ROCTRACER_VERSION_4_1`  
*Enable runtime API callback for a specific operation of a domain.*
- `ROCTRACER_API roctracer_status_t roctracer_enable_domain_callback (activity_domain_t domain, activity_rtapi_callback_t callback, void *arg) ROCTRACER_VERSION_4_1`  
*Enable runtime API callback for all operations of a domain.*
- `ROCTRACER_API roctracer_status_t roctracer_disable_op_callback (activity_domain_t domain, uint32_t op) ROCTRACER_VERSION_4_1`  
*Disable runtime API callback for a specific operation of a domain.*
- `ROCTRACER_API roctracer_status_t roctracer_disable_domain_callback (activity_domain_t domain) ROCTRACER_VERSION_4_1`  
*Disable runtime API callback for all operations of a domain.*

### 3.5.1 Detailed Description

ROC tracer provides support for runtime API callbacks and activity records logging.

The API callbacks provide the API calls arguments and are called on different phases, on enter, on exit, on kernel completion.

### 3.5.2 Typedef Documentation

#### 3.5.2.1 `typedef activity_rtapi_callback_t roctracer_rtapi_callback_t`

Runtime API callback type.

The callback that will be invoked when an enabled runtime API is called. The callback is invoked on entry and on exit.

### 3.5.3 Function Documentation

#### 3.5.3.1 `ROCTRACER_API roctracer_status_t roctracer_disable_domain_callback ( activity_domain_t domain )`

Disable runtime API callback for all operations of a domain.

## Parameters

<i>domain</i>	The domain
---------------	------------

## Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<a href="#"><i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i></a>	<i>domain</i> is invalid.

### 3.5.3.2 ROCTRACER\_API roctracer\_status\_t roctracer\_disable\_op\_callback ( activity\_domain\_t *domain*, uint32\_t *op* )

Disable runtime API callback for a specific operation of a domain.

## Parameters

<i>domain</i>	The domain
<i>op</i>	The operation in <i>domain</i> .

## Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<a href="#"><i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i></a>	<i>domain</i> is invalid.
<a href="#"><i>ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT</i></a>	<i>op</i> is invalid for <i>domain</i> .

### 3.5.3.3 ROCTRACER\_API roctracer\_status\_t roctracer\_enable\_domain\_callback ( activity\_domain\_t *domain*, activity\_rtapi\_callback\_t *callback*, void \* *arg* )

Enable runtime API callback for all operations of a domain.

## Parameters

<i>domain</i>	The domain
<i>callback</i>	The callback to invoke each time the operation is performed on entry and exit.
<i>arg</i>	Value to pass as last argument of <i>callback</i> .

## Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<a href="#"><i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i></a>	<i>domain</i> is invalid.

3.5.3.4 **ROCTRACER\_API** `roctracer_status_t roctracer_enable_op_callback ( activity_domain_t domain, uint32_t op, activity_rtapi_callback_t callback, void * arg )`

Enable runtime API callback for a specific operation of a domain.

Parameters

<i>domain</i>	The domain.
<i>op</i>	The operation ID in <code>domain</code> .
<i>callback</i>	The callback to invoke each time the operation is performed on entry and exit.
<i>arg</i>	Value to pass as last argument of <code>callback</code> .

Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<a href="#"><i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i></a>	<code>domain</code> is invalid.
<a href="#"><i>ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT</i></a>	<code>op</code> is invalid for <code>domain</code> .

## 3.6 Activity API

The activity records are asynchronously logged to the pool and can be associated with the respective API callbacks using the correlation ID.

### Data Structures

- struct [roctracer\\_properties\\_t](#)  
*Memory pool properties.*

### Typedefs

- typedef activity\_record\_t [roctracer\\_record\\_t](#)  
*Activity record.*
- typedef void(\* [roctracer\\_allocator\\_t](#)) (char \*\*ptr, size\_t size, void \*arg)  
*Memory pool allocator callback.*
- typedef void(\* [roctracer\\_buffer\\_callback\\_t](#)) (const char \*begin, const char \*end, void \*arg)  
*Memory pool buffer callback.*
- typedef void [roctracer\\_pool\\_t](#)  
*Tracer memory pool type.*

### Functions

- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_next\\_record](#) (const activity\_record\_t \*record, const activity\_record\_t \*\*next) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Get a pointer to the next activity record.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_open\\_pool\\_expl](#) (const [roctracer\\_properties\\_t](#) \*properties, [roctracer\\_pool\\_t](#) \*\*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Create tracer memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_open\\_pool](#) (const [roctracer\\_properties\\_t](#) \*properties) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Create tracer memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_close\\_pool\\_expl](#) ([roctracer\\_pool\\_t](#) \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Close tracer memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_close\\_pool](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Close default tracer memory pool, if defined, and set to undefined.*
- [ROCTRACER\\_API roctracer\\_pool\\_t \\* roctracer\\_default\\_pool\\_expl](#) ([roctracer\\_pool\\_t](#) \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query and set the default memory pool.*
- [ROCTRACER\\_API roctracer\\_pool\\_t \\* roctracer\\_default\\_pool](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query the current default memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_enable\\_op\\_activity\\_expl](#) (activity\_domain\_t domain, uint32\_t op, [roctracer\\_pool\\_t](#) \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable activity record logging for a specified operation of a domain providing a memory pool.*

- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_enable\\_op\\_activity](#) (activity\_domain\_t domain, uint32\_t op) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable activity record logging for a specified operation of a domain using the default memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_enable\\_domain\\_activity\\_expl](#) (activity\_domain\_t domain, roctracer\_pool\_t \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable activity record logging for all operations of a domain providing a memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_enable\\_domain\\_activity](#) (activity\_domain\_t domain) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable activity record logging for all operations of a domain using the default memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_disable\\_op\\_activity](#) (activity\_domain\_t domain, uint32\_t op) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Disable activity record logging for a specified operation of a domain.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_disable\\_domain\\_activity](#) (activity\_domain\_t domain) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Disable activity record logging for all operations of a domain.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_flush\\_activity\\_expl](#) (roctracer\_pool\_t \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Flush available activity records for a memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_flush\\_activity](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Flush available activity records for the default memory pool.*

### 3.6.1 Detailed Description

The activity records are asynchronously logged to the pool and can be associated with the respective API callbacks using the correlation ID.

Activity API can be used to enable collecting of the records with timestamping data for API calls and the kernel submits.

### 3.6.2 Typedef Documentation

#### 3.6.2.1 typedef void(\* roctracer\_allocator\_t) (char \*\*ptr, size\_t size, void \*arg)

Memory pool allocator callback.

If `*ptr` is NULL, then allocate memory of `size` bytes and save address in `*ptr`.

If `*ptr` is non-NULL and `size` is non-0, then reallocate the memory at `*ptr` with `size` and save the address in `*ptr`. The memory will have been allocated by the same callback.

If `*ptr` is non-NULL and `size` is 0, then deallocate the memory at `*ptr`. The memory will have been allocated by the same callback.

`size` is the size of the memory allocation or reallocation, or 0 if deallocating.

`arg` Argument provided in the [roctracer\\_properties\\_t](#) passed to the [roctracer\\_open\\_pool](#) function.

### 3.6.2.2 `typedef void(* roctracer_buffer_callback_t) (const char *begin, const char *end, void *arg)`

Memory pool buffer callback.

The callback that will be invoked when a memory pool buffer becomes full or is flushed.

`begin` pointer to first entry in the buffer.

`end` pointer to one past the end entry in the buffer.

`arg` the argument specified when the callback was defined.

### 3.6.2.3 `typedef void roctracer_pool_t`

Tracer memory pool type.

### 3.6.2.4 `typedef activity_record_t roctracer_record_t`

Activity record.

Asynchronous activity events generate activity records.

## 3.6.3 Function Documentation

### 3.6.3.1 `ROCTRACER_API roctracer_status_t roctracer_close_pool ( )`

Close default tracer memory pool, if defined, and set to undefined.

All enabled activities that use the pool must have completed writing to the pool, before deleting the pool. Deleting a pool automatically disables any activities that specify the pool, and flushes it.

Return values

<code>ROCTRACER_STATUS_SUCCESS</code>	The function has been executed successfully or there is no default pool.
---------------------------------------	--

### 3.6.3.2 `ROCTRACER_API roctracer_status_t roctracer_close_pool_expl ( roctracer_pool_t * pool )`

Close tracer memory pool.

All enabled activities that use the pool must have completed writing to the pool, before deleting the pool. Deleting a pool automatically disables any activities that specify the pool, and flushes it.

## Parameters

in	<i>pool</i>	Memory pool to close. If NULL, the default memory pool is closed if defined. The default memory pool is set to undefined if closed.
----	-------------	---

## Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully or pool was NULL and there is no default pool.
---	--

3.6.3.3 **ROCTRACER\_API** `roctracer_pool_t* roctracer_default_pool ( )`

Query the current default memory pool.

## Returns

Return the current default memory pool, or NULL if none is defined.

3.6.3.4 **ROCTRACER\_API** `roctracer_pool_t* roctracer_default_pool_expl ( roctracer_pool_t * pool )`

Query and set the default memory pool.

## Parameters

in	<i>pool</i>	If not NULL, change the current default pool to <i>pool</i> . If NULL, the default pool is not changed.
----	-------------	---

## Returns

Return the current default memory pool before any change, or NULL if none is defined.

3.6.3.5 **ROCTRACER\_API** `roctracer_status_t roctracer_disable_domain_activity ( activity_domain_t domain )`

Disable activity record logging for all operations of a domain.

## Parameters

in	<i>domain</i>	The domain.
----	---------------	-------------

## Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
---	--

### 3.6.3.6 ROCTRACER\_API roctracer\_status\_t roctracer\_disable\_op\_activity ( activity\_domain\_t domain, uint32\_t op )

Disable activity record logging for a specified operation of a domain.

#### Parameters

in	<i>domain</i>	The domain.
in	<i>op</i>	The activity operation ID in domain.

#### Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
---	--

### 3.6.3.7 ROCTRACER\_API roctracer\_status\_t roctracer\_enable\_domain\_activity ( activity\_domain\_t domain )

Enable activity record logging for all operations of a domain using the default memory pool.

#### Parameters

in	<i>domain</i>	The domain.
----	---------------	-------------

#### Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<i>ROCTRACER_STATUS_ERROR</i>	No default pool is defined.

### 3.6.3.8 ROCTRACER\_API roctracer\_status\_t roctracer\_enable\_domain\_activity\_expl ( activity\_domain\_t domain, roctracer\_pool\_t \* pool )

Enable activity record logging for all operations of a domain providing a memory pool.

#### Parameters

in	<i>domain</i>	The domain.
in	<i>pool</i>	The memory pool to write the activity record. If NULL, use the default memory pool.

#### Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<i>ROCTRACER_STATUS_ERROR</i>	<i>pool</i> is NULL and no default pool is defined.



### 3.6.3.9 ROCTRACER\_API roctracer\_status\_t roctracer\_enable\_op\_activity ( activity\_domain\_t domain, uint32\_t op )

Enable activity record logging for a specified operation of a domain using the default memory pool.

#### Parameters

in	<i>domain</i>	The domain.
in	<i>op</i>	The activity operation ID in domain.

#### Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<i>ROCTRACER_STATUS_ERROR</i>	No default pool is defined.

### 3.6.3.10 ROCTRACER\_API roctracer\_status\_t roctracer\_enable\_op\_activity\_expl ( activity\_domain\_t domain, uint32\_t op, roctracer\_pool\_t \* pool )

Enable activity record logging for a specified operation of a domain providing a memory pool.

#### Parameters

in	<i>domain</i>	The domain.
in	<i>op</i>	The activity operation ID in domain.
in	<i>pool</i>	The memory pool to write the activity record. If NULL, use the default memory pool.

#### Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<i>ROCTRACER_STATUS_ERROR</i>	<i>pool</i> is NULL and no default pool is defined.

### 3.6.3.11 ROCTRACER\_API roctracer\_status\_t roctracer\_flush\_activity ( )

Flush available activity records for the default memory pool.

If flushing encounters an activity record still being written, flushing stops. Use a subsequent flush when the record has completed being written to resume the flush.

#### Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
---	--

### 3.6.3.12 ROCTRACER\_API roctracer\_status\_t roctracer\_flush\_activity\_expl ( roctracer\_pool\_t \* pool )

Flush available activity records for a memory pool.

If flushing encounters an activity record still being written, flushing stops. Use a subsequent flush when the record has completed being written to resume the flush.

#### Parameters

in	<i>pool</i>	The memory pool to flush. If NULL, flushes the default memory pool.
----	-------------	---

#### Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
---	--

### 3.6.3.13 ROCTRACER\_API roctracer\_status\_t roctracer\_next\_record ( const activity\_record\_t \* record, const activity\_record\_t \*\* next )

Get a pointer to the next activity record.

A memory pool generates buffers that contain multiple activity records. This function steps to the next activity record.

#### Parameters

in	<i>record</i>	Pointer to an activity record in a memory pool buffer.
out	<i>next</i>	Pointer to the following activity record in the memory pool buffer.

#### Return values

<a href="#"><i>ROCTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
---	--

### 3.6.3.14 ROCTRACER\_API roctracer\_status\_t roctracer\_open\_pool ( const roctracer\_properties\_t \* properties )

Create tracer memory pool.

Sets the default memory pool to the created pool if not already defined. Otherwise, return an error.

#### Parameters

in	<i>properties</i>	Tracer memory pool properties.
----	-------------------	--------------------------------

## Return values

<a href="#"><i>ROTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<i>ROTRACER_STATUS_ERROR_DEFAULT_POOL_ALREADY_DEFINED</i>	The default pool is already defined. Unable to create the pool.
<i>ROTRACER_STATUS_ERROR_MEMORY_ALLOCATION</i>	Unable to allocate memory for the pool. Unable to create the pool.

**3.6.3.15** `ROTRACER_API rotracer_status_t rotracer_open_pool_expl ( const rotracer_properties_t * properties, rotracer_pool_t ** pool )`

Create tracer memory pool.

If `pool` is not NULL, returns the created memory pool. Does not change the default memory pool.

If `pool` is NULL, sets the default memory pool to the created pool if not already defined. Otherwise, return an error.

## Parameters

in	<i>properties</i>	Tracer memory pool properties.
out	<i>pool</i>	Tracer memory pool created if not NULL.

## Return values

<a href="#"><i>ROTRACER_STATUS_SUCCESS</i></a>	The function has been executed successfully.
<i>ROTRACER_STATUS_ERROR_DEFAULT_POOL_ALREADY_DEFINED</i>	<code>pool</code> is NULL and the default pool is already defined. Unable to create the pool.
<i>ROTRACER_STATUS_ERROR_MEMORY_ALLOCATION</i>	Unable to allocate memory for the pool. Unable to create the pool.

## 3.7 Timestamp Operations

### Functions

- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_get\\_timestamp](#) (roctracer\_timestamp\_t \*timestamp) [ROCTRACER\\_VERSION\\_4\\_1](#)

*Get the system clock timestamp.*

### 3.7.1 Detailed Description

### 3.7.2 Function Documentation

#### 3.7.2.1 [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_get\\_timestamp](#) ( roctracer\_timestamp\_t \* *timestamp* )

Get the system clock timestamp.

#### Parameters

out	<i>timestamp</i>	The system clock timestamp in nano seconds.
-----	------------------	---

#### Return values

<a href="#">ROCTRACER_STATUS_SUCCESS</a>	The function has been executed successfully.
--	--

## 3.8 Initialization and Finalization

The ROCTracer Plugin API must be initialized before using any of the operations to report trace data, and finalized after the last trace data has been reported.

### Functions

- **ROCTRACER\_EXPORT** int `roctracer_plugin_initialize` (uint32\_t `roctracer_major_version`, uint32\_t `roctracer_↔`  
`minor_version`)  
*Initialize plugin.*
- **ROCTRACER\_EXPORT** void `roctracer_plugin_finalize` ()  
*Finalize plugin.*

### 3.8.1 Detailed Description

The ROCTracer Plugin API must be initialized before using any of the operations to report trace data, and finalized after the last trace data has been reported.

### 3.8.2 Function Documentation

#### 3.8.2.1 **ROCTRACER\_EXPORT** void `roctracer_plugin_finalize` ( )

Finalize plugin.

This must be called after `roctracer_plugin_initialize` and after all trace data has been reported by `roctracer_plugin_↔`  
`write_callback_record` and `roctracer_plugin_write_activity_records`.

#### 3.8.2.2 **ROCTRACER\_EXPORT** int `roctracer_plugin_initialize` ( uint32\_t `roctracer_major_version`, uint32\_t `roctracer_minor_version` )

Initialize plugin.

Must be called before any other operation.

#### Parameters

in	<code>roctracer_major_version</code>	The major version of the ROCTracer API being used by the ROCTracer Tool. An error is reported if this does not match the major version of the ROCTracer API used to build the plugin library. This ensures compatibility of the trace data format.
in	<code>roctracer_minor_version</code>	The minor version of the ROCTracer API being used by the ROCTracer Tool. An error is reported if the <code>roctracer_major_version</code> matches and this is greater than the minor version of the ROCTracer API used to build the plugin library. This ensures compatibility of the trace data format.

### Returns

Returns 0 on success and -1 on error.

## 3.9 Trace data reporting

Operations to output trace data.

### Functions

- **ROCTRACER\_EXPORT** int `roctracer_plugin_write_callback_record` (const `roctracer_record_t` \*record, const void \*callback\_data)  
*Report a single callback trace data.*
- **ROCTRACER\_EXPORT** int `roctracer_plugin_write_activity_records` (const `roctracer_record_t` \*begin, const `roctracer_record_t` \*end)  
*Report a range of activity trace data.*

### 3.9.1 Detailed Description

Operations to output trace data.

### 3.9.2 Function Documentation

**3.9.2.1** **ROCTRACER\_EXPORT** int `roctracer_plugin_write_activity_records` ( const `roctracer_record_t` \* *begin*, const `roctracer_record_t` \* *end* )

Report a range of activity trace data.

Reports a range of primarily domain independent trace data. The range is specified by a pointer to the first record and a pointer to one past the last record. `roctracer_next_record` is used to iterate the range in forward order.

#### Parameters

in	<i>begin</i>	Pointer to the first record.
in	<i>end</i>	Pointer to one past the last record.

#### Returns

Returns 0 on success and -1 on error.

**3.9.2.2** **ROCTRACER\_EXPORT** int `roctracer_plugin_write_callback_record` ( const `roctracer_record_t` \* *record*, const void \* *callback\_data* )

Report a single callback trace data.

**Parameters**

in	<i>record</i>	Primarily domain independent trace data.
in	<i>callback_data</i>	Domain specific trace data. The type of this argument depends on the values of <code>record.domain</code> .

**Returns**

Returns 0 on success and -1 on error.



## Chapter 4

# Data Structure Documentation

### 4.1 roctracer\_properties\_t Struct Reference

Memory pool properties.

```
#include <roctracer.h>
```

#### Data Fields

- `uint32_t mode`  
*ROC Tracer mode.*
- `size_t buffer_size`  
*Size of buffer in bytes.*
- `roctracer_allocator_t alloc_fun`  
*The allocator function to use to allocate and deallocate the buffer.*
- `void * alloc_arg`  
*The argument to pass when invoking the `alloc_fun` allocator.*
- `roctracer_buffer_callback_t buffer_callback_fun`  
*The function to call when a buffer becomes full or is flushed.*
- `void * buffer_callback_arg`  
*The argument to pass when invoking the `buffer_callback_fun` callback.*

#### 4.1.1 Detailed Description

Memory pool properties.

Defines the properties when a tracer memory pool is created.

## 4.1.2 Field Documentation

### 4.1.2.1 `void* roctracer_properties_t::alloc_arg`

The argument to pass when invoking the `alloc_fun` allocator.

### 4.1.2.2 `roctracer_allocator_t roctracer_properties_t::alloc_fun`

The allocator function to use to allocate and deallocate the buffer.

If `NULL` then `malloc`, `realloc`, and `free` are used.

### 4.1.2.3 `void* roctracer_properties_t::buffer_callback_arg`

The argument to pass when invoking the `buffer_callback_fun` callback.

### 4.1.2.4 `roctracer_buffer_callback_t roctracer_properties_t::buffer_callback_fun`

The function to call when a buffer becomes full or is flushed.

### 4.1.2.5 `size_t roctracer_properties_t::buffer_size`

Size of buffer in bytes.

### 4.1.2.6 `uint32_t roctracer_properties_t::mode`

ROC Tracer mode.

The documentation for this struct was generated from the following file:

- `/long_pathname_so_that_rpms_can_package_the_debug_info/src/roctracer/inc/roctracer.h`

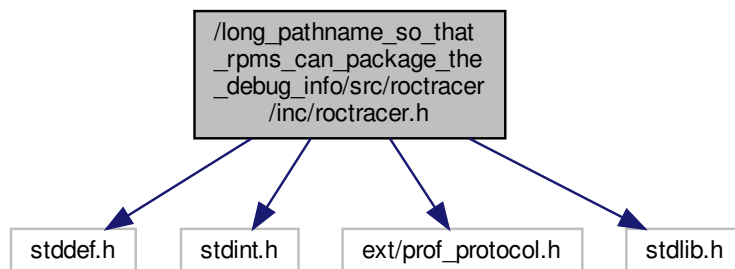
## Chapter 5

# File Documentation

### 5.1 /long\_pathname\_so\_that\_rpms\_can\_package\_the\_debug\_info/src/roctracer/inc/roctracer.h File Reference

ROCracer API interface.

```
#include <stddef.h>
#include <stdint.h>
#include "ext/prof_protocol.h"
Include dependency graph for roctracer.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [roctracer\\_properties\\_t](#)  
*Memory pool properties.*

## Macros

- #define [ROCTRACER\\_CALL](#)
- #define [ROCTRACER\\_EXPORT](#) [ROCTRACER\\_EXPORT\\_DECORATOR](#) [ROCTRACER\\_CALL](#)
- #define [ROCTRACER\\_IMPORT](#) [ROCTRACER\\_IMPORT\\_DECORATOR](#) [ROCTRACER\\_CALL](#)
- #define [ROCTRACER\\_API](#) [ROCTRACER\\_IMPORT](#)
- #define [ROCTRACER\\_VERSION\\_4\\_1](#)  
*The function was introduced in version 4.1 of the interface and has the symbol version string of "ROCTRACER\_4.1".*
- #define [ROCTRACER\\_VERSION\\_MAJOR](#) 4  
*The major version of the interface as a macro so it can be used by the preprocessor.*
- #define [ROCTRACER\\_VERSION\\_MINOR](#) 1  
*The minor version of the interface as a macro so it can be used by the preprocessor.*

## Typedefs

- typedef activity\_domain\_t [roctracer\\_domain\\_t](#)  
*Enumeration of domains that can be traced.*
- typedef activity\_rtapi\_callback\_t [roctracer\\_rtapi\\_callback\\_t](#)  
*Runtime API callback type.*
- typedef activity\_record\_t [roctracer\\_record\\_t](#)

*Activity record.*

- typedef void(\* [roctracer\\_allocator\\_t](#)) (char \*\*ptr, size\_t size, void \*arg)

*Memory pool allocator callback.*

- typedef void(\* [roctracer\\_buffer\\_callback\\_t](#)) (const char \*begin, const char \*end, void \*arg)

*Memory pool buffer callback.*

- typedef void [roctracer\\_pool\\_t](#)

*Tracer memory pool type.*

## Enumerations

- enum [roctracer\\_status\\_t](#) {  
[ROCTRACER\\_STATUS\\_SUCCESS](#) = 0, [ROCTRACER\\_STATUS\\_ERROR](#) = -1, [ROCTRACER\\_STATUS\\_ERROR\\_INVALID\\_DOMAIN\\_ID](#) = -2, [ROCTRACER\\_STATUS\\_ERROR\\_INVALID\\_ARGUMENT](#) = -3,  
[ROCTRACER\\_STATUS\\_ERROR\\_DEFAULT\\_POOL\\_UNDEFINED](#) = -4, [ROCTRACER\\_STATUS\\_ERROR\\_DEFAULT\\_POOL\\_ALREADY\\_DEFINED](#) = -5, [ROCTRACER\\_STATUS\\_ERROR\\_MEMORY\\_ALLOCATION](#) = -6,  
[ROCTRACER\\_STATUS\\_ERROR\\_MISMATCHED\\_EXTERNAL\\_CORRELATION\\_ID](#) = -7,  
[ROCTRACER\\_STATUS\\_ERROR\\_NOT\\_IMPLEMENTED](#) = -8, [ROCTRACER\\_STATUS\\_UNINIT](#) = 2, [ROCTRACER\\_STATUS\\_BREAK](#) = 3, [ROCTRACER\\_STATUS\\_BAD\\_DOMAIN](#) = [ROCTRACER\\_STATUS\\_ERROR\\_INVALID\\_DOMAIN\\_ID](#),  
[ROCTRACER\\_STATUS\\_BAD\\_PARAMETER](#) = [ROCTRACER\\_STATUS\\_ERROR\\_INVALID\\_ARGUMENT](#), [ROCTRACER\\_STATUS\\_HIP\\_API\\_ERR](#) = 6, [ROCTRACER\\_STATUS\\_HIP\\_OPS\\_ERR](#) = 7, [ROCTRACER\\_STATUS\\_HCC\\_OPS\\_ERR](#) = [ROCTRACER\\_STATUS\\_HIP\\_OPS\\_ERR](#),  
[ROCTRACER\\_STATUS\\_HSA\\_ERR](#) = 7, [ROCTRACER\\_STATUS\\_ROCTX\\_ERR](#) = 8 }

*ROC Tracer API status codes.*

## Functions

- [ROCTRACER\\_API](#) uint32\_t [roctracer\\_version\\_major](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query the major version of the installed library.*
- [ROCTRACER\\_API](#) uint32\_t [roctracer\\_version\\_minor](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query the minor version of the installed library.*
- [ROCTRACER\\_API](#) const char \* [roctracer\\_error\\_string](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query the textual description of the last error for the current thread.*
- [ROCTRACER\\_API](#) const char \* [roctracer\\_op\\_string](#) (uint32\_t domain, uint32\_t op, uint32\_t kind) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query textual name of an operation of a domain.*
- [ROCTRACER\\_API](#) [roctracer\\_status\\_t](#) [roctracer\\_op\\_code](#) (uint32\_t domain, const char \*str, uint32\_t \*op, uint32\_t \*kind) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query the operation code given a domain and the name of an operation.*
- [ROCTRACER\\_API](#) [roctracer\\_status\\_t](#) [roctracer\\_set\\_properties](#) ([roctracer\\_domain\\_t](#) domain, void \*properties) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Set the properties of a domain.*
- [ROCTRACER\\_API](#) [roctracer\\_status\\_t](#) [roctracer\\_enable\\_op\\_callback](#) ([activity\\_domain\\_t](#) domain, uint32\_t op, [activity\\_rtapi\\_callback\\_t](#) callback, void \*arg) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable runtime API callback for a specific operation of a domain.*
- [ROCTRACER\\_API](#) [roctracer\\_status\\_t](#) [roctracer\\_enable\\_domain\\_callback](#) ([activity\\_domain\\_t](#) domain, [activity\\_rtapi\\_callback\\_t](#) callback, void \*arg) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable runtime API callback for all operations of a domain.*

- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_disable\\_op\\_callback](#) (activity\_domain\_t domain, uint32\_t op) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Disable runtime API callback for a specific operation of a domain.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_disable\\_domain\\_callback](#) (activity\_domain\_t domain) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Disable runtime API callback for all operations of a domain.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_next\\_record](#) (const activity\_record\_t \*record, const activity\_record\_t \*\*next) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Get a pointer to the next activity record.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_open\\_pool\\_expl](#) (const roctracer\_properties\_t \*properties, roctracer\_pool\_t \*\*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Create tracer memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_open\\_pool](#) (const roctracer\_properties\_t \*properties) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Create tracer memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_close\\_pool\\_expl](#) (roctracer\_pool\_t \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Close tracer memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_close\\_pool](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Close default tracer memory pool, if defined, and set to undefined.*
- [ROCTRACER\\_API roctracer\\_pool\\_t \\* roctracer\\_default\\_pool\\_expl](#) (roctracer\_pool\_t \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query and set the default memory pool.*
- [ROCTRACER\\_API roctracer\\_pool\\_t \\* roctracer\\_default\\_pool](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Query the current default memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_enable\\_op\\_activity\\_expl](#) (activity\_domain\_t domain, uint32\_t op, roctracer\_pool\_t \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable activity record logging for a specified operation of a domain providing a memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_enable\\_op\\_activity](#) (activity\_domain\_t domain, uint32\_t op) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable activity record logging for a specified operation of a domain using the default memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_enable\\_domain\\_activity\\_expl](#) (activity\_domain\_t domain, roctracer\_pool\_t \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable activity record logging for all operations of a domain providing a memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_enable\\_domain\\_activity](#) (activity\_domain\_t domain) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Enable activity record logging for all operations of a domain using the default memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_disable\\_op\\_activity](#) (activity\_domain\_t domain, uint32\_t op) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Disable activity record logging for a specified operation of a domain.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_disable\\_domain\\_activity](#) (activity\_domain\_t domain) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Disable activity record logging for all operations of a domain.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_flush\\_activity\\_expl](#) (roctracer\_pool\_t \*pool) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Flush available activity records for a memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_flush\\_activity](#) () [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Flush available activity records for the default memory pool.*
- [ROCTRACER\\_API roctracer\\_status\\_t roctracer\\_get\\_timestamp](#) (roctracer\_timestamp\_t \*timestamp) [ROCTRACER\\_VERSION\\_4\\_1](#)  
*Get the system clock timestamp.*

### 5.1.1 Detailed Description

ROCracer API interface.

### 5.1.2 Macro Definition Documentation

5.1.2.1 `#define ROCTRACER_API ROCTRACER_IMPORT`

5.1.2.2 `#define ROCTRACER_CALL`

5.1.2.3 `#define ROCTRACER_EXPORT ROCTRACER_EXPORT_DECORATOR ROCTRACER_CALL`

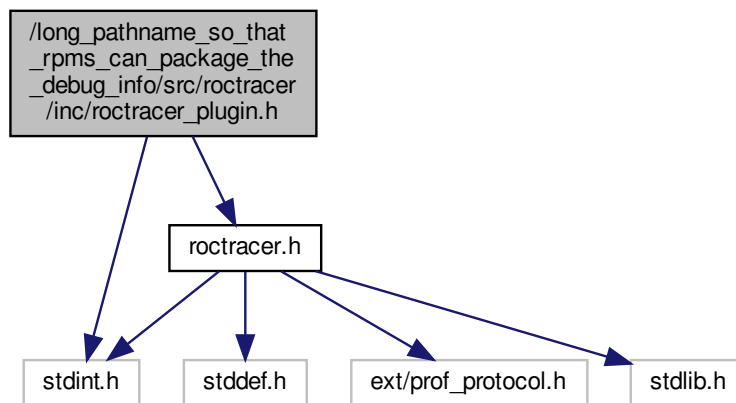
5.1.2.4 `#define ROCTRACER_IMPORT ROCTRACER_IMPORT_DECORATOR ROCTRACER_CALL`

## 5.2 /long\_pathname\_so\_that\_rpms\_can\_package\_the\_debug\_info/src/roctracer/inc/roctracer\_plugin.h File Reference

ROCracer Tool Plugin API interface.

```
#include "roctracer.h"  
#include <stdint.h>
```

Include dependency graph for roctracer\_plugin.h:



## Functions

- **ROCTRACER\_EXPORT** int `roctracer_plugin_initialize` (uint32\_t `roctracer_major_version`, uint32\_t `roctracer_↵minor_version`)  
*Initialize plugin.*
- **ROCTRACER\_EXPORT** void `roctracer_plugin_finalize` ()  
*Finalize plugin.*
- **ROCTRACER\_EXPORT** int `roctracer_plugin_write_callback_record` (const `roctracer_record_t` \*`record`, const void \*`callback_data`)  
*Report a single callback trace data.*
- **ROCTRACER\_EXPORT** int `roctracer_plugin_write_activity_records` (const `roctracer_record_t` \*`begin`, const `roctracer_record_t` \*`end`)  
*Report a range of activity trace data.*

### 5.2.1 Detailed Description

ROCracer Tool Plugin API interface.

### 5.2.2 ROCracer Plugin API

The ROCracer Plugin API is used by the ROCracer Tool to output all tracing information. Different implementations of the ROCracer Plugin API can be developed that output the tracing data in different formats. The ROCracer Tool can be configured to load a specific library that supports the user desired format.

The API is not thread safe. It is the responsibility of the ROCracer Tool to ensure the operations are synchronized and not called concurrently. There is no requirement for the ROCracer Tool to report trace data in any specific order. If the format supported by plugin requires specific ordering, it is the responsibility of the plugin implementation to perform any necessary sorting.



# Index

/long\_pathname\_so\_that\_rpms\_can\_package\_the\_↔  
debug\_info/src/roctracer/inc/roctracer.h, [29](#)  
/long\_pathname\_so\_that\_rpms\_can\_package\_the\_↔  
debug\_info/src/roctracer/inc/roctracer\_plugin.h,  
[33](#)

## Activity API, [14](#)

- roctracer\_allocator\_t, [15](#)
- roctracer\_buffer\_callback\_t, [15](#)
- roctracer\_close\_pool, [16](#)
- roctracer\_close\_pool\_expl, [16](#)
- roctracer\_default\_pool, [17](#)
- roctracer\_default\_pool\_expl, [17](#)
- roctracer\_disable\_domain\_activity, [17](#)
- roctracer\_disable\_op\_activity, [18](#)
- roctracer\_enable\_domain\_activity, [18](#)
- roctracer\_enable\_domain\_activity\_expl, [18](#)
- roctracer\_enable\_op\_activity, [18](#)
- roctracer\_enable\_op\_activity\_expl, [19](#)
- roctracer\_flush\_activity, [19](#)
- roctracer\_flush\_activity\_expl, [19](#)
- roctracer\_next\_record, [20](#)
- roctracer\_open\_pool, [20](#)
- roctracer\_open\_pool\_expl, [21](#)
- roctracer\_pool\_t, [16](#)
- roctracer\_record\_t, [16](#)

## alloc\_arg

- roctracer\_properties\_t, [28](#)

## alloc\_fun

- roctracer\_properties\_t, [28](#)

## buffer\_callback\_arg

- roctracer\_properties\_t, [28](#)

## buffer\_callback\_fun

- roctracer\_properties\_t, [28](#)

## buffer\_size

- roctracer\_properties\_t, [28](#)

## Callback API, [11](#)

- roctracer\_disable\_domain\_callback, [11](#)
- roctracer\_disable\_op\_callback, [12](#)
- roctracer\_enable\_domain\_callback, [12](#)
- roctracer\_enable\_op\_callback, [12](#)
- roctracer\_rtapi\_callback\_t, [11](#)

## Initialization and Finalization, [23](#)

- roctracer\_plugin\_finalize, [23](#)
- roctracer\_plugin\_initialize, [23](#)

## mode

- roctracer\_properties\_t, [28](#)

## ROCTRACER\_API

- roctracer.h, [33](#)

## ROCTRACER\_CALL

- roctracer.h, [33](#)

## ROCTRACER\_EXPORT

- roctracer.h, [33](#)

## ROCTRACER\_IMPORT

- roctracer.h, [33](#)

## ROCTRACER\_STATUS\_BAD\_DOMAIN

- Status Codes, [8](#)

## ROCTRACER\_STATUS\_BAD\_PARAMETER

- Status Codes, [8](#)

## ROCTRACER\_STATUS\_BREAK

- Status Codes, [8](#)

## ROCTRACER\_STATUS\_ERROR\_DEFAULT\_POOL\_↔ ALREADY\_DEFINED

- Status Codes, [7](#)

## ROCTRACER\_STATUS\_ERROR\_DEFAULT\_POOL\_↔ UNDEFINED

- Status Codes, [7](#)

## ROCTRACER\_STATUS\_ERROR\_INVALID\_ARGUME↔ NT

- Status Codes, [7](#)

## ROCTRACER\_STATUS\_ERROR\_INVALID\_DOMAIN\_ID

- Status Codes, [7](#)

## ROCTRACER\_STATUS\_ERROR\_MEMORY\_ALLOCA↔ TION

- Status Codes, [8](#)

## ROCTRACER\_STATUS\_ERROR\_MISMATCHED\_EX↔ TERNAL\_CORRELATION\_ID

- Status Codes, [8](#)

## ROCTRACER\_STATUS\_ERROR\_NOT\_IMPLEMENTED

- Status Codes, [8](#)

## ROCTRACER\_STATUS\_ERROR

- Status Codes, [7](#)

## ROCTRACER\_STATUS\_HCC\_OPS\_ERR

- Status Codes, [8](#)

## ROCTRACER\_STATUS\_HIP\_API\_ERR

- Status Codes, [8](#)

## ROCTRACER\_STATUS\_HIP\_OPS\_ERR

- Status Codes, [8](#)
- ROCTRACER\_STATUS\_HSA\_ERR
  - Status Codes, [8](#)
- ROCTRACER\_STATUS\_ROCTX\_ERR
  - Status Codes, [8](#)
- ROCTRACER\_STATUS\_SUCCESS
  - Status Codes, [7](#)
- ROCTRACER\_STATUS\_UNINIT
  - Status Codes, [8](#)
- ROCTRACER\_VERSION\_4\_1
  - Symbol Versions, [5](#)
- ROCTRACER\_VERSION\_MAJOR
  - Versioning, [6](#)
- ROCTRACER\_VERSION\_MINOR
  - Versioning, [6](#)
- roctracer.h
  - ROCTRACER\_API, [33](#)
  - ROCTRACER\_CALL, [33](#)
  - ROCTRACER\_EXPORT, [33](#)
  - ROCTRACER\_IMPORT, [33](#)
- roctracer\_allocator\_t
  - Activity API, [15](#)
- roctracer\_buffer\_callback\_t
  - Activity API, [15](#)
- roctracer\_close\_pool
  - Activity API, [16](#)
- roctracer\_close\_pool\_expl
  - Activity API, [16](#)
- roctracer\_default\_pool
  - Activity API, [17](#)
- roctracer\_default\_pool\_expl
  - Activity API, [17](#)
- roctracer\_disable\_domain\_activity
  - Activity API, [17](#)
- roctracer\_disable\_domain\_callback
  - Callback API, [11](#)
- roctracer\_disable\_op\_activity
  - Activity API, [18](#)
- roctracer\_disable\_op\_callback
  - Callback API, [12](#)
- roctracer\_domain\_t
  - Traced Runtime Domains, [9](#)
- roctracer\_enable\_domain\_activity
  - Activity API, [18](#)
- roctracer\_enable\_domain\_activity\_expl
  - Activity API, [18](#)
- roctracer\_enable\_domain\_callback
  - Callback API, [12](#)
- roctracer\_enable\_op\_activity
  - Activity API, [18](#)
- roctracer\_enable\_op\_activity\_expl
  - Activity API, [19](#)
- roctracer\_enable\_op\_callback
  - Callback API, [12](#)
- roctracer\_error\_string
  - Status Codes, [8](#)
- roctracer\_flush\_activity
  - Activity API, [19](#)
- roctracer\_flush\_activity\_expl
  - Activity API, [19](#)
- roctracer\_get\_timestamp
  - Timestamp Operations, [22](#)
- roctracer\_next\_record
  - Activity API, [20](#)
- roctracer\_op\_code
  - Traced Runtime Domains, [9](#)
- roctracer\_op\_string
  - Traced Runtime Domains, [10](#)
- roctracer\_open\_pool
  - Activity API, [20](#)
- roctracer\_open\_pool\_expl
  - Activity API, [21](#)
- roctracer\_plugin\_finalize
  - Initialization and Finalization, [23](#)
- roctracer\_plugin\_initialize
  - Initialization and Finalization, [23](#)
- roctracer\_plugin\_write\_activity\_records
  - Trace data reporting, [25](#)
- roctracer\_plugin\_write\_callback\_record
  - Trace data reporting, [25](#)
- roctracer\_pool\_t
  - Activity API, [16](#)
- roctracer\_properties\_t, [27](#)
  - alloc\_arg, [28](#)
  - alloc\_fun, [28](#)
  - buffer\_callback\_arg, [28](#)
  - buffer\_callback\_fun, [28](#)
  - buffer\_size, [28](#)
  - mode, [28](#)
- roctracer\_record\_t
  - Activity API, [16](#)
- roctracer\_rtapi\_callback\_t
  - Callback API, [11](#)
- roctracer\_set\_properties
  - Traced Runtime Domains, [10](#)
- roctracer\_status\_t
  - Status Codes, [7](#)
- roctracer\_version\_major
  - Versioning, [6](#)
- roctracer\_version\_minor
  - Versioning, [6](#)
- Status Codes, [7](#)
  - ROCTRACER\_STATUS\_BAD\_DOMAIN, [8](#)
  - ROCTRACER\_STATUS\_BAD\_PARAMETER, [8](#)
  - ROCTRACER\_STATUS\_BREAK, [8](#)
  - ROCTRACER\_STATUS\_ERROR\_DEFAULT\_PO↔OL\_ALREADY\_DEFINED, [7](#)

- ROCTRACER\_STATUS\_ERROR\_DEFAULT\_PO↔  
OL\_UNDEFINED, [7](#)
- ROCTRACER\_STATUS\_ERROR\_INVALID\_ARG↔  
UMENT, [7](#)
- ROCTRACER\_STATUS\_ERROR\_INVALID\_DOM↔  
AIN\_ID, [7](#)
- ROCTRACER\_STATUS\_ERROR\_MEMORY\_ALL↔  
OCATION, [8](#)
- ROCTRACER\_STATUS\_ERROR\_MISMATCHED↔  
\_EXTERNAL\_CORRELATION\_ID, [8](#)
- ROCTRACER\_STATUS\_ERROR\_NOT\_IMPLM↔  
ENTED, [8](#)
- ROCTRACER\_STATUS\_ERROR, [7](#)
- ROCTRACER\_STATUS\_HCC\_OPS\_ERR, [8](#)
- ROCTRACER\_STATUS\_HIP\_API\_ERR, [8](#)
- ROCTRACER\_STATUS\_HIP\_OPS\_ERR, [8](#)
- ROCTRACER\_STATUS\_HSA\_ERR, [8](#)
- ROCTRACER\_STATUS\_ROCTX\_ERR, [8](#)
- ROCTRACER\_STATUS\_SUCCESS, [7](#)
- ROCTRACER\_STATUS\_UNINIT, [8](#)
- roctracer\_error\_string, [8](#)
- roctracer\_status\_t, [7](#)
- Symbol Versions, [5](#)
  - ROCTRACER\_VERSION\_4\_1, [5](#)
- Timestamp Operations, [22](#)
  - roctracer\_get\_timestamp, [22](#)
- Trace data reporting, [25](#)
  - roctracer\_plugin\_write\_activity\_records, [25](#)
  - roctracer\_plugin\_write\_callback\_record, [25](#)
- Traced Runtime Domains, [9](#)
  - roctracer\_domain\_t, [9](#)
  - roctracer\_op\_code, [9](#)
  - roctracer\_op\_string, [10](#)
  - roctracer\_set\_properties, [10](#)
- Versioning, [6](#)
  - ROCTRACER\_VERSION\_MAJOR, [6](#)
  - ROCTRACER\_VERSION\_MINOR, [6](#)
  - roctracer\_version\_major, [6](#)
  - roctracer\_version\_minor, [6](#)