

roctracer

4.1.0

Generated by Doxygen 1.8.18

Wed Apr 19 2023 00:20:50

| | |
|---|----------|
| 1 ROC Tracer API Specification | 1 |
| 1.1 Introduction | 1 |
| 1.2 Known Limitations and Restrictions | 1 |
| 2 Todo List | 3 |
| 3 Module Documentation | 5 |
| 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 | 7 |
| 3.2.3.1 roctracer_version_major() | 7 |
| 3.2.3.2 roctracer_version_minor() | 7 |
| 3.3 Status Codes | 8 |
| 3.3.1 Detailed Description | 8 |
| 3.3.2 Enumeration Type Documentation | 8 |
| 3.3.2.1 roctracer_status_t | 8 |
| 3.3.3 Function Documentation | 9 |
| 3.3.3.1 roctracer_error_string() | 9 |
| 3.4 Traced Runtime Domains | 10 |
| 3.4.1 Detailed Description | 10 |
| 3.4.2 Typedef Documentation | 10 |
| 3.4.2.1 roctracer_domain_t | 10 |
| 3.4.3 Function Documentation | 10 |
| 3.4.3.1 roctracer_op_code() | 10 |
| 3.4.3.2 roctracer_op_string() | 11 |
| 3.4.3.3 roctracer_set_properties() | 11 |
| 3.5 Callback API | 13 |
| 3.5.1 Detailed Description | 13 |
| 3.5.2 Typedef Documentation | 13 |
| 3.5.2.1 roctracer_rtapi_callback_t | 13 |
| 3.5.3 Function Documentation | 13 |
| 3.5.3.1 roctracer_disable_domain_callback() | 13 |
| 3.5.3.2 roctracer_disable_op_callback() | 14 |

| | |
|---|----|
| 3.5.3.3 roctracer_enable_domain_callback() | 14 |
| 3.5.3.4 roctracer_enable_op_callback() | 15 |
| 3.6 Activity API | 16 |
| 3.6.1 Detailed Description | 17 |
| 3.6.2 Typedef Documentation | 17 |
| 3.6.2.1 roctracer_allocator_t | 17 |
| 3.6.2.2 roctracer_buffer_callback_t | 18 |
| 3.6.2.3 roctracer_pool_t | 18 |
| 3.6.2.4 roctracer_record_t | 18 |
| 3.6.3 Function Documentation | 18 |
| 3.6.3.1 roctracer_close_pool() | 18 |
| 3.6.3.2 roctracer_close_pool_expl() | 19 |
| 3.6.3.3 roctracer_default_pool() | 19 |
| 3.6.3.4 roctracer_default_pool_expl() | 19 |
| 3.6.3.5 roctracer_disable_domain_activity() | 20 |
| 3.6.3.6 roctracer_disable_op_activity() | 20 |
| 3.6.3.7 roctracer_enable_domain_activity() | 21 |
| 3.6.3.8 roctracer_enable_domain_activity_expl() | 21 |
| 3.6.3.9 roctracer_enable_op_activity() | 21 |
| 3.6.3.10 roctracer_enable_op_activity_expl() | 22 |
| 3.6.3.11 roctracer_flush_activity() | 22 |
| 3.6.3.12 roctracer_flush_activity_expl() | 23 |
| 3.6.3.13 roctracer_next_record() | 23 |
| 3.6.3.14 roctracer_open_pool() | 24 |
| 3.6.3.15 roctracer_open_pool_expl() | 24 |
| 3.7 Timestamp Operations | 26 |
| 3.7.1 Detailed Description | 26 |
| 3.7.2 Function Documentation | 26 |
| 3.7.2.1 roctracer_get_timestamp() | 26 |
| 3.8 Initialization and Finalization | 27 |
| 3.8.1 Detailed Description | 27 |
| 3.8.2 Function Documentation | 27 |
| 3.8.2.1 roctracer_plugin_finalize() | 27 |
| 3.8.2.2 roctracer_plugin_initialize() | 27 |
| 3.9 Trace data reporting | 29 |
| 3.9.1 Detailed Description | 29 |
| 3.9.2 Function Documentation | 29 |
| 3.9.2.1 roctracer_plugin_write_activity_records() | 29 |
| 3.9.2.2 roctracer_plugin_write_callback_record() | 30 |

| | |
|--|-----------|
| 4 Data Structure Documentation | 31 |
| 4.1 roctracer_properties_t Struct Reference | 31 |
| 4.1.1 Detailed Description | 31 |
| 4.1.2 Field Documentation | 31 |
| 4.1.2.1 alloc_arg | 32 |
| 4.1.2.2 alloc_fun | 32 |
| 4.1.2.3 buffer_callback_arg | 32 |
| 4.1.2.4 buffer_callback_fun | 32 |
| 4.1.2.5 buffer_size | 32 |
| 4.1.2.6 mode | 32 |
| 5 File Documentation | 33 |
| 5.1 /long_pathname_so_that_rpms_can_package_the_debug_info/src/roctracer/inc/roctracer.h File Reference | 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 |
| Index | 35 |

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 ROCTRACER_VERSION_4_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 ROCTRACER_VERSION_MAJOR

```
#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 ROCTRACER_VERSION_MINOR

```
#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_version_major()

```
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_version_minor()

```
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_EXISTS` = -5, `ROCTRACER_STATUS_ERROR_MEMORY_ALLOCATION` = -6, `ROCTRACER_STATUS_ERROR_MISMATCHED_EXTENSION` = -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 `roctracer_status_t`

enum `roctracer_status_t`

ROC Tracer API status codes.

Enumerator

| | |
|---------------------------------------|---|
| <code>ROCTRACER_STATUS_SUCCESS</code> | The function has executed successfully. |
| <code>ROCTRACER_STATUS_ERROR</code> | A generic error has occurred. |

Enumerator

| | |
|---|--|
| 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_error_string()

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

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 roctracer_domain_t

```
typedef activity_domain_t roctracer\_domain\_t
```

Enumeration of domains that can be traced.

3.4.3 Function Documentation

3.4.3.1 roctracer_op_code()

```
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

| | |
|---|---|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. <i>op</i> and <i>kind</i> have been updated. |
| <i>ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT</i> | The <i>op</i> is invalid for <i>domain</i> . |
| <i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i> | The domain is invalid or not supported. |

3.4.3.2 roctracer_op_string()

```
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> | |

Todo Define kind.

Returns

Returns the NUL terminated string for the operation name, or NULL if the domain or operation are invalid. The string is owned by the ROC Tracer library.

3.4.3.3 roctracer_set_properties()

```
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

| | |
|---|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
|---|--|

3.5 Callback API

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.

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 roctracer_rtapi_callback_t

```
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_disable_domain_callback()

```
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

| | |
|---|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
| <i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i> | domain is invalid. |

3.5.3.2 roctracer_disable_op_callback()

```
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 domain. |

Return values

| | |
|---|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
| <i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i> | domain is invalid. |
| <i>ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT</i> | op is invalid for domain. |

3.5.3.3 roctracer_enable_domain_callback()

```
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 callback. |

Return values

| | |
|---|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
| <i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i> | domain is invalid. |

3.5.3.4 roctracer_enable_op_callback()

```
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 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

| | |
|---|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
| <i>ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID</i> | domain is invalid. |
| <i>ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT</i> | op is invalid for domain. |

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

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 roctracer_allocator_t

```
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` `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 roctracer_buffer_callback_t

```
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 roctracer_pool_t

```
typedef void roctracer_pool_t
```

Tracer memory pool type.

3.6.2.4 roctracer_record_t

```
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_close_pool()

```
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_close_pool_expl()

```
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

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

3.6.3.3 roctracer_default_pool()

```
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_default_pool_expl()

```
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 <code>pool</code> . 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_disable_domain_activity()`

```
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

| | |
|---|--|
| <code>ROCTRACER_STATUS_SUCCESS</code> | The function has been executed successfully. |
|---|--|

3.6.3.6 `roctracer_disable_op_activity()`

```
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

| | |
|---|--|
| <code>ROCTRACER_STATUS_SUCCESS</code> | The function has been executed successfully. |
|---|--|

3.6.3.7 roctracer_enable_domain_activity()

```
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

| | |
|---|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
| <i>ROCTRACER_STATUS_ERROR</i> | No default pool is defined. |

3.6.3.8 roctracer_enable_domain_activity_expl()

```
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

| | |
|---|---|
| <i>ROCTRACER_STATUS_SUCCESS</i> | 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_enable_op_activity()

```
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

| | |
|---|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
| <i>ROCTRACER_STATUS_ERROR</i> | No default pool is defined. |

3.6.3.10 roctracer_enable_op_activity_expl()

```
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

| | |
|---|---|
| <i>ROCTRACER_STATUS_SUCCESS</i> | 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_flush_activity()

```
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

| | |
|---------------------------------------|--|
| <code>ROCTRACER_STATUS_SUCCESS</code> | The function has been executed successfully. |
|---------------------------------------|--|

3.6.3.12 roctracer_flush_activity_expl()

```
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

| | |
|---------------------------------------|--|
| <code>ROCTRACER_STATUS_SUCCESS</code> | The function has been executed successfully. |
|---------------------------------------|--|

3.6.3.13 roctracer_next_record()

```
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

| | |
|---------------------------------------|--|
| <code>ROCTRACER_STATUS_SUCCESS</code> | The function has been executed successfully. |
|---------------------------------------|--|

3.6.3.14 roctracer_open_pool()

```
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

| | |
|--|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
| <i>ROCTRACER_STATUS_ERROR_DEFAULT_POOL_ALREADY_DEFINED</i> | The default pool is already defined. Unable to create the pool. |
| <i>ROCTRACER_STATUS_ERROR_MEMORY_ALLOCATION</i> | Unable to allocate memory for the pool. Unable to create the pool. |

3.6.3.15 roctracer_open_pool_expl()

```
ROCTRACER_API roctracer_status_t roctracer_open_pool_expl (
    const roctracer_properties_t * properties,
    roctracer_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

| | |
|---|--|
| <i>ROCTRACER_STATUS_SUCCESS</i> | The function has been executed successfully. |
|---|--|

Return values

| | |
|--|--|
| <i>ROCTRACER_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>ROCTRACER_STATUS_ERROR_MEMORY_ALLO↔ CATION</i> | Unable to allocate memory for the <code>pool</code> . 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_0](#)

Get the system clock timestamp.

3.7.1 Detailed Description

3.7.2 Function Documentation

3.7.2.1 roctracer_get_timestamp()

```
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

| | |
|--|--|
| ROCTRACER_STATUS_SUCCESS | 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_plugin_finalize()`

```
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_plugin_initialize()`

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

```
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

| | | |
|-----------------|--------------------|--------------------------------------|
| <code>in</code> | <code>begin</code> | Pointer to the first record. |
| <code>in</code> | <code>end</code> | Pointer to one past the last record. |

Returns

Returns 0 on success and -1 on error.

3.9.2.2 roctracer_plugin_write_callback_record()

```
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 alloc_arg

```
void* roctracer_properties_t::alloc_arg
```

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

4.1.2.2 alloc_fun

```
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 buffer_callback_arg

```
void* roctracer_properties_t::buffer_callback_arg
```

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

4.1.2.4 buffer_callback_fun

```
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 buffer_size

```
size_t roctracer_properties_t::buffer_size
```

Size of buffer in bytes.

4.1.2.6 mode

```
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

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

5.2 /long_pathname_so_that_rpms_can_package_the_debug_↵ info/src/roctracer/inc/roctracer_plugin.h File Reference

```
#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

5.2.2 ROTracer Plugin API

The ROTracer Plugin API is used by the ROTracer Tool to output all tracing information. Different implementations of the ROTracer Plugin API can be developed that output the tracing data in different formats. The ROTracer 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 ROTracer Tool to ensure the operations are synchronized and not called concurrently. There is no requirement for the ROTracer 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. ROTracer Tool Plugin API interface.

Index

- [/long_pathname_so_that_rpms_can_package_the_debug_info/src/roctracer/plugin/roctracer.h, 27](#)
- [33](#)
- [/long_pathname_so_that_rpms_can_package_the_debug_info/src/roctracer/inc/roctracer_plugin.h, 32](#)
- [33](#)
- [roctracer_allocator_t, 17](#)
- [roctracer_buffer_callback_t, 17](#)
- [roctracer_close_pool, 18](#)
- [roctracer_close_pool_expl, 19](#)
- [roctracer_default_pool, 19](#)
- [roctracer_default_pool_expl, 19](#)
- [roctracer_disable_domain_activity, 20](#)
- [roctracer_disable_op_activity, 20](#)
- [roctracer_enable_domain_activity, 21](#)
- [roctracer_enable_domain_activity_expl, 21](#)
- [roctracer_enable_op_activity, 21](#)
- [roctracer_enable_op_activity_expl, 22](#)
- [roctracer_flush_activity, 22](#)
- [roctracer_flush_activity_expl, 23](#)
- [roctracer_next_record, 23](#)
- [roctracer_open_pool, 24](#)
- [roctracer_open_pool_expl, 24](#)
- [roctracer_pool_t, 18](#)
- [roctracer_record_t, 18](#)
- [alloc_arg](#)
- [roctracer_properties_t, 31](#)
- [alloc_fun](#)
- [roctracer_properties_t, 32](#)
- [buffer_callback_arg](#)
- [roctracer_properties_t, 32](#)
- [buffer_callback_fun](#)
- [roctracer_properties_t, 32](#)
- [buffer_size](#)
- [roctracer_properties_t, 32](#)
- [Callback API, 13](#)
- [roctracer_disable_domain_callback, 13](#)
- [roctracer_disable_op_callback, 14](#)
- [roctracer_enable_domain_callback, 14](#)
- [roctracer_enable_op_callback, 15](#)
- [roctracer_rtapi_callback_t, 13](#)
- [Initialization and Finalization, 27](#)
- [roctracer_plugin_finalize, 27](#)
- [roctracer_allocator_t](#)
- [Activity API, 17](#)
- [roctracer_buffer_callback_t](#)
- [Activity API, 17](#)
- [roctracer_close_pool](#)
- [Activity API, 18](#)
- [roctracer_close_pool_expl](#)
- [Activity API, 19](#)
- [roctracer_default_pool](#)
- [Activity API, 19](#)
- [roctracer_default_pool_expl](#)
- [Activity API, 19](#)
- [roctracer_disable_domain_activity](#)
- [Activity API, 20](#)
- [roctracer_disable_domain_callback](#)
- [Callback API, 13](#)
- [roctracer_disable_op_activity](#)
- [Activity API, 20](#)
- [roctracer_disable_op_callback](#)
- [Callback API, 14](#)
- [roctracer_domain_t](#)
- [Traced Runtime Domains, 10](#)
- [roctracer_enable_domain_activity](#)
- [Activity API, 21](#)
- [roctracer_enable_domain_activity_expl](#)
- [Activity API, 21](#)
- [roctracer_enable_domain_callback](#)
- [Callback API, 14](#)
- [roctracer_enable_op_activity](#)
- [Activity API, 21](#)
- [roctracer_enable_op_activity_expl](#)
- [Activity API, 22](#)
- [roctracer_enable_op_callback](#)
- [Callback API, 15](#)
- [roctracer_error_string](#)
- [Status Codes, 9](#)
- [roctracer_flush_activity](#)
- [Activity API, 22](#)
- [roctracer_flush_activity_expl](#)
- [Activity API, 23](#)
- [roctracer_get_timestamp](#)

- Timestamp Operations, [26](#)
- roctracer_next_record
 - Activity API, [23](#)
- roctracer_op_code
 - Traced Runtime Domains, [10](#)
- roctracer_op_string
 - Traced Runtime Domains, [11](#)
- roctracer_open_pool
 - Activity API, [24](#)
- roctracer_open_pool_expl
 - Activity API, [24](#)
- roctracer_plugin_finalize
 - Initialization and Finalization, [27](#)
- roctracer_plugin_initialize
 - Initialization and Finalization, [27](#)
- roctracer_plugin_write_activity_records
 - Trace data reporting, [29](#)
- roctracer_plugin_write_callback_record
 - Trace data reporting, [29](#)
- roctracer_pool_t
 - Activity API, [18](#)
- roctracer_properties_t, [31](#)
 - alloc_arg, [31](#)
 - alloc_fun, [32](#)
 - buffer_callback_arg, [32](#)
 - buffer_callback_fun, [32](#)
 - buffer_size, [32](#)
 - mode, [32](#)
- roctracer_record_t
 - Activity API, [18](#)
- roctracer_rtapi_callback_t
 - Callback API, [13](#)
- roctracer_set_properties
 - Traced Runtime Domains, [11](#)
- ROCTRACER_STATUS_BAD_DOMAIN
 - Status Codes, [9](#)
- ROCTRACER_STATUS_BAD_PARAMETER
 - Status Codes, [9](#)
- ROCTRACER_STATUS_BREAK
 - Status Codes, [9](#)
- ROCTRACER_STATUS_ERROR
 - Status Codes, [8](#)
- ROCTRACER_STATUS_ERROR_DEFAULT_POOL_ALREADY_DEFINED
 - Status Codes, [9](#)
- ROCTRACER_STATUS_ERROR_DEFAULT_POOL_UNDEFINED
 - Status Codes, [9](#)
- ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT
 - Status Codes, [9](#)
- ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID
 - Status Codes, [9](#)
- ROCTRACER_STATUS_ERROR_MEMORY_ALLOCATION
 - Status Codes, [9](#)
- ROCTRACER_STATUS_ERROR_MISMATCHED_EXTERNAL_CONFIGURATION
 - Status Codes, [9](#)

- ROCTRACER_STATUS_ERROR_NOT_IMPLEMENTED
 - Status Codes, [9](#)
- ROCTRACER_STATUS_HCC_OPS_ERR
 - Status Codes, [9](#)
- ROCTRACER_STATUS_HIP_API_ERR
 - Status Codes, [9](#)
- ROCTRACER_STATUS_HIP_OPS_ERR
 - Status Codes, [9](#)
- ROCTRACER_STATUS_HSA_ERR
 - Status Codes, [9](#)
- ROCTRACER_STATUS_ROCTX_ERR
 - Status Codes, [9](#)
- ROCTRACER_STATUS_SUCCESS
 - Status Codes, [8](#)
- roctracer_status_t
 - Status Codes, [8](#)
- ROCTRACER_STATUS_UNINIT
 - Status Codes, [9](#)
- ROCTRACER_VERSION_4_1
 - Symbol Versions, [5](#)
- ROCTRACER_VERSION_MAJOR
 - Versioning, [6](#)
- roctracer_version_major
 - Versioning, [7](#)
- ROCTRACER_VERSION_MINOR
 - Versioning, [6](#)
- roctracer_version_minor
 - Versioning, [7](#)
- Status Codes, [8](#)
 - roctracer_error_string, [9](#)
 - ROCTRACER_STATUS_BAD_DOMAIN, [9](#)
 - ROCTRACER_STATUS_BAD_PARAMETER, [9](#)
 - ROCTRACER_STATUS_BREAK, [9](#)
 - ROCTRACER_STATUS_ERROR, [8](#)
 - ROCTRACER_STATUS_ERROR_DEFAULT_POOL_ALREADY_DEFINED, [9](#)
 - ROCTRACER_STATUS_ERROR_DEFAULT_POOL_UNDEFINED, [9](#)
 - ROCTRACER_STATUS_ERROR_INVALID_ARGUMENT, [9](#)
 - ROCTRACER_STATUS_ERROR_INVALID_DOMAIN_ID, [9](#)
 - ROCTRACER_STATUS_ERROR_MEMORY_ALLOCATION, [9](#)
 - ROCTRACER_STATUS_ERROR_MISMATCHED_EXTERNAL_CONFIGURATION, [9](#)
 - ROCTRACER_STATUS_ERROR_NOT_IMPLEMENTED, [9](#)
 - ROCTRACER_STATUS_HCC_OPS_ERR, [9](#)
 - ROCTRACER_STATUS_HIP_API_ERR, [9](#)
 - ROCTRACER_STATUS_HIP_OPS_ERR, [9](#)
 - ROCTRACER_STATUS_HSA_ERR, [9](#)
 - ROCTRACER_STATUS_ROCTX_ERR, [9](#)

- ROCTRACER_STATUS_SUCCESS, [8](#)
- roctracer_status_t, [8](#)
- ROCTRACER_STATUS_UNINIT, [9](#)
- Symbol Versions, [5](#)
 - ROCTRACER_VERSION_4_1, [5](#)
- Timestamp Operations, [26](#)
 - roctracer_get_timestamp, [26](#)
- Trace data reporting, [29](#)
 - roctracer_plugin_write_activity_records, [29](#)
 - roctracer_plugin_write_callback_record, [29](#)
- Traced Runtime Domains, [10](#)
 - roctracer_domain_t, [10](#)
 - roctracer_op_code, [10](#)
 - roctracer_op_string, [11](#)
 - roctracer_set_properties, [11](#)
- Versioning, [6](#)
 - ROCTRACER_VERSION_MAJOR, [6](#)
 - roctracer_version_major, [7](#)
 - ROCTRACER_VERSION_MINOR, [6](#)
 - roctracer_version_minor, [7](#)