OPEN TRACE FORMAT 2

USER MANUAL

1.5.1 (revision 4026)

Tue Dec 9 2014 16:23:25

OTF2 License Agreement

Contents

Chapter 1

Open Trace Format 2

1.1 Introduction

The OTF2 library provides an interface to write and read trace data.

OTF2 is developed within the Score-P project. The Score-P project is funded by the German Federal Ministry of Education and Research. OTF2 is available under the BSD open source license that allows free usage for academic and commercial applications.

1.2 Get started

OTF2 usage examples

OTF2 records

OTF2 callbacks

Usage of OTF2 tools

Appendices

Appendix A

OTF2 INSTALL

```
For generic installation instructions see below.
When building for an Intel MIC platform, carefully follow the
platform-specific instructions below.

Configuration of OTF2
*********************

‘configure’ configures OTF2 to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help              display this help and exit
      --help=short        display options specific to this package
      --help=recursive    display the short help of all the included packages
  -V, --version           display version information and exit
  -q, --quiet, --silent   do not print ‘checking ...’ messages
      --cache-file=FILE   cache test results in FILE [disabled]
  -C, --config-cache      alias for ‘--cache-file=config.cache’
  -n, --no-create         do not create output files
      --srcdir=DIR        find the sources in DIR [configure dir or ‘..’]

Installation directories:
  --prefix=PREFIX         install architecture-independent files in PREFIX
                          [/opt/otf2]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
                          [PREFIX]

By default, ‘make install’ will install all the files in
‘/opt/otf2/bin’, ‘/opt/otf2/lib’ etc.  You can specify
an installation prefix other than ‘/opt/otf2’ using ‘--prefix’,
for instance `--prefix=$HOME'.

For better control, use the options below.

Fine tuning of the installation directories:
  --bindir=DIR            user executables [EPREFIX/bin]
  --sbindir=DIR           system admin executables [EPREFIX/sbin]
  --libexecdir=DIR        program executables [EPREFIX/libexec]
  --sysconfdir=DIR        read-only single-machine data [PREFIX/etc]
  --sharedstatedir=DIR    modifiable architecture-independent data [PREFIX/com]
  --localstatedir=DIR     modifiable single-machine data [PREFIX/var]
  --libdir=DIR            object code libraries [EPREFIX/lib]
  --includedir=DIR        C header files [PREFIX/include]
  --oldincludedir=DIR     C header files for non-gcc [/usr/include]
  --datarootdir=DIR       read-only arch.-independent data root [PREFIX/share]
  --datadir=DIR           read-only architecture-independent data [DATAROOTDIR]
  --infodir=DIR           info documentation [DATAROOTDIR/info]
  --localedir=DIR         locale-dependent data [DATAROOTDIR/locale]
  --mandir=DIR            man documentation [DATAROOTDIR/man]
  --docdir=DIR            documentation root [DATAROOTDIR/doc/otf2]
  --htmldir=DIR           html documentation [DOCDIR]
  --dvidir=DIR            dvi documentation [DOCDIR]
  --pdfdir=DIR            pdf documentation [DOCDIR]
  --psdir=DIR             ps documentation [DOCDIR]

Program names:
  --program-prefix=PREFIX            prepend PREFIX to installed program names
  --program-suffix=SUFFIX            append SUFFIX to installed program names
  --program-transform-name=PROGRAM   run sed PROGRAM on installed program names

System types:
  --build=BUILD     configure for building on BUILD [guessed]
  --host=HOST       cross-compile to build programs to run on HOST [BUILD]

Optional Features:
  --disable-option-checking  ignore unrecognized --enable/--with options
  --disable-FEATURE       do not include FEATURE (same as --enable-FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --enable-silent-rules          less verbose build output (undo: `make V=1')
  --disable-silent-rules         verbose build output (undo: `make V=0')
  --disable-dependency-tracking  speeds up one-time build
  --enable-dependency-tracking   do not reject slow dependency extractors
  --enable-platform-mic   Force build for Intel MIC platform [no]
  --enable-debug          activate internal debug output [no]
  --enable-backend-test-runs
                          Run tests at make check [no]. If disabled, tests are
                          still build at make check. Additionally, scripts
                          (scorep_*tests.sh) containing the tests are
                          generated in <builddir>/build-backend.
  --enable-shared[=PKGS]  build shared libraries [default=no]
  --enable-static[=PKGS]  build static libraries [default=yes]
  --enable-fast-install[=PKGS]
                          optimize for fast installation [default=yes]
  --disable-libtool-lock  avoid locking (might break parallel builds)
```

```
Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --with-sionlib[=<sionlib-bindir>]
                          Use an already installed sionlib. Provide path to
                          sionconfig. Auto-detected if already in $PATH.
  --with-pic              try to use only PIC/non-PIC objects [default=use
                          both]
  --with-gnu-ld           assume the C compiler uses GNU ld [default=no]
  --with-sysroot=DIR Search for dependent libraries within DIR
                        (or the compiler's sysroot if not specified).

Some influential environment variables:
  CC_FOR_BUILD
              C compiler command for the frontend build
  CXX_FOR_BUILD
              C++ compiler command for the frontend build
  F77_FOR_BUILD
              Fortran 77 compiler command for the frontend build
  FC_FOR_BUILD
              Fortran compiler command for the frontend build
  CPPFLAGS_FOR_BUILD
              (Objective) C/C++ preprocessor flags for the frontend build,
              e.g. -I<include dir> if you have headers in a nonstandard
              directory <include dir>
  CFLAGS_FOR_BUILD
              C compiler flags for the frontend build
  CXXFLAGS_FOR_BUILD
              C++ compiler flags for the frontend build
  FFLAGS_FOR_BUILD
              Fortran 77 compiler flags for the frontend build
  FCFLAGS_FOR_BUILD
              Fortran compiler flags for the frontend build
  LDFLAGS_FOR_BUILD
              linker flags for the frontend build, e.g. -L<lib dir> if you
              have libraries in a nonstandard directory <lib dir>
  LIBS_FOR_BUILD
              libraries to pass to the linker for the frontend build, e.g.
              -l<library>
  CC          C compiler command
  CFLAGS      C compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
              nonstandard directory <lib dir>
  LIBS        libraries to pass to the linker, e.g. -l<library>
  CPPFLAGS    (Objective) C/C++ preprocessor flags, e.g. -I<include dir> if
              you have headers in a nonstandard directory <include dir>
  CXX         C++ compiler command
  CXXFLAGS    C++ compiler flags
  CPP         C preprocessor
  CXXCPP      C++ preprocessor
  PYTHON      The python interpreter to use. Not a build requirement, but
              needed when developing. Python 2.5 or above, but no python 3.
              Use PYTHON=: to disable python support.
Use these variables to override the choices made by `configure' or to help
it to find libraries and programs with nonstandard names/locations.

Please report bugs to <support@score-p.org>.


Platform-specific instructions
******************************

Intel MIC
---------
   Building OTF2 for the Intel MIC platform requires some extra care, and in
some cases two installations into the same location.  Therefore, we strongly
recommend to strictly follow the procedure as described below.

  1. Ensure that Intel compilers are installed and available in $PATH, and
     that the Intel Manycore Platform Software Stack (MPSS) is installed.

  2. Configure OTF2 to use the MIC platform:

       ./configure --enable-platform-mic [other options, e.g., '--prefix']

  3. Build and install:

       make; make install

   On non-cross compiling systems (e.g., typical Linux clusters), that's it.
On cross-compiling systems (e.g., Cray XC30 with Xeon Phi daughter board), a
second installation of OTF2 *on top* of the just installed one is required to
provide a single installation serving login nodes, compute nodes, and MIC:

  4. Remove MIC program binaries, object files, and configure-generated files
     from the source code directory:

       make distclean

  5. Reconfigure for login/compute nodes using *identical directory options*
     (e.g., '--prefix' or '--bindir') as in step 2:

       ./configure [other options as usid in step 2]

     This will automatically detect the already existing native MIC build and
     enable the required support in the login node tools.

  6. Build and install:

       make; make install

   Note that this approach also works with VPATH builds (even with with two
separate build directories) as long as the same options defining directory
locations are passed in steps 2 and 5.
```

Installation Instructions

Basic Installation

Briefly, the shell commands './configure; make; make install' should configure, build, and install this package. The following more-detailed instructions are generic; see the 'README' file for instructions specific to this package. Some packages provide this 'INSTALL' file but do not implement all of the features documented below. The lack of an optional feature in a given package is not necessarily a bug. More recommendations for GNU packages can be found in *note Makefile Conventions: (standards)Makefile Conventions.

The 'configure' shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a 'Makefile' in each directory of the package. It may also create one or more '.h' files containing system-dependent definitions. Finally, it creates a shell script 'config.status' that you can run in the future to recreate the current configuration, and a file 'config.log' containing compiler output (useful mainly for debugging 'configure').

It can also use an optional file (typically called 'config.cache' and enabled with '--cache-file=config.cache' or simply '-C') that saves the results of its tests to speed up reconfiguring. Caching is disabled by default to prevent problems with accidental use of stale cache files.

If you need to do unusual things to compile the package, please try to figure out how 'configure' could check whether to do them, and mail diffs or instructions to the address given in the 'README' so they can be considered for the next release. If you are using the cache, and at some point 'config.cache' contains results you don't want to keep, you may remove or edit it.

The file 'configure.ac' (or 'configure.in') is used to create 'configure' by a program called 'autoconf'. You need 'configure.ac' if you want to change it or regenerate 'configure' using a newer version of 'autoconf'.

The simplest way to compile this package is:

1. 'cd' to the directory containing the package's source code and type './configure' to configure the package for your system.

Running ‘configure’ might take a while. While running, it prints some messages telling which features it is checking for.

2. Type ‘make’ to compile the package.

3. Optionally, type ‘make check’ to run any self-tests that come with the package, generally using the just-built uninstalled binaries.

4. Type ‘make install’ to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the ‘make install’ phase executed with root privileges.

5. Optionally, type ‘make installcheck’ to repeat any self-tests, but this time using the binaries in their final installed location. This target does not install anything. Running this target as a regular user, particularly if the prior ‘make install’ required root privileges, verifies that the installation completed correctly.

6. You can remove the program binaries and object files from the source code directory by typing ‘make clean’. To also remove the files that ‘configure’ created (so you can compile the package for a different kind of computer), type ‘make distclean’. There is also a ‘make maintainer-clean’ target, but that is intended mainly for the package’s developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.

7. Often, you can also type ‘make uninstall’ to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.

8. Some packages, particularly those that use Automake, provide ‘make distcheck’, which can by used by developers to test that all other targets like ‘make install’ and ‘make uninstall’ work correctly. This target is generally not run by end users.

Compilers and Options

Some systems require unusual options for compilation or linking that the ‘configure’ script does not know about. Run ‘./configure --help’ for details on some of the pertinent environment variables.

You can give ‘configure’ initial values for configuration parameters by setting variables in the command line or in the environment. Here is an example:

```
     ./configure CC=c99 CFLAGS=-g LIBS=-lposix
```

   *Note Defining Variables::, for more details.

Compiling For Multiple Architectures

   You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you can use GNU 'make'. 'cd' to the directory where you want the object files and executables to go and run the 'configure' script. 'configure' automatically checks for the source code in the directory that 'configure' is in and in '..'. This is known as a "VPATH" build.

   With a non-GNU 'make', it is safer to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use 'make distclean' before reconfiguring for another architecture.

   On MacOS X 10.5 and later systems, you can create libraries and executables that work on multiple system types--known as "fat" or "universal" binaries--by specifying multiple '-arch' options to the compiler but only a single '-arch' option to the preprocessor. Like this:

```
     ./configure CC="gcc -arch i386 -arch x86_64 -arch ppc -arch ppc64" \
                 CXX="g++ -arch i386 -arch x86_64 -arch ppc -arch ppc64" \
                 CPP="gcc -E" CXXCPP="g++ -E"
```

   This is not guaranteed to produce working output in all cases, you may have to build one architecture at a time and combine the results using the 'lipo' tool if you have problems.

Installation Names

   By default, 'make install' installs the package's commands under '/usr/local/bin', include files under '/usr/local/include', etc. You can specify an installation prefix other than '/usr/local' by giving 'configure' the option '--prefix=PREFIX', where PREFIX must be an absolute file name.

   You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you pass the option '--exec-prefix=PREFIX' to 'configure', the package uses PREFIX as the prefix for installing programs and libraries. Documentation and other data files still use the regular prefix.

   In addition, if you use an unusual directory layout you can give options like '--bindir=DIR' to specify different values for particular kinds of files. Run 'configure --help' for a list of the directories you can set and what kinds of files go in them. In general, the default for these options is expressed in terms of '${prefix}', so that specifying just '--prefix' will affect all of the other directory specifications that were not explicitly provided.

The most portable way to affect installation locations is to pass the correct locations to 'configure'; however, many packages provide one or both of the following shortcuts of passing variable assignments to the 'make install' command line to change installation locations without having to reconfigure or recompile.

The first method involves providing an override variable for each affected directory. For example, 'make install prefix=/alternate/directory' will choose an alternate location for all directory configuration variables that were expressed in terms of '${prefix}'. Any directories that were specified during 'configure', but not in terms of '${prefix}', must each be overridden at install time for the entire installation to be relocated. The approach of makefile variable overrides for each directory variable is required by the GNU Coding Standards, and ideally causes no recompilation. However, some platforms have known limitations with the semantics of shared libraries that end up requiring recompilation when using this method, particularly noticeable in packages that use GNU Libtool.

The second method involves providing the 'DESTDIR' variable. For example, 'make install DESTDIR=/alternate/directory' will prepend '/alternate/directory' before all installation names. The approach of 'DESTDIR' overrides is not required by the GNU Coding Standards, and does not work on platforms that have drive letters. On the other hand, it does better at avoiding recompilation issues, and works well even when some directory options were not specified in terms of '${prefix}' at 'configure' time.

Optional Features

If the package supports it, you can cause programs to be installed with an extra prefix or suffix on their names by giving 'configure' the option '--program-prefix=PREFIX' or '--program-suffix=SUFFIX'.

Some packages pay attention to '--enable-FEATURE' options to 'configure', where FEATURE indicates an optional part of the package. They may also pay attention to '--with-PACKAGE' options, where PACKAGE is something like 'gnu-as' or 'x' (for the X Window System). The 'README' should mention any '--enable-' and '--with-' options that the package recognizes.

For packages that use the X Window System, 'configure' can usually find the X include and library files automatically, but if it doesn't, you can use the 'configure' options '--x-includes=DIR' and '--x-libraries=DIR' to specify their locations.

Some packages offer the ability to configure how verbose the execution of 'make' will be. For these packages, running './configure --enable-silent-rules' sets the default to minimal output, which can be overridden with 'make V=1'; while running './configure --disable-silent-rules' sets the default to verbose, which can be overridden with 'make V=0'.

Particular systems

   On HP-UX, the default C compiler is not ANSI C compatible. If GNU CC is not installed, it is recommended to use the following options in order to use an ANSI C compiler:

```
     ./configure CC="cc -Ae -D_XOPEN_SOURCE=500"
```

and if that doesn't work, install pre-built binaries of GCC for HP-UX.

   On OSF/1 a.k.a. Tru64, some versions of the default C compiler cannot parse its '<wchar.h>' header file. The option '-nodtk' can be used as a workaround. If GNU CC is not installed, it is therefore recommended to try

```
     ./configure CC="cc"
```

and if that doesn't work, try

```
     ./configure CC="cc -nodtk"
```

   On Solaris, don't put '/usr/ucb' early in your 'PATH'. This directory contains several dysfunctional programs; working variants of these programs are available in '/usr/bin'. So, if you need '/usr/ucb' in your 'PATH', put it _after_ '/usr/bin'.

   On Haiku, software installed for all users goes in '/boot/common', not '/usr/local'. It is recommended to use the following options:

```
     ./configure --prefix=/boot/common
```

Specifying the System Type

   There may be some features 'configure' cannot figure out automatically, but needs to determine by the type of machine the package will run on. Usually, assuming the package is built to be run on the _same_ architectures, 'configure' can figure that out, but if it prints a message saying it cannot guess the machine type, give it the '--build=TYPE' option. TYPE can either be a short name for the system type, such as 'sun4', or a canonical name which has the form:

```
     CPU-COMPANY-SYSTEM
```

where SYSTEM can have one of these forms:

```
     OS
     KERNEL-OS
```

   See the file 'config.sub' for the possible values of each field. If 'config.sub' isn't included in this package, then this package doesn't need to know the machine type.

```
   If you are _building_ compiler tools for cross-compiling, you should
use the option `--target=TYPE' to select the type of system they will
produce code for.

   If you want to _use_ a cross compiler, that generates code for a
platform different from the build platform, you should specify the
"host" platform (i.e., that on which the generated programs will
eventually be run) with `--host=TYPE'.

Sharing Defaults
================

   If you want to set default values for `configure' scripts to share,
you can create a site shell script called `config.site' that gives
default values for variables like `CC', `cache_file', and `prefix'.
`configure' looks for `PREFIX/share/config.site' if it exists, then
`PREFIX/etc/config.site' if it exists.  Or, you can set the
`CONFIG_SITE' environment variable to the location of the site script.
A warning: not all `configure' scripts look for a site script.

Defining Variables
==================

   Variables not defined in a site shell script can be set in the
environment passed to `configure'.  However, some packages may run
configure again during the build, and the customized values of these
variables may be lost.  In order to avoid this problem, you should set
them in the `configure' command line, using `VAR=value'.  For example:

     ./configure CC=/usr/local2/bin/gcc

causes the specified `gcc' to be used as the C compiler (unless it is
overridden in the site shell script).

Unfortunately, this technique does not work for `CONFIG_SHELL' due to
an Autoconf bug.  Until the bug is fixed you can use this workaround:

     CONFIG_SHELL=/bin/bash /bin/bash ./configure CONFIG_SHELL=/bin/bash

`configure' Invocation
======================

   `configure' recognizes the following options to control how it
operates.

`--help'
`-h'
     Print a summary of all of the options to `configure', and exit.

`--help=short'
`--help=recursive'
     Print a summary of the options unique to this package's
     `configure', and exit.  The `short' variant lists options used
     only in the top level, while the 'recursive' variant lists options
     also present in any nested packages.

'--version'
'-V'
     Print the version of Autoconf used to generate the 'configure'
     script, and exit.

'--cache-file=FILE'
     Enable the cache: use and save the results of the tests in FILE,
     traditionally 'config.cache'.  FILE defaults to '/dev/null' to
     disable caching.

'--config-cache'
'-C'
     Alias for '--cache-file=config.cache'.

'--quiet'
'--silent'
'-q'
     Do not print messages saying which checks are being made.  To
     suppress all normal output, redirect it to '/dev/null' (any error
     messages will still be shown).

'--srcdir=DIR'
     Look for the package's source code in directory DIR.  Usually
     'configure' can determine that directory automatically.

'--prefix=DIR'
     Use DIR as the installation prefix.  *note Installation Names::
     for more details, including other options available for fine-tuning
     the installation locations.

'--no-create'
'-n'
     Run the configure checks, but stop before creating any output
     files.

'configure' also accepts some other, not widely useful, options.  Run
'configure --help' for more details.
```

Appendix B

Deprecated List

Global OTF2_AttributeList_AddString(OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute,
Use *OTF2_AttributeList_AddStringRef()* instead.

Global OTF2_AttributeList_GetString(const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attri
Use *OTF2_AttributeList_GetStringRef()* instead.

Global OTF2_EventSizeEstimator_GetSizeOfOmpAcquireLockEvent(OTF2_EventSizeEstimator ∗estimat
In version 1.2

Global OTF2_EventSizeEstimator_GetSizeOfOmpForkEvent(OTF2_EventSizeEstimator ∗estimator)
In version 1.2

Global OTF2_EventSizeEstimator_GetSizeOfOmpJoinEvent(OTF2_EventSizeEstimator ∗estimator)
In version 1.2

Global OTF2_EventSizeEstimator_GetSizeOfOmpReleaseLockEvent(OTF2_EventSizeEstimator ∗estimat
In version 1.2

Global OTF2_EventSizeEstimator_GetSizeOfOmpTaskCompleteEvent(OTF2_EventSizeEstimator ∗estima
In version 1.2

Global OTF2_EventSizeEstimator_GetSizeOfOmpTaskCreateEvent(OTF2_EventSizeEstimator ∗estimato
In version 1.2

Global OTF2_EventSizeEstimator_GetSizeOfOmpTaskSwitchEvent(OTF2_EventSizeEstimator ∗estimato
In version 1.2

Global OTF2_EvtWriter_OmpAcquireLock(OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList
In version 1.2

Global OTF2_EvtWriter_OmpFork(OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_
In version 1.2

Global OTF2_EvtWriter_OmpJoin(OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_T
In version 1.2

Global OTF2_EvtWriter_OmpReleaseLock(OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList,
In version 1.2

Global OTF2_EvtWriter_OmpTaskComplete(OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeLis
In version 1.2

Global OTF2_EvtWriter_OmpTaskCreate(OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList,
In version 1.2

Global OTF2_EvtWriter_OmpTaskSwitch(OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList,
In version 1.2

Group records_event In version 1.2

In version 1.2

In version 1.2

In version 1.2

In version 1.2

In version 1.2

In version 1.2

Appendix C

Module Documentation

C.1 Usage of OTF2 tools

Modules

- OTF2 config tool
- OTF2 print tool
- OTF2 snapshots tool
- OTF2 marker tool
- OTF2 estimator tool

C.2 OTF2 config tool

A call to otf2-config has the following syntax:

```
Usage: otf2-config [OPTION]... COMMAND

Commands:
  --cflags     prints additional compiler flags. They already contain
               the include flags
  --cppflags   prints the include flags for the OTF2 headers
  --libs       prints the required libraries for linking
  --ldflags    prints the required linker flags
  --cc         prints the C compiler name
  --features <FEATURE-CATEGORY>
               prints available features selected by <FEATURE-CATEGORY>.
               Available feature categories:
                * substrates
                * compressions
                * targets
  --help       prints this usage information
  --version    prints the version number of the OTF2 package and
  --otf2-revision
               prints the revision number of the OTF2 package
  --common-revision
               prints the revision number of the common package
  --interface-version
               prints the interface version number

Options:
  --target <TARGET>
               displays the requested information for the given <TARGET>.
               On non-cross compiling systems, the 'backend' target is ignored.
  --backend    equivalent to '--target backend' (deprecated)
  --cuda       specifies that the required flags are for the CUDA compiler
               nvcc
```

C.3 OTF2 print tool

A call to oft2-print has the following syntax:

```
Usage: otf2-print [OPTION]... [--] ANCHORFILE
Print selected content of the OTF2 archive specified by ANCHORFILE.

Options:
  -A, --show-all          print all output including definitions and anchor
                          file
  -G, --show-global-defs  print all global definitions
  -I, --show-info         print information from the anchor file
  -T, --show-thumbnails   print the headers from all thumbnails
  -M, --show-mappings     print mappings to global definitions
  -C, --show-clock-offsets
                          print clock offsets to global timer
      --timestamps=<FORMAT>
                          format of the timestamps. <FORMAT> is one of:
                           plain -  no formatting is done (default)
                           offset - timestamps are relative to the global offset
                                    (taken form the ClockProperties definition)
  -L, --location <LID>    limit output to location <LID>
  -s, --step <N>          step through output by steps of <N> events
      --time <MIN> <MAX>  limit output to events within time interval
      --system-tree       output system tree to dot-file
      --silent            only validate trace and do not print any events
      --unwind-calling-context
                          Unwind the calling context for each calling context
                          sample
  -d, --debug             turn on debug mode
  -V, --version           print version information
  -h, --help              print this help information
```

C.4 OTF2 snapshots tool

A call to oft2-snapshots has the following syntax:

```
Usage: otf2-snapshots [OPTION]... ANCHORFILE
Append snapshots to existing otf2 traces at given 'break' timestamps.

Options:
  -n, --number <BREAKS> Number of breaks (distributed regularly)
                        if -p and -t are not set, the default for -n is 10
                        breaks.
  -p <TICK_RATE>        Create break every <TICK_RATE> ticks
                        if both, -n and -p are specified the one producing
                        more breaks wins.
      --progress        Brief mode, print progress information.
      --verbose         Verbose mode, print break timestamps, i.e. snapshot
                        informations to stdout.
  -V, --version         Print version information.
  -h, --help            Print this help information.
```

C.5 OTF2 marker tool

A call to oft2-marker has the following syntax:

```
Usage: otf2-marker [OPTION] [ARGUMENTS]... ANCHORFILE
Read or edit a marker file.

Options:
                      Print all markers sorted by group.
      --def <GROUP> [<CATEGORY>]
                      Print all marker definitions of group <GROUP> or of
                      category <CATEGORY> from group <GROUP>.
      --defs-only     Print only marker definitions.
      --add-def <GROUP> <CATEGORY> <SEVERITY>
                      Add a new marker definition.
      --add <GROUP> <CATEGORY> <TIME> <SCOPE> <TEXT>
                      Add a marker to an existing definition.
      --remove-def <GROUP> [<CATEGORY>]
                      Remove all marker classes of group <GROUP> or only the
                      category <CATEGORY> of group <GROUP>; and all according
                      markers.
      --clear-def <GROUP> [<CATEGORY>]
                      Remove all markers of group <GROUP> or only of category
                      <CATEGORY> of group <GROUP>.
      --reset         Reset all marker.
  -V, --version       Print version information.
  -h, --help          Print this help information.

Argument descriptions:
  <GROUP>, <CATEGORY>, <TEXT>
```

```
                      Arbitrary strings.
  <SEVERITY>          One of:
                       * NONE
                       * LOW
                       * MEDIUM
                       * HIGH
  <TIME>              One of the following formats:
                       * <TIMESTAMP>
                         A valid timestamp inside the trace range
                         'global offset' and 'global offset' + 'trace
                         length'.
                       * <TIMESTAMP>+<DURATION>
                         <TIMESTAMP> and <TIMESTAMP> + <DURATION> must be valid
                         timestamps inside the trace range 'global
                         offset' and 'global offset' + 'trace length'.
                       * <TIMESTAMP-START>-<TIMESTAMP-END>
                         Two valid timestamps inside the trace range 'global
                         offset' and 'global offset' + 'trace length', with
                         <TIMESTAMP-START> <= <TIMESTAMP-END>.
                      See the CLOCK_PROPERTIES definition with the help
                      of the 'otf2-print -G' tool.
  <SCOPE>[:<SCOPE-REF>]
                       The <SCOPE> must be one of:
                       * GLOBAL
                       * LOCATION:<LOCATION-REF>
                       * LOCATION_GROUP:<LOCATION-GROUP-REF>
                       * SYSTEM_TREE_NODE:<SYSTEM-TREE-NODE-REF>
                       * GROUP:<GROUP-REF>
                       * COMM:<COMMUNICATOR-REF>
                      <SCOPE-REF> must be a valid definition reference of
                      the specified scope. Use 'otf2-print -G' for a list of
                      defined references.
                      There is no <SCOPE-REF> for <SCOPE> 'GLOBAL'.
                      For a scope 'GROUP' the type of the referenced
                      group must be 'OTF2_GROUP_TYPE_LOCATIONS' or
                      'OTF2_GROUP_TYPE_COMM_LOCATIONS'.
```

C.6 OTF2 estimator tool

A call to oft2-estimator has the following syntax:

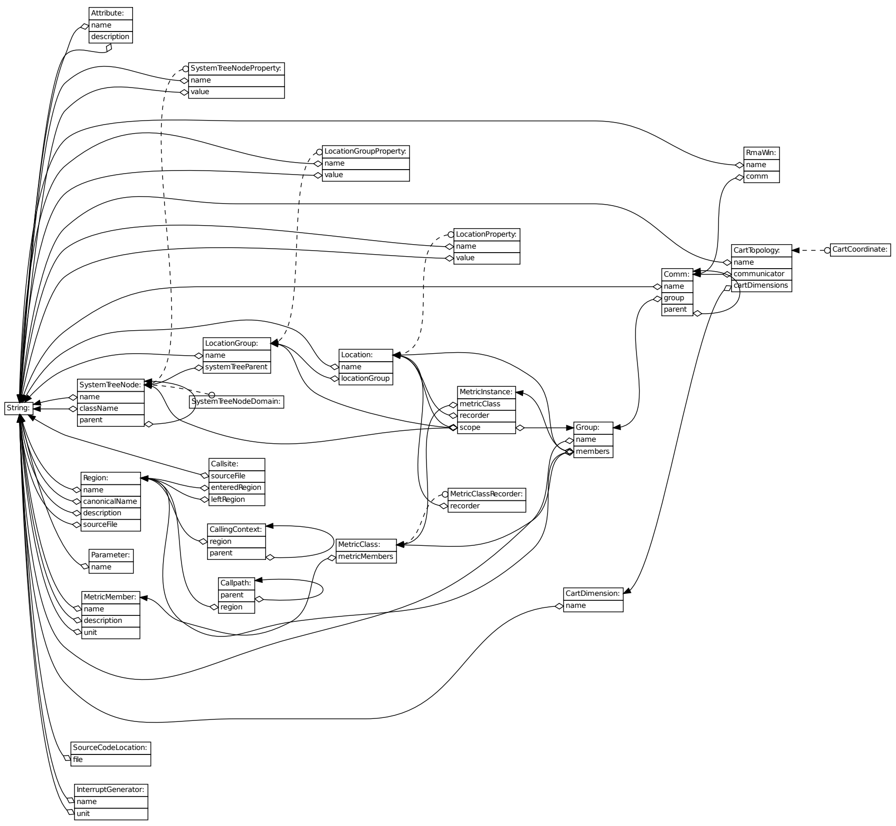
```
Usage: otf2-estimator [OPTION]...
This tool estimates the size of OTF2 events.
It will open a prompt to type in commands.

Options:
  -V, --version       Print version information.
  -h, --help          Print this help information.

Commands:
   list definitions              Lists all known definition names.
   list events                  Lists all known event names.
   list types                   Lists all known type names.
   set <DEFINITION> <NUMBER>    Specifies the number of definitions of a
                                type of defintions.
   get Timestamp                Prints the size an timestamp.
   get AttributeList [TYPES...] Prints the estimated size for an attribute
                                list with the given number of entries and
                                types.
   get <EVENT> [ARGS...]        Prints the estimated size of records for
                                <EVENT>.
   exit                         Exits the tool.

This tool provides an command line interface to the estimator API of the OTF2
library. It is based on an stream based protocol. Commands are send to the
standard input stream of the program and the result is written to the standard
output stream of the program. All definition and event names are in the
canonical CamelCase form. Numbers are printed in decimal. The TYPES are in
ALL_CAPS. See the output of the appropriate list commands. Arguments are
separated with an arbitrary number of white space. The get commands use
everything after the first white space separator verbatim as an key, which is
then printed as the result appended with the estimated size.

Here is a simple example. We have at most 4 region definitions and one metric
definition. We want to know the size of an timestamp, enter and leave event,
and an metric event with 4 values.

cat <<EOC | otf2-estimator
set Region 4
set Metric 1
get Timestamp
get Enter
get Leave
get Metric  4
exit
EOC
Timestamp 9
Enter 3
Leave 3
Metric  4 44
```

C.7 OTF2 records

Modules

- List of all definition records
- List of all event records
- List of all marker records
- List of all snapshot records

C.8 List of all definition records



C.9 ClockProperties

Defines the timer resolution and time range of this trace. There will be no event with a timestamp less than `globalOffset`, and no event with timestamp greater than (`globalOffset` + `traceLength`).

This definition is only valid as a global definition.

Attributes

uint64_t	timerResolution	Ticks per seconds.
uint64_t	globalOffset	A timestamp smaller than all event timestamps.

uint64_t	trace-Length	A timespan which includes the timespan between the smallest and greatest timestamp of all event timestamps.

See also

OTF2_GlobalDefWriter_WriteClockProperties()

Since

Version 1.0

C.10 Paradigm

Attests that the following parallel paradigm was available at the time when the trace was recorded, and vice versa. Note that this does not attest that the paradigm was used. For convenience, this also includes a proper name for the paradigm and a classification. This definition is only allowed to appear at most once in the definitions per *Paradigm*.

This definition is only valid as a global definition.

Attributes

OTF2_Paradigm	paradigm	The paradigm to attest.
OTF2_StringRef	name	The name of the paradigm. References a *String* definition.
OTF2_-ParadigmClass	paradigm-Class	The class of this paradigm.

See also

OTF2_GlobalDefWriter_WriteParadigm()

Since

Version 1.5

C.11 ParadigmProperty

Extensible annotation for the *Paradigm* definition.

The tuple (*paradigm*, *property*) must be unique.

This definition is only valid as a global definition.

Attributes

OTF2_Paradigm	paradigm	The paradigm to annotate.
OTF2_-ParadigmProperty	property	The property.
OTF2_Type	type	The type of this property. Must match with the defined type of the *property*.
OTF2_-AttributeValue	attribute-Value	The value of this property.value

See also

OTF2_GlobalDefWriter_WriteParadigmProperty()

Since

Version 1.5

C.12 MappingTable

Mapping tables are needed for situations where an ID is not globally known at measurement time. They are applied automatically at reading.

This definition is only valid as a local definition.

Attributes

OTF2_-MappingType	mapping-Type	Says to what type of ID the mapping table has to be applied.
const OTF2_IdMap∗	idMap	Mapping table.

See also

OTF2_DefWriter_WriteMappingTable()

Since

Version 1.0

C.13 ClockOffset

Clock offsets are used for clock corrections.

This definition is only valid as a local definition.

Attributes

OTF2_TimeStamp	time	Time when this offset was determined.
int64_t	offset	The offset to the global clock which was determined at `time`.
double	standard-Deviation	A possible standard deviation, which can be used as a metric for the quality of the offset.

See also

OTF2_DefWriter_WriteClockOffset()

Since

Version 1.0

C.14 *OTF2_StringRef* String

The string definition.

Attributes

const char∗	string	The string, null terminated.

See also

OTF2_GlobalDefWriter_WriteString()
OTF2_DefWriter_WriteString()

Since

Version 1.0

C.15 *OTF2_AttributeRef* Attribute

The attribute definition.

Attributes

OTF2_StringRef	name	Name of the attribute. References a *String* definition.
OTF2_StringRef	description	Description of the attribute. References a *String* definition. Since version 1.4.
OTF2_Type	type	Type of the attribute value.

See also

OTF2_GlobalDefWriter_WriteAttribute()
OTF2_DefWriter_WriteAttribute()

Since

Version 1.0

C.16 *OTF2_SystemTreeNodeRef* SystemTreeNode

The system tree node definition.

Attributes

OTF2_StringRef	name	Free form instance name of this node. References a *String* definition.
OTF2_StringRef	className	Free form class name of this node References a *String* definition.
OTF2_SystemTreeNodeRef	parent	Parent id of this node. May be *OTF2_UNDEFINED_SYSTEM_TREE_NODE* to indicate that there is no parent. References a *SystemTreeNode* definition.

Supplements

SystemTreeNodeProperty
SystemTreeNodeDomain

See also

OTF2_GlobalDefWriter_WriteSystemTreeNode()
OTF2_DefWriter_WriteSystemTreeNode()

Since

Version 1.0

C.17 *OTF2_LocationGroupRef* LocationGroup

The location group definition.

Attributes

OTF2_StringRef	name	Name of the group. References a *String* definition.
OTF2_-LocationGroupType	location-GroupType	Type of this group.
OTF2_-SystemTreeNodeRef	sys-temTreePar-ent	Parent of this location group in the sys-tem tree. References a *SystemTreeNode* definition.

Supplements

LocationGroupProperty

See also

OTF2_GlobalDefWriter_WriteLocationGroup()
OTF2_DefWriter_WriteLocationGroup()

Since

Version 1.0

C.18 *OTF2_LocationRef* Location

The location definition.

Attributes

OTF2_StringRef	name	Name of the location References a *String* definition.
OTF2_-LocationType	location-Type	Location type.
uint64_t	numberO-fEvents	Number of events this location has recorded.
OTF2_-LocationGroupRef	location-Group	Location group which includes this loca-tion. References a *LocationGroup* defini-tion.

Supplements

LocationProperty

See also

OTF2_GlobalDefWriter_WriteLocation()
OTF2_DefWriter_WriteLocation()

Since

Version 1.0

C.19 *OTF2_RegionRef* Region

The region definition.

Attributes

OTF2_StringRef	name	Name of the region (demangled name if available). References a *String* definition.
OTF2_StringRef	canonical-Name	Alternative name of the region (e.g. mangled name). References a *String* definition. Since version 1.1.
OTF2_StringRef	description	A more detailed description of this region. References a *String* definition.
OTF2_RegionRole	regionRole	Region role. Since version 1.1.
OTF2_Paradigm	paradigm	Paradigm. Since version 1.1.
OTF2_RegionFlag	regionFlags	Region flags. Since version 1.1.
OTF2_StringRef	sourceFile	The source file where this region was declared. References a *String* definition.
uint32_t	beginLi-neNumber	Starting line number of this region in the source file.
uint32_t	endLi-neNumber	Ending line number of this region in the source file.

See also

OTF2_GlobalDefWriter_WriteRegion()
OTF2_DefWriter_WriteRegion()

Since

Version 1.0

C.20 *OTF2_CallsiteRef* Callsite

The callsite definition.

Attributes

OTF2_StringRef	sourceFile	The source file where this call was made. References a *String* definition.
uint32_t	lineNumber	Line number in the source file where this call was made.
OTF2_RegionRef	enteredRegion	The region which was called. References a *Region* definition.
OTF2_RegionRef	leftRegion	The region which made the call. References a *Region* definition.

See also

OTF2_GlobalDefWriter_WriteCallsite()
OTF2_DefWriter_WriteCallsite()

Since

Version 1.0

C.21 *OTF2_CallpathRef* Callpath

The callpath definition.

Attributes

OTF2_CallpathRef	parent	The parent of this callpath. References a *Callpath* definition.
OTF2_RegionRef	region	The region of this callpath. References a *Region* definition.

See also

OTF2_GlobalDefWriter_WriteCallpath()
OTF2_DefWriter_WriteCallpath()

Since

Version 1.0

C.22 *OTF2_GroupRef* Group

The group definition.

Attributes

OTF2_StringRef	name	Name of this group References a *String* definition.
OTF2_GroupType	groupType	The type of this group. Since version 1.2.
OTF2_Paradigm	paradigm	The paradigm of this communication group. Since version 1.2.
OTF2_GroupFlag	groupFlags	Flags for this group. Since version 1.2.
uint32_t	numberOfMembers	The number of members in this group.
uint64_t	members [numberOfMembers]	The identifiers of the group members.

See also

OTF2_GlobalDefWriter_WriteGroup()
OTF2_DefWriter_WriteGroup()

Since

Version 1.0

C.23 *OTF2_MetricMemberRef* MetricMember

A metric is defined by a metric member definition. A metric member is always a member of a metric class. Therefore, a single metric is a special case of a metric class with only one member. It is not allowed to reference a metric member id in a metric event, but only metric class IDs.

Attributes

OTF2_StringRef	name	Name of the metric. References a *String* definition.
OTF2_StringRef	description	Description of the metric. References a *String* definition.

OTF2_MetricType	metricType	Metric type: PAPI, etc.
OTF2_MetricMode	metricMode	Metric mode: accumulative, fix, relative, etc.
OTF2_Type	valueType	Type of the value. Only *OTF2_TYPE_INT64*, *OTF2_TYPE_UINT64*, and *OTF2_TYPE_DOUBLE* are valid types. If this metric member is recorded in an *Metric* event, than this type and the type in the event must match.
OTF2_MetricBase	metricBase	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
int64_t	exponent	The values inside the Metric events should be scaled by the factor base^exponent, to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be *OTF2_BASE_BINARY* and the exponent 10. Than the writer does not need to scale the values up to bytes, but can directly write the KiBi values into the Metric event. At reading time, the reader can apply the scaling factor to get the value in its base unit, ie. in bytes.
OTF2_StringRef	unit	Unit of the metric. This needs to be the scale free base unit, ie. "bytes", "operations", or "seconds". In particular this unit should not have any scale prefix. References a *String* definition.

See also

OTF2_GlobalDefWriter_WriteMetricMember()
OTF2_DefWriter_WriteMetricMember()

Since

Version 1.0

C.24 *OTF2_MetricRef* MetricClass

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

Attributes

uint8_t	numberOf-Metrics	Number of metrics within the set.
OTF2_-MetricMemberRef	met-ricMembers [numberOf-Metrics]	List of metric members. References a *MetricMember* definition.
OTF2_-MetricOccurrence	metricOc-currence	Defines occurrence of a metric set.
OTF2_-RecorderKind	recorderKind	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

Supplements

MetricClassRecorder

See also

OTF2_GlobalDefWriter_WriteMetricClass()
OTF2_DefWriter_WriteMetricClass()

Since

Version 1.0

C.25 *OTF2_MetricRef* MetricInstance

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type *OTF2_METRIC_ASYNCHRONOUS*.

Attributes

OTF2_MetricRef	metricClass	The instanced *MetricClass*. This metric class must be of kind *OTF2_-RECORDER_KIND_ABSTRACT*. References a *MetricClass* definition.
OTF2_LocationRef	recorder	Recorder of the metric: location ID. References a *Location* definition.
OTF2_MetricScope	metric-Scope	Defines type of scope: location, location group, system tree node, or a generic group of locations.
uint64_t	scope	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

See also

OTF2_GlobalDefWriter_WriteMetricInstance()
OTF2_DefWriter_WriteMetricInstance()

Since

Version 1.0

C.26 *OTF2_CommRef* Comm

The communicator definition.

Attributes

OTF2_StringRef	name	The name given by calling MPI_Comm_-set_name on this communicator. Or the empty name to indicate that no name was given. References a *String* definition.
OTF2_GroupRef	group	The describing MPI group of this MPI communicator The group needs to be of type *OTF2_-GROUP_TYPE_COMM_GROUP* or *OTF2_GROUP_TYPE_COMM_SELF*. References a *Group* definition.
OTF2_CommRef	parent	The parent MPI communicator from which this communicator was created, if any. Use *OTF2_UNDEFINED_COMM* to indicate no parent. References a *Comm* definition.

See also

OTF2_GlobalDefWriter_WriteComm()
OTF2_DefWriter_WriteComm()

Since

Version 1.0

C.27 *OTF2_ParameterRef* Parameter

The parameter definition.

Attributes

OTF2_StringRef	name	Name of the parameter (variable name etc.) References a *String* definition.
OTF2_-ParameterType	parameter-Type	Type of the parameter, *OTF2_-ParameterType* for possible types.

See also

OTF2_GlobalDefWriter_WriteParameter()
OTF2_DefWriter_WriteParameter()

Since

Version 1.0

C.28 *OTF2_RmaWinRef* RmaWin

A window defines the communication context for any remote-memory access operation.

Attributes

OTF2_StringRef	name	Name, e.g. ’GASPI Queue 1’, ’NVidia Card 2’, etc.. References a *String* definition.
OTF2_CommRef	comm	Communicator object used to create the window. References a *Comm* definition.

See also

OTF2_GlobalDefWriter_WriteRmaWin()
OTF2_DefWriter_WriteRmaWin()

Since

Version 1.2

C.29 MetricClassRecorder

The metric class recorder definition.

Attributes

OTF2_MetricRef	metricClass	Parent *MetricClass* definition to which this one is a supplementary definition. References a *MetricClass* definition.
OTF2_LocationRef	recorder	The location which recorded the referenced metric class. References a *Location* definition.

See also

OTF2_GlobalDefWriter_WriteMetricClassRecorder()
OTF2_DefWriter_WriteMetricClassRecorder()

Since

Version 1.2

C.30 SystemTreeNodeProperty

An arbitrary key/value property for a *SystemTreeNode* definition.

Attributes

OTF2_SystemTreeNodeRef	systemTreeNode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
OTF2_StringRef	name	Name of the property. References a *String* definition.

OTF2_StringRef	value	Property value. References a *String* definition.

See also

OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty()
OTF2_DefWriter_WriteSystemTreeNodeProperty()

Since

Version 1.2

C.31 SystemTreeNodeDomain

The system tree node domain definition.

Attributes

OTF2_-SystemTreeNodeRef	sys-temTreeN-ode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
OTF2_-SystemTreeDomain	sys-temTreeDo-main	The domain in which the referenced *SystemTreeNode* operates in.

See also

OTF2_GlobalDefWriter_WriteSystemTreeNodeDomain()
OTF2_DefWriter_WriteSystemTreeNodeDomain()

Since

Version 1.2

C.32 LocationGroupProperty

An arbitrary key/value property for a *LocationGroup* definition.

Attributes

OTF2_-LocationGroupRef	location-Group	Parent *LocationGroup* definition to which this one is a supplementary definition. References a *LocationGroup* definition.
OTF2_StringRef	name	Name of the property. References a *String* definition.
OTF2_StringRef	value	Property value. References a *String* definition.

See also

OTF2_GlobalDefWriter_WriteLocationGroupProperty()
OTF2_DefWriter_WriteLocationGroupProperty()

Since

Version 1.3

C.33 LocationProperty

An arbitrary key/value property for a *Location* definition.

Attributes

OTF2_LocationRef	location	Parent *Location* definition to which this one is a supplementary definition. References a *Location* definition.
OTF2_StringRef	name	Name of the property. References a *String* definition.
OTF2_StringRef	value	Property value. References a *String* definition.

See also

OTF2_GlobalDefWriter_WriteLocationProperty()
OTF2_DefWriter_WriteLocationProperty()

Since

Version 1.3

C.34 *OTF2_CartDimensionRef* CartDimension

Each dimension in a Cartesian topology is composed of a global id, a name, its size, and whether it is periodic or not.

Attributes

OTF2_StringRef	name	The name of the cartesian topology dimension. References a *String* definition.
uint32_t	size	The size of the cartesian topology dimension.
OTF2_-CartPeriodicity	cartPeriodicity	Periodicity of the cartesian topology dimension.

See also

OTF2_GlobalDefWriter_WriteCartDimension()
OTF2_DefWriter_WriteCartDimension()

Since

Version 1.3

C.35 *OTF2_CartTopologyRef* CartTopology

Each topology is described by a global id, a reference to its name, a reference to a communicator, the number of dimensions, and references to those dimensions. The topology type is defined by the paradigm of the group referenced by the associated communicator.

Attributes

OTF2_StringRef	name	The name of the topology. References a *String* definition.
OTF2_CommRef	communicator	Communicator object used to create the topology. References a *Comm* definition.
uint8_t	numberOfDimensions	Number of dimensions.
OTF2_-CartDimensionRef	cartDimensions [numberOfDimensions]	The dimensions of this topology. References a *CartDimension* definition.

Supplements

CartCoordinate

See also

OTF2_GlobalDefWriter_WriteCartTopology()
OTF2_DefWriter_WriteCartTopology()

Since

Version 1.3

C.36 CartCoordinate

Defines the coordinate of the location referenced by the given rank (w.r.t. the communicator associated to the topology) in the referenced topology.

Attributes

OTF2_-CartTopologyRef	cartTopology	Parent *CartTopology* definition to which this one is a supplementary definition. References a *CartTopology* definition.
uint32_t	rank	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
uint8_t	numberOfDimensions	Number of dimensions.
uint32_t	coordinates [numberOfDimensions]	Coordinates, indexed by dimension.

See also

OTF2_GlobalDefWriter_WriteCartCoordinate()
OTF2_DefWriter_WriteCartCoordinate()

Since

Version 1.3

C.37 *OTF2_SourceCodeLocationRef* SourceCodeLocation

The definition of a source code location as tuple of the corresponding file name and line number.

When used to attach source code annotations to events, use the *OTF2_AttributeList* with a *Attribute* definition named `"SOURCE_CODE_LOCATION"` and typed *OTF2_-TYPE_SOURCE_CODE_LOCATION*.

Attributes

OTF2_StringRef	file	The name of the file for the source code location. References a *String* definition.
uint32_t	lineNum-ber	The line number for the source code location.

See also

OTF2_GlobalDefWriter_WriteSourceCodeLocation()
OTF2_DefWriter_WriteSourceCodeLocation()

Since

Version 1.5

C.38 *OTF2_CallingContextRef* CallingContext

Attributes

uint64_t	ip	Instruction pointer as the offset to the start of the function.
OTF2_RegionRef	region	The region. References a *Region* definition.
uint32_t	offsetLi-neNumber	The line offset inside the region.
OTF2_-CallingContextRef	parent	Parent id of this context. References a *CallingContext* definition.

See also

OTF2_GlobalDefWriter_WriteCallingContext()
OTF2_DefWriter_WriteCallingContext()

Since

Version 1.5

C.39 *OTF2_InterruptGeneratorRef* InterruptGenerator

Attributes

OTF2_StringRef	name	The name of this interrupt generator. References a *String* definition.
OTF2_StringRef	unit	The unit used by this interrupt generator for the period. References a *String* definition.
uint64_t	period	The period this interrupt generator generates interrupts.

See also

OTF2_GlobalDefWriter_WriteInterruptGenerator()
OTF2_DefWriter_WriteInterruptGenerator()

Since

Version 1.5

C.40 List of all event records

C.41 BufferFlush

This event signals that the internal buffer was flushed at the given time.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_TimeStamp	stopTime	The time the buffer flush finished.

See also

OTF2_EvtWriter_BufferFlush()

Since

Version 1.0

C.42 MeasurementOnOff

This event signals where the measurement system turned measurement on or off.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_-MeasurementMode	measure-mentMode	Is the measurement turned on (*OTF2_-MEASUREMENT_ON*) or off (*OTF2_-MEASUREMENT_OFF*)?

See also

OTF2_EvtWriter_MeasurementOnOff()

Since

Version 1.0

C.43 Enter

An enter record indicates that the program enters a code region.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RegionRef	region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_REGION* is available.

See also

OTF2_EvtWriter_Enter()

Since

Version 1.0

C.44 Leave

A leave record indicates that the program leaves a code region.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RegionRef	region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

See also

OTF2_EvtWriter_Leave()

Since

Version 1.0

C.45 MpiSend

A MpiSend record indicates that a MPI message send process was initiated (MPI_SEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint32_t	receiver	MPI rank of receiver in `communicator`.
OTF2_CommRef	communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length

See also

OTF2_EvtWriter_MpiSend()

Since

Version 1.0

C.46 MpiIsend

A MpiIsend record indicates that a MPI message send process was initiated (MPI_-ISEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint32_t	receiver	MPI rank of receiver in `communicator`.
OTF2_CommRef	communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length
uint64_t	requestID	ID of the related request

See also

OTF2_EvtWriter_MpiIsend()

Since

Version 1.0

C.47 MpiIsendComplete

Signals the completion of non-blocking send request.

Attributes

OTF2_LocationRef	location	The location where this event happened.

OTF2_TimeStamp	timestamp	The time when this event happened.
uint64_t	requestID	ID of the related request

See also

OTF2_EvtWriter_MpiIsendComplete()

Since

Version 1.0

C.48 MpiIrecvRequest

Signals the request of an receive, which can be completed later.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint64_t	requestID	ID of the requested receive

See also

OTF2_EvtWriter_MpiIrecvRequest()

Since

Version 1.0

C.49 MpiRecv

A MpiRecv record indicates that a MPI message was received (MPI_RECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint32_t	sender	MPI rank of sender in `communicator`.

OTF2_CommRef	communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length

See also

OTF2_EvtWriter_MpiRecv()

Since

Version 1.0

C.50 MpiIrecv

A MpiIrecv record indicates that a MPI message was received (MPI_IRECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint32_t	sender	MPI rank of sender in `communicator`.
OTF2_CommRef	communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length
uint64_t	requestID	ID of the related request

See also

OTF2_EvtWriter_MpiIrecv()

Since

Version 1.0

C.51 MpiRequestTest

This events appears if the program tests if a request has already completed but the test failed.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint64_t	requestID	ID of the related request

See also

OTF2_EvtWriter_MpiRequestTest()

Since

Version 1.0

C.52 MpiRequestCancelled

This events appears if the program canceled a request.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint64_t	requestID	ID of the related request

See also

OTF2_EvtWriter_MpiRequestCancelled()

Since

Version 1.0

C.53 MpiCollectiveBegin

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI_-GATHER, MPI_SCATTER etc.).

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.

See also

OTF2_EvtWriter_MpiCollectiveBegin()

Since

Version 1.0

C.54 MpiCollectiveEnd

A MpiCollectiveEnd record marks the end of an MPI collective operation (MPI_-GATHER, MPI_SCATTER etc.). It keeps the necessary information for this event: type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CollectiveOp	collectiveOp	Determines which collective operation it is.
OTF2_CommRef	communicator	Communicator References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	root	MPI rank of root in `communicator`.
uint64_t	sizeSent	Size of the sent message.
uint64_t	sizeReceived	Size of the received message.

See also

OTF2_EvtWriter_MpiCollectiveEnd()

Since

Version 1.0

C.55 OmpFork

An OmpFork record marks that an OpenMP Thread forks a thread team.

This event record is superseded by the *ThreadFork* event record and should not be used when the *ThreadFork* event record is in use.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint32_t	numberOfRequestedThreads	Requested size of the team.

See also

OTF2_EvtWriter_OmpFork()

Since

Version 1.0

Deprecated

In version 1.2

C.56 OmpJoin

An OmpJoin record marks that a team of threads is joint and only the master thread continues execution.

This event record is superseded by the *ThreadJoin* event record and should not be used when the *ThreadJoin* event record is in use.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.

See also

OTF2_EvtWriter_OmpJoin()

Since

Version 1.0

Deprecated

In version 1.2

C.57 OmpAcquireLock

An OmpAcquireLock record marks that a thread acquires an OpenMP lock.

This event record is superseded by the *ThreadAcquireLock* event record and should not be used when the *ThreadAcquireLock* event record is in use.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint32_t	lockID	ID of the lock.
uint32_t	acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

See also

OTF2_EvtWriter_OmpAcquireLock()

Since

Version 1.0

Deprecated

In version 1.2

C.58 OmpReleaseLock

An OmpReleaseLock record marks that a thread releases an OpenMP lock.

This event record is superseded by the *ThreadReleaseLock* event record and should not be used when the *ThreadReleaseLock* event record is in use.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint32_t	lockID	ID of the lock.
uint32_t	acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

See also

OTF2_EvtWriter_OmpReleaseLock()

Since

Version 1.0

Deprecated

In version 1.2

C.59 OmpTaskCreate

An OmpTaskCreate record marks that an OpenMP Task was/will be created in the current region.

This event record is superseded by the *ThreadTaskCreate* event record and should not be used when the *ThreadTaskCreate* event record is in use.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint64_t	taskID	Identifier of the newly created task instance.

See also

OTF2_EvtWriter_OmpTaskCreate()

Since

Version 1.0

Deprecated

In version 1.2

C.60 OmpTaskSwitch

An OmpTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

This event record is superseded by the *ThreadTaskSwitch* event record and should not be used when the *ThreadTaskSwitch* event record is in use.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint64_t	taskID	Identifier of the now active task instance.

See also

OTF2_EvtWriter_OmpTaskSwitch()

Since

Version 1.0

Deprecated

In version 1.2

C.61 OmpTaskComplete

An OmpTaskComplete record indicates that the execution of an OpenMP task has finished.

This event record is superseded by the *ThreadTaskComplete* event record and should not be used when the *ThreadTaskComplete* event record is in use.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
uint64_t	taskID	Identifier of the completed task instance.

See also

OTF2_EvtWriter_OmpTaskComplete()

Since

Version 1.0

Deprecated

In version 1.2

C.62 Metric

A metric event is always stored at the location that recorded the metric. A metric event can reference a metric class or metric instance. Therefore, metric classes and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_MetricRef	metric	Could be a metric class or a metric instance. References a *MetricClass*, or a *MetricInstance* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_METRIC* is available.
uint8_t	numberOfMetrics	Number of metrics with in the set.
OTF2_Type	typeIDs [numberOfMetrics]	List of metric types. These types must match that of the corresponding *MetricMember* definitions.

OTF2_MetricValue	metricValues [numberOfMetrics]	List of metric values.

See also

OTF2_EvtWriter_Metric()

Since

Version 1.0

C.63 ParameterString

A ParameterString record marks that in the current region, the specified string parameter has the specified value.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_ParameterRef	parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_PARAMETER* is available.
OTF2_StringRef	string	Value: Handle of a string definition References a *String* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_STRING* is available.

See also

OTF2_EvtWriter_ParameterString()

Since

Version 1.0

C.64 ParameterInt

A ParameterInt record marks that in the current region, the specified integer parameter has the specified value.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_-ParameterRef	parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_PARAMETER* is available.
int64_t	value	Value of the recorded parameter.

See also

OTF2_EvtWriter_ParameterInt()

Since

Version 1.0

C.65 ParameterUnsignedInt

A ParameterUnsignedInt record marks that in the current region, the specified unsigned integer parameter has the specified value.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_-ParameterRef	parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_PARAMETER* is available.
uint64_t	value	Value of the recorded parameter.

See also

OTF2_EvtWriter_ParameterUnsignedInt()

Since

Version 1.0

C.66 RmaWinCreate

An RmaWinCreate record denotes the creation of an RMA window.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window created. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.

See also

OTF2_EvtWriter_RmaWinCreate()

Since

Version 1.2

C.67 RmaWinDestroy

An RmaWinDestroy record denotes the destruction of an RMA window.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window destructed. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.

See also

OTF2_EvtWriter_RmaWinDestroy()

Since

Version 1.2

C.68 RmaCollectiveBegin

An RmaCollectiveBegin record denotes the beginnig of a collective RMA operation.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.

See also

OTF2_EvtWriter_RmaCollectiveBegin()

Since

Version 1.2

C.69 RmaCollectiveEnd

An RmaCollectiveEnd record denotes the end of a collective RMA operation.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CollectiveOp	collectiveOp	Determines which collective operation it is.
OTF2_RmaSyncLevel	syncLevel	Synchronization level of this collective operation.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
uint32_t	root	Root process for this operation.
uint64_t	bytesSent	Bytes sent in operation.
uint64_t	bytesReceived	Bytes receives in operation.

See also

OTF2_EvtWriter_RmaCollectiveEnd()

Since

Version 1.2

C.70 RmaGroupSync

An RmaGroupSync record denotes the synchronization with a subgroup of processes on a window.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_-RmaSyncLevel	syncLevel	Synchronization level of this collective operation.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
OTF2_GroupRef	group	Group of remote processes involved in synchronization. References a *Group* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_GROUP* is available.

See also

OTF2_EvtWriter_RmaGroupSync()

Since

Version 1.2

C.71 RmaRequestLock

An RmaRequestLock record denotes the time a lock was requested and with it the earliest time it could have been granted. It is used to mark (possibly) non-blocking lock request, as defined by the MPI standard.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint32_t	remote	Rank of the locked remote process.
uint64_t	lockId	ID of the lock acquired, if multiple locks are defined on a window.
OTF2_LockType	lockType	Type of lock acquired.

See also

OTF2_EvtWriter_RmaRequestLock()

Since

Version 1.2

C.72 RmaAcquireLock

An RmaAcquireLock record denotes the time a lock was acquired by the process.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint32_t	remote	Rank of the locked remote process.
uint64_t	lockId	ID of the lock acquired, if multiple locks are defined on a window.
OTF2_LockType	lockType	Type of lock acquired.

See also

OTF2_EvtWriter_RmaAcquireLock()

Since

Version 1.2

C.73 RmaTryLock

An RmaTryLock record denotes the time of an unsuccessful attempt to acquire the lock.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint32_t	remote	Rank of the locked remote process.
uint64_t	lockId	ID of the lock acquired, if multiple locks are defined on a window.
OTF2_LockType	lockType	Type of lock acquired.

See also

OTF2_EvtWriter_RmaTryLock()

Since

Version 1.2

C.74 RmaReleaseLock

An RmaReleaseLock record denotes the time the lock was released.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.

OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint32_t	remote	Rank of the locked remote process.
uint64_t	lockId	ID of the lock released, if multiple locks are defined on a window.

See also

OTF2_EvtWriter_RmaReleaseLock()

Since

Version 1.2

C.75 RmaSync

An RmaSync record denotes the direct synchronization with a possibly remote process.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint32_t	remote	Rank of the locked remote process.
OTF2_-RmaSyncType	syncType	Type of synchronization.

See also

OTF2_EvtWriter_RmaSync()

Since

Version 1.2

C.76 RmaWaitChange

An RmaWaitChange record denotes the change of a window that was waited for.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.

See also

OTF2_EvtWriter_RmaWaitChange()

Since

Version 1.2

C.77 RmaPut

An RmaPut record denotes the time a put operation was issued.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint32_t	remote	Rank of the target process.
uint64_t	bytes	Bytes sent to target.
uint64_t	matchingId	ID used for matching the corresponding completion record.

See also

OTF2_EvtWriter_RmaPut()

Since

Version 1.2

C.78 RmaGet

An RmaGet record denotes the time a get operation was issued.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint32_t	remote	Rank of the target process.
uint64_t	bytes	Bytes received from target.
uint64_t	matchingId	ID used for matching the corresponding completion record.

See also

OTF2_EvtWriter_RmaGet()

Since

Version 1.2

C.79 RmaAtomic

An RmaAtomic record denotes the time a atomic operation was issued.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.

uint32_t	remote	Rank of the target process.
OTF2_-RmaAtomicType	type	Type of atomic operation.
uint64_t	bytesSent	Bytes sent to target.
uint64_t	bytesRe-ceived	Bytes received from target.
uint64_t	matchingId	ID used for matching the corresponding completion record.

See also

OTF2_EvtWriter_RmaAtomic()

Since

Version 1.2

C.80 RmaOpCompleteBlocking

An RmaOpCompleteBlocking record denotes the local completion of a blocking RMA operation.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint64_t	matchingId	ID used for matching the corresponding RMA operation record.

See also

OTF2_EvtWriter_RmaOpCompleteBlocking()

Since

Version 1.2

C.81 RmaOpCompleteNonBlocking

An RmaOpCompleteNonBlocking record denotes the local completion of a non-blocking RMA operation.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint64_t	matchingId	ID used for matching the corresponding RMA operation record.

See also

OTF2_EvtWriter_RmaOpCompleteNonBlocking()

Since

Version 1.2

C.82 RmaOpTest

An RmaOpTest record denotes that a non-blocking RMA operation has been tested for completion unsuccessfully.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint64_t	matchingId	ID used for matching the corresponding RMA operation record.

See also

OTF2_EvtWriter_RmaOpTest()

Since

Version 1.2

C.83 RmaOpCompleteRemote

An RmaOpCompleteRemote record denotes the remote completion of an RMA operation.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.
uint64_t	matchingId	ID used for matching the corresponding RMA operation record.

See also

OTF2_EvtWriter_RmaOpCompleteRemote()

Since

Version 1.2

C.84 ThreadFork

An ThreadFork record marks that an thread forks a thread team.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_Paradigm	model	The threading paradigm this event took place.

uint32_t	numberOfRequestedThreads	Requested size of the team.

See also

OTF2_EvtWriter_ThreadFork()

Since

Version 1.2

C.85 ThreadJoin

An ThreadJoin record marks that a team of threads is joint and only the master thread continues execution.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_Paradigm	model	The threading paradigm this event took place.

See also

OTF2_EvtWriter_ThreadJoin()

Since

Version 1.2

C.86 ThreadTeamBegin

The current location enters the specified thread team.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.

OTF2_CommRef	threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_COMM* is available.

See also

OTF2_EvtWriter_ThreadTeamBegin()

Since

Version 1.2

C.87 ThreadTeamEnd

The current location leaves the specified thread team.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CommRef	threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_COMM* is available.

See also

OTF2_EvtWriter_ThreadTeamEnd()

Since

Version 1.2

C.88 ThreadAcquireLock

An ThreadAcquireLock record marks that a thread acquires an lock.

Attributes

OTF2_LocationRef	location	The location where this event happened.

OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_Paradigm	model	The threading paradigm this event took place.
uint32_t	lockID	ID of the lock.
uint32_t	acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

See also

OTF2_EvtWriter_ThreadAcquireLock()

Since

Version 1.2

C.89 ThreadReleaseLock

An ThreadReleaseLock record marks that a thread releases an lock.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_Paradigm	model	The threading paradigm this event took place.
uint32_t	lockID	ID of the lock.
uint32_t	acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

See also

OTF2_EvtWriter_ThreadReleaseLock()

Since

Version 1.2

C.90 ThreadTaskCreate

An ThreadTaskCreate record marks that an task in was/will be created and will be processed by the specified thread team.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CommRef	threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	creatingThread	Creating thread of this task.
uint32_t	generationNumber	Thread-private generation number of task's creating thread.

See also

OTF2_EvtWriter_ThreadTaskCreate()

Since

Version 1.2

C.91 ThreadTaskSwitch

An ThreadTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.

OTF2_CommRef	threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_COMM* is available.
uint32_t	creatingThread	Creating thread of this task.
uint32_t	generationNumber	Thread-private generation number of task's creating thread.

See also

OTF2_EvtWriter_ThreadTaskSwitch()

Since

Version 1.2

C.92 ThreadTaskComplete

An ThreadTaskComplete record indicates that the execution of an OpenMP task has finished.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CommRef	threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_COMM* is available.
uint32_t	creatingThread	Creating thread of this task.
uint32_t	generationNumber	Thread-private generation number of task's creating thread.

See also

OTF2_EvtWriter_ThreadTaskComplete()

Since

Version 1.2

C.93 ThreadCreate

The location created successfully a new thread.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CommRef	thread-Contingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint64_t	sequence-Count	A `threadContingent` unique number. The corresponding *ThreadBegin* event does have the same number.

See also

OTF2_EvtWriter_ThreadCreate()

Since

Version 1.3

C.94 ThreadBegin

Marks the begin of a thread created by another thread.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CommRef	thread-Contingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint64_t	sequence-Count	A `threadContingent` unique number. The corresponding *ThreadCreate* event does have the same number.

See also

OTF2_EvtWriter_ThreadBegin()

Since

Version 1.3

C.95 ThreadWait

The location waits for the completion of another thread.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CommRef	thread-Contingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint64_t	sequence-Count	A `threadContingent` unique number. The corresponding *ThreadEnd* event does have the same number.

See also

OTF2_EvtWriter_ThreadWait()

Since

Version 1.3

C.96 ThreadEnd

Marks the end of a thread.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.

OTF2_CommRef	threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint64_t	sequenceCount	A `threadContingent` unique number. The corresponding *ThreadWait* event does have the same number. *OTF2_UNDEFINED_UINT64* in case no corresponding *ThreadWait* event exists.

See also

OTF2_EvtWriter_ThreadEnd()

Since

Version 1.3

C.97 CallingContextSample

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_CallingContextRef	callingContext	References a *CallingContext* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_CALLING_CONTEXT* is available.

uint32_t	unwind-Distance	The unwindContext specifies the first context whose ip(return adress) was still marked since the last sample this means that no progress was made in the repsective region The last region that was not returned from since the last sample Is one stack level higher, but may now be at at different line number OTF2_CallingContextRef unwindContext; However, instead of this we specify the distance (number of intermediate edges) between the calling context and the unwind context Note: unwindDistance=0 would mean no progress in the leaf region since the last sample which is unlikely If not available, UNDEFINED should be used.
OTF2_-InterruptGeneratorRe	interrupt-Generator	References a *InterruptGenerator* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_INTERRUPT_-GENERATOR* is available.

See also

OTF2_EvtWriter_CallingContextSample()

Since

Version 1.5

C.98 List of all marker records

C.99 *OTF2_MarkerRef* DefMarker

Group markers by name and severity.

Attributes

const char∗	marker-Group	Group name, e.g., "MUST", ...
const char∗	marker-Category	Marker category, e.g., "Argument type error", ...

OTF2_-MarkerSeverity	severity	The severity for these markers.

See also

OTF2_MarkerWriter_WriteDefMarker()

Since

Version 1.2

C.100 Marker

A user marker instance, with implied time stamp.

Attributes

OTF2_TimeStamp	timestamp	The time when this marker happened.
OTF2_TimeStamp	duration	A possible duration of this marker. May be 0.
OTF2_MarkerRef	marker	Groups this marker by name and severity. References a *DefMarker* definition.
OTF2_-MarkerScope	scope	The type of scope of this marker instance.
uint64_t	scopeRef	The scope instance of this marker. Depends on `scope`.
const char∗	text	A textual description for this marker.

See also

OTF2_MarkerWriter_WriteMarker()

Since

Version 1.2

C.101 List of all snapshot records

C.102 SnapshotStart

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one *SnapshotStart* record and closes with one *SnapshotEnd* record. All snapshot records inbetween are ordered by the `origEventTime`, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
uint64_t	numberOfRecord	Number of snapshot event records in this snapshot. Excluding the *SnapshotEnd* record.

See also

OTF2_SnapWriter_SnapshotStart()

Since

Version 1.2

C.103 SnapshotEnd

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use *OTF2_EvtReader_Seek* with `contReadPos` as the position.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
uint64_t	contReadPos	Position to continue reading in the event trace.

See also

OTF2_SnapWriter_SnapshotEnd()

Since

Version 1.2

C.104 MeasurementOnOffSnap

The last occurrence of an *MeasurementOnOff* event of this location, if any.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
OTF2_-MeasurementMode	measure-mentMode	Is the measurement turned on (*OTF2_-MEASUREMENT_ON*) or off (*OTF2_-MEASUREMENT_OFF*)?

See also

MeasurementOnOff event
OTF2_SnapWriter_MeasurementOnOff()

Since

Version 1.2

C.105 EnterSnap

This record exists for each *Enter* event where the corresponding *Leave* event did not occur before the snapshot.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
OTF2_RegionRef	region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_REGION* is available.

See also

Enter event

OTF2_SnapWriter_Enter()

Since

Version 1.2

C.106 MpiSendSnap

This record exists for each *MpiSend* event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event. Note that it may so, that a previous *MpiIsend* with the same envelope than this one is neither completed not canceled yet, thus the matching receive may already occurred, but the matching couldn't be done yet.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint32_t	receiver	MPI rank of receiver in `communicator`.
OTF2_CommRef	communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length

See also

MpiSend event
OTF2_SnapWriter_MpiSend()

Since

Version 1.2

C.107 MpiIsendSnap

This record exists for each *MpiIsend* event where an corresponding *MpiIsendComplete* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIsendComplete* did occurred (the *MpiIsendCompleteSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. (This could either be an*MpiRecv* or an *MpiIrecv* event.)

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint32_t	receiver	MPI rank of receiver in `communicator`.
OTF2_CommRef	communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length
uint64_t	requestID	ID of the related request

See also

MpiIsend event
OTF2_SnapWriter_MpiIsend()

Since

Version 1.2

C.108 MpiIsendCompleteSnap

This record exists for each *MpiIsend* event where the corresponding *MpiIsendComplete* event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an *MpiRecv* or an *MpiIrecv* event.) .

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint64_t	requestID	ID of the related request

See also

MpiIsendComplete event
OTF2_SnapWriter_MpiIsendComplete()

Since

Version 1.2

C.109 MpiRecvSnap

This record exists for each *MpiRecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint32_t	sender	MPI rank of sender in `communicator`.
OTF2_CommRef	communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length

See also

MpiRecv event
OTF2_SnapWriter_MpiRecv()

Since

Version 1.2

C.110 MpiIrecvRequestSnap

This record exists for each *MpiIrecvRequest* event where an corresponding *MpiIrecv* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIrecv* did occurred (the *MpiIrecvSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEventTime	The original time this event happended.
uint64_t	requestID	ID of the requested receive

See also

MpiIrecvRequest event
OTF2_SnapWriter_MpiIrecvRequest()

Since

Version 1.2

C.111 MpiIrecvSnap

This record exists for each *MpiIrecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint32_t	sender	MPI rank of sender in `communicator`.
OTF2_CommRef	communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length
uint64_t	requestID	ID of the related request

See also

MpiIrecv event
OTF2_SnapWriter_MpiIrecv()

Since

Version 1.2

C.112 MpiCollectiveBeginSnap

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.

See also

MpiCollectiveBegin event
OTF2_SnapWriter_MpiCollectiveBegin()

Since

Version 1.2

C.113 MpiCollectiveEndSnap

Indicates that this location completed a collective operation localy but not all of the participating locations completed the operation yet. The corresponding *MpiCollectiveBeginSnap* record is still in the snapshot though.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEventTime	The original time this event happended.
OTF2_CollectiveOp	collectiveOp	Determines which collective operation it is.
OTF2_CommRef	communicator	Communicator References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
uint32_t	root	MPI rank of root in `communicator`.
uint64_t	sizeSent	Size of the sent message.
uint64_t	sizeReceived	Size of the received message.

See also

MpiCollectiveEnd event
OTF2_SnapWriter_MpiCollectiveEnd()

Since

Version 1.2

C.114 OmpForkSnap

This record exists for each *OmpFork* event where the corresponding *OmpJoin* did not occurred before this snapshot.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint32_t	num-berOfRe-quest-edThreads	Requested size of the team.

See also

OmpFork event
OTF2_SnapWriter_OmpFork()

Since

Version 1.2

C.115 OmpAcquireLockSnap

This record exists for each *OmpAcquireLock* event where the corresponding *OmpReleaseLock* did not occurred before this snapshot yet.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint32_t	lockID	ID of the lock.
uint32_t	acquisi-tionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

See also

OmpAcquireLock event
OTF2_SnapWriter_OmpAcquireLock()

Since

Version 1.2

C.116 OmpTaskCreateSnap

This record exists for each *OmpTaskCreate* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEventTime	The original time this event happended.
uint64_t	taskID	Identifier of the newly created task instance.

See also

OmpTaskCreate event
OTF2_SnapWriter_OmpTaskCreate()

Since

Version 1.2

C.117 OmpTaskSwitchSnap

This record exists for each *OmpTaskSwitch* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEventTime	The original time this event happended.
uint64_t	taskID	Identifier of the now active task instance.

See also

OmpTaskSwitch event
OTF2_SnapWriter_OmpTaskSwitch()

Since

Version 1.2

C.118 MetricSnap

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

As an exception for metric classes where the metric mode detontes an *OTF2_-METRIC_VALUE_RELATIVE* mode the value indicates the accumulation of all previous metric values recorded.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
OTF2_MetricRef	metric	Could be a metric class or a metric instance. References a *MetricClass*, or a *MetricInstance* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-METRIC* is available.
uint8_t	numberOf-Metrics	Number of metrics with in the set.
OTF2_Type	typeIDs [numberOf-Metrics]	List of metric types. These types must match that of the corresponding *Met-ricMember* definitions.
OTF2_MetricValue	metricVal-ues [numberOf-Metrics]	List of metric values.

See also

Metric event
OTF2_SnapWriter_Metric()

Since

Version 1.2

C.119 ParameterStringSnap

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
OTF2_-ParameterRef	parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_PARAMETER* is available.
OTF2_StringRef	string	Value: Handle of a string definition References a *String* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-STRING* is available.

See also

ParameterString event
OTF2_SnapWriter_ParameterString()

Since

Version 1.2

C.120 ParameterIntSnap

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
OTF2_-ParameterRef	parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_PARAMETER* is available.
int64_t	value	Value of the recorded parameter.

See also

ParameterInt event
OTF2_SnapWriter_ParameterInt()

Since

Version 1.2

C.121 ParameterUnsignedIntSnap

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
OTF2_-ParameterRef	parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_PARAMETER* is available.

uint64_t	value	Value of the recorded parameter.

See also

ParameterUnsignedInt event
OTF2_SnapWriter_ParameterUnsignedInt()

Since

Version 1.2

C.122 OTF2 usage examples

Modules

- Usage in writing mode - a simple example
- How to use the attribute list for writing additional attributes to event records
- Usage in reading mode - MPI example
- Usage in writing mode - MPI example
- Usage in reading mode - a simple example

C.123 Usage in writing mode - a simple example

This is a short example of how to use the OTF2 writing interface. This example is available as source code in the file **otf2_writer_example.c**. See also **otf2_openmp_writer_example.c** and **otf2_pthread_writer_example.c** when writing with multiple threads.
First include the OTF2 header.

```
#include <otf2/otf2.h>
```

For this example an additional include statement is necessary.

```
#include <stdlib.h>
```

Furthermore this example uses a function delivering dummy timestamps. Real world applications will use a timer like clock_gettime.

```
static OTF2_TimeStamp
get_time( void )
```

```
{
    static uint64_t sequence;
    return sequence++;
}
```

Define a pre and post flush callback. If no memory is left in OTF2's internal memory buffer or the writer handle is closed a memory buffer flushing routine is triggered. The pre flush callback is triggered right before a buffer flush. It needs to return either OTF2_FLUSH to flush the recorded data to a file or OTF2_NO_-FLUSH to supress flushing data to a file. The post flush callback is triggered right after a memory buffer flush. It has to return a current timestamp which is recorded to mark the time spend in a buffer flush. The callbacks are passed via a struct to OTF2.

```
static OTF2_FlushType
pre_flush( void*            userData,
           OTF2_FileType    fileType,
           OTF2_LocationRef location,
           void*            callerData,
           bool             final )
{
    return OTF2_FLUSH;
}

static OTF2_TimeStamp
post_flush( void*            userData,
            OTF2_FileType    fileType,
            OTF2_LocationRef location )
{
    return get_time();
}

static OTF2_FlushCallbacks flush_callbacks =
{
    .otf2_pre_flush  = pre_flush,
    .otf2_post_flush = post_flush
};
```

Now everything is prepared to begin with the main program.

```
int
main( int    argc,
      char** argv )
{
```

Create new archive handle.

```
    OTF2_Archive* archive = OTF2_Archive_Open( "ArchivePath",
                                               "ArchiveName",
                                               OTF2_FILEMODE_WRITE,
                                               1024 * 1024 /* event chunk size */
      ,
                                               4 * 1024 * 1024 /* def chunk size
      */,
                                               OTF2_SUBSTRATE_POSIX,
                                               OTF2_COMPRESSION_NONE );
```

Set the previously defined flush callbacks.

```
    OTF2_Archive_SetFlushCallbacks( archive, &flush_callbacks, NULL );
```

We will operate in an serial context.

```
    OTF2_Archive_SetSerialCollectiveCallbacks( archive );
```

Now we can create the event files. Though physical files aren't created yet.

```
    OTF2_Archive_OpenEvtFiles( archive );
```

Get a local event writer for location 0.

```
    OTF2_EvtWriter* evt_writer = OTF2_Archive_GetEvtWriter( archive, 0 );
```

Write an enter and a leave record for region 0 to the local event writer.

```
    OTF2_EvtWriter_Enter( evt_writer,
                          NULL,
                          get_time(),
                          0 /* region */ );
    OTF2_EvtWriter_Leave( evt_writer,
                          NULL,
                          get_time(),
                          0 /* region */ );
```

Now close the event writer, before closing the event files collectivly.

```
    OTF2_Archive_CloseEvtWriter( archive, evt_writer );
```

After we wrote all of the events we close the event files again.

```
    OTF2_Archive_CloseEvtFiles( archive );
```

Now write the global definitions by getting an writer object for it.

```
OTF2_GlobalDefWriter* global_def_writer = OTF2_Archive_GetGlobalDefWriter( ar
  chive );
```

We need to define the clock used for this trace and the overall timestamp range.

```
OTF2_GlobalDefWriter_WriteClockProperties( global_def_writer,
                                           1 /* 1 tick per second */,
                                           0 /* epoch */,
                                           2 /* length */ );
```

Now we can start writing the referenced definitions, starting with the strings.

```
OTF2_GlobalDefWriter_WriteString( global_def_writer, 0, "" );
OTF2_GlobalDefWriter_WriteString( global_def_writer, 1, "Master Process" );
OTF2_GlobalDefWriter_WriteString( global_def_writer, 2, "Main Thread" );
OTF2_GlobalDefWriter_WriteString( global_def_writer, 3, "MyFunction" );
OTF2_GlobalDefWriter_WriteString( global_def_writer, 4, "Alternative function
   name (e.g. mangled one)" );
OTF2_GlobalDefWriter_WriteString( global_def_writer, 5, "Computes something"
  );
OTF2_GlobalDefWriter_WriteString( global_def_writer, 6, "MyHost" );
OTF2_GlobalDefWriter_WriteString( global_def_writer, 7, "node" );
```

Write definition for the code region which was just entered and left to the global definition writer.

```
OTF2_GlobalDefWriter_WriteRegion( global_def_writer,
                                  0 /* id */,
                                  3 /* region name  */,
                                  4 /* alternative name */,
                                  5 /* description */,
                                  OTF2_REGION_ROLE_FUNCTION,
                                  OTF2_PARADIGM_USER,
                                  OTF2_REGION_FLAG_NONE,
                                  0 /* source file */,
                                  0 /* begin lno */,
                                  0 /* end lno */ );
```

Write the system tree including a definition for the location group to the global definition writer.

```
OTF2_GlobalDefWriter_WriteSystemTreeNode( global_def_writer,
                                          0 /* id */,
                                          6 /* name */,
                                          7 /* class */,
                                                OTF2_UNDEFINED_SYSTEM_TREE_NODE /*
      parent */ );
    OTF2_GlobalDefWriter_WriteLocationGroup( global_def_writer,
                                             0 /* id */,
                                             1 /* name */,
                                             OTF2_LOCATION_GROUP_TYPE_PROCESS,
                                             0 /* system tree */ );
```

Write a definition for the location to the global definition writer.

```
    OTF2_GlobalDefWriter_WriteLocation( global_def_writer,
                                        0 /* id */,
                                        2 /* name */,
                                        OTF2_LOCATION_TYPE_CPU_THREAD,
                                        2 /* # events */,
                                        0 /* location group */ );
```

At the end, close the archive and exit.

```
    OTF2_Archive_Close( archive );

    return EXIT_SUCCESS;
}
```

To compile your program use a command like the following. Note that we need to activate the C99 standard explicitly for GCC.

```
    gcc -std=c99 `otf2-config --cflags` \
            -c otf2_writer_example.c \
            -o otf2_writer_example.o
```

Now you can link your program with:

```
    gcc otf2_writer_example.o \
            `otf2-config --ldflags` \
            `otf2-config --libs` \
            -o otf2_writer_example
```

C.124 How to use the attribute list for writing additional attributes to event records

First create an attribute list handle.

```
  OTF2_AttributeList attribute_list = OTF2_AttributeList_New();
```

To write your additional attribute to an event record add your attributes to an empty attribute list right before you call the routine to write the event.

```
OTF2_AttributeValue attr_value;
attr_value.uint32 = attribute_value;
OTF2_AttributeList_AddAttribute( attribute_list, attribute_id,
    OTF2_TYPE_UINT32, attr_value );
...
```

Then call the routine to write the event and pass the attribute list. The additional attributes are added to the event record and will be appended when reading the event later on. Please note: All attributes in the list will be added to event record. So make sure that there are only those attributes in the attribute list that you actually like to write. Please note: After writing the event record all attributes are removed from the attribute list. So the attribute list is empty again. If you want to write identical attributes to multiple events you have to add them each time new.

```
OTF2_EvtWriter_WriteEnter( ..., attribute_list, ... );
```

C.125 OTF2 callbacks

Modules

- Controlling OTF2 flush behavior in writing mode
- Memory pooling for OTF2
- Operating OTF2 in an collective context
- Operating OTF2 in a multi-threads context

C.126 Controlling OTF2 flush behavior in writing mode

Data Structures

- struct OTF2_FlushCallbacks

  Structure holding the flush callbacks.

Typedefs

- typedef OTF2_TimeStamp(∗ OTF2_PostFlushCallback)(void ∗userData, OTF2_-FileType fileType, OTF2_LocationRef location)

  Definition for the post flush callback.
- typedef OTF2_FlushType(∗ OTF2_PreFlushCallback)(void ∗userData, OTF2_-FileType fileType, OTF2_LocationRef location, void ∗callerData, bool final)

  Definition for the pre flush callback.

C.126.1 Detailed Description

The flushing behavior from OTF2 can be controlled via callbacks. Calling *OTF2_-Archive_SetFlushCallbacks* is mandatory when writing and erroneous when reading an archive.

The pre-flush callback decides whether an flush should actually happen. When missing, the default is not to flush any data for event writers, all others will flush there data by default.

The post-flush callback is used to decide whether an buffer flush record should be written after the flush finished. This only applies to event writers.

C.126.2 Typedef Documentation

C.126.2.1 typedef OTF2_TimeStamp(∗ OTF2_PostFlushCallback)(void ∗userData, OTF2_FileType fileType, OTF2_LocationRef location)

Definition for the post flush callback.

This callback is triggered right after flushing the recorded data into file when running out of memory. The main function of this callback is to provide a timestamp for the end of flushing data into a file. So an according record can be written correctly.

Parameters

userData	Data passed to the call *OTF2_Archive_SetFlushCallbacks*.
fileType	The file type for which the flush has happened.
location	The location ID of the writer for which the flush has happened (for file types without an ID this is *OTF2_UNDEFINED_LOCATION*).

Returns

Returns a timestamp for the end of flushing data into a file.

C.126.2.2 typedef OTF2_FlushType(∗ OTF2_PreFlushCallback)(void ∗userData, OTF2_FileType fileType, OTF2_LocationRef location, void ∗callerData, bool final)

Definition for the pre flush callback.

This callback is triggered right before flushing the recorded data into file when running out of memory.

Parameters

userData	Data passed to the call *OTF2_Archive_SetFlushCallbacks*.
fileType	The type of file for what this buffer holds data.
location	The location id for what this buffer holds data. This is only valid for files of type *OTF2_FILETYPE_LOCAL_DEFS* or *OTF2_FILETYPE_EVENTS*. For other files this is *OTF2_UNDEFINED_LOCATION*. A special case exists for files of type *OTF2_FILETYPE_EVENTS* in writing mode. The location ID may still be *OTF2_UNDEFINED_LOCATION*. In this case if the application wants to write the data from the buffer into the file, the application needs to provide a valid location ID via a call to *OTF2_EvtWriter_SetLocationID*() and utilizing the `callerData` argument.
callerData	Depending of the fileType, this can be an *OTF2_EvtWriter*, *OTF2_GlobalDefWriter*, *OTF2_DefWriter*.
final	Indicates whether this is the final flush when closing the writer objects.

Returns

Returns *OTF2_FLUSH* or *OTF2_NO_FLUSH*.

C.127 Memory pooling for OTF2

Data Structures

- struct OTF2_MemoryCallbacks

  Structure holding the memory callbacks.

Typedefs

- typedef void ∗(∗ OTF2_MemoryAllocate)(void ∗userData, OTF2_FileType fileType, OTF2_LocationRef location, void ∗∗perBufferData, uint64_t chunkSize)

  Function pointer for allocating memory for chunks.
- typedef void(∗ OTF2_MemoryFreeAll)(void ∗userData, OTF2_FileType fileType, OTF2_LocationRef location, void ∗∗perBufferData, bool final)

  Function pointer to release all allocated chunks.

C.127.1 Detailed Description

It is possible to provide memory for the record chunks to OTF2 via this callback interface. It is only used for writing. The default memory pool has a size of 128 MiB per writer.

Note that these callbacks must be thread safe. They are not protected by the locking callbacks.

C.127.2 Typedef Documentation

C.127.2.1 typedef void∗(∗ OTF2_MemoryAllocate)(void ∗userData, OTF2_FileType fileType, OTF2_LocationRef location, void ∗∗perBufferData, uint64_t chunkSize)

Function pointer for allocating memory for chunks.

Please note: Do not use this feature if you do not really understand it. The OTF2 library is not able to do any kind of checks to validate if your memory management works properly. If you do not use it correctly OTF2's behavior is undefined including dead locks and all that nasty stuff.

This function must return a pointer to a valid allocated memory location (just like malloc). This memory location must be of exact same size as the parameter 'chunk-Size' provided with OTF2_Archive_Open().

Parameters

userData	Data passed to the call *OTF2_Archive_SetMemoryCallbacks*.
fileType	The file type for which the chunk is requested.
location	The location ID of the writer for which the flush has happened (for file types without an ID this is *OTF2_UNDEFINED_LOCATION*).
perBuffer-Data	A writable pointer to store callee data. For the first call this will be `NULL`.
chunkSize	The size of the requested chunk.

Returns

Returns a the allocated memory on success, `NULL` if an error occurs.

C.127.2.2 typedef void(∗ OTF2_MemoryFreeAll)(void ∗userData, OTF2_FileType fileType, OTF2_LocationRef location, void ∗∗perBufferData, bool final)

Function pointer to release all allocated chunks.

Please note: Do not use this feature if you do not really understand it. The OTF2 library is not able to do any kind of checks to validate if your memory management works properly. If you do not use it correctly OTF2's behavior is undefined including dead locks and all that nasty stuff.

This function must free all those memory locations that were allocated for a buffer handle with the according allocate function. Please note: This is different from a posix free(). You must free _all_ memory locations for that were allocated for exactly this buffer handle.

Parameters

userData	Data passed to the call *OTF2_Archive_SetMemoryCallbacks*.
fileType	The file type for which free is requested.
location	The location ID of the writer for which the flush has happened (for file types without an ID this is *OTF2_UNDEFINED_LOCATION*).
perBuffer-Data	A writable pointer to store callee data. For the first call this will be `NULL`.
final	Indicates whether this is the final free when closing the writer objects. `perBufferData` should be handled than.

C.128 Operating OTF2 in an collective context

Data Structures

- struct OTF2_CollectiveCallbacks

  Struct which holds all collective callbacks.

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Barrier)(void ∗userData, OTF2_CollectiveContext ∗commContext)

  Performs an barrier collective on the given communication context.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Bcast)(void ∗userData, OTF2_CollectiveContext ∗commContext, void ∗data, uint32_t numberElements, OTF2_Type type, uint32_t root)

  Performs an broadcast collective on the given communication context.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_CreateLocalComm)(void ∗userData, OTF2_CollectiveContext ∗∗localCommContext, OTF2_CollectiveContext ∗globalCommContext, uint32_t globalRank, uint32_t globalSize, uint32_t localRank, uint32_t localSize, uint32_t fileNumber, uint32_t numberOfFiles)

  Create a new disjoint partitioning of the the globalCommContext communication context. numberOfFiles denotes the number of the partitions. fileNumber denotes in which of the partitions this OTF2_Archive should belong. localSize is the size of this partition and localRank the rank of this OTF2_Archive in the partition.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_FreeLocalComm)(void ∗userData, OTF2_CollectiveContext ∗localCommContext)

Destroys the communication context previous created by the OTF2_Collectives_-CreateLocalComm callback.

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Gather)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, void ∗outData, uint32_t numberElements, OTF2_Type type, uint32_t root)

  Performs an gather collective on the given communication context where each ranks contribute the same number of elements. outData is only valid at rank root.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Gatherv)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, uint32_t inElements, void ∗outData, const uint32_t ∗outElements, OTF2_Type type, uint32_t root)

  Performs an gather collective on the given communication context where each ranks contribute different number of elements. outData and outElements are only valid at rank root.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_GetRank)(void ∗userData, OTF2_CollectiveContext ∗commContext, uint32_t ∗rank)

  Returns the rank of this OTF2_Archive objects in this communication context. A number between 0 and one less of the size of the communication context.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_GetSize)(void ∗userData, OTF2_CollectiveContext ∗commContext, uint32_t ∗size)

  Returns the number of OTF2_Archive objects operating in this communication context.
- typedef void(∗ OTF2_Collectives_Release)(void ∗userData, OTF2_CollectiveContext ∗globalCommContext, OTF2_CollectiveContext ∗localCommContext)

  Optionally called in OTF2_Archive_Close or OTF2_Reader_Close respectively.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Scatter)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, void ∗outData, uint32_t numberElements, OTF2_Type type, uint32_t root)

  Performs an scatter collective on the given communication context where each ranks contribute the same number of elements. inData is only valid at rank root.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Scatterv)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, const uint32_-t ∗inElements, void ∗outData, uint32_t outElements, OTF2_Type type, uint32_-t root)

  Performs an scatter collective on the given communication context where each ranks contribute different number of elements. inData and inElements are only valid at rank root.

C.128.1 Detailed Description

To operate multiple OTF2_Archive objects in an collective context, the following callbacks need to be implemented. These are mandatory, when writing an trace file with multiple OTF2_Archive objects. For reading a set of serial callbacks are provided (See *OTF2_Archive_SetSerialCollectiveCallbacks* and *OTF2_Reader_-SetSerialCollectiveCallbacks*). The struct OTF2_CollectiveContext needs to be declared too.

Only *OTF2_Type* of the integer and floating point category need to be considered as values when the callbacks are called.

Except for the *OTF2_Collectives_GetSize* and *OTF2_Collectives_GetRank* callbacks, the return value must always be the same for all participating tasks. In particular all calls should either return *OTF2_CALLBACK_SUCCESS* or !*OTF2_-CALLBACK_SUCCESS*, but it is undefined, if some of the calls return *OTF2_-CALLBACK_SUCCESS* and other !*OTF2_CALLBACK_SUCCESS*.

The *OTF2_Collectives_CreateLocalComm* and *OTF2_Collectives_FreeLocalComm* are ignored when writing and optional when reading, but than both are mandatory. These are used to created the same local communication context as was given at writing time, if possible.

On the contrary the *localCommContext* to *OTF2_Archive_SetCollectiveCallbacks* is ignored when reading and optional (i.e., not NULL) when writing. It determines the number of files to use when the SION substrate is used. these *localCommContext* must be an disjoint partitioning of the used *globalCommContext* than.

The *OTF2_Collectives_Release* is optional and will be called as one of the last actions before the OTF2_Archive or the OTF2_Reader will be closed.

If any collective callback returns !*OTF2_CALLBACK_SUCCESS*, than OTF2 returns to the caller the error *OTF2_ERROR_COLLECTIVE_CALLBACK*.

C.128.2 Typedef Documentation

C.128.2.1 typedef OTF2_CallbackCode(∗ OTF2_Collectives_Barrier)(void ∗userData, OTF2_CollectiveContext ∗commContext)

Performs an barrier collective on the given communication context.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.2 typedef OTF2_CallbackCode(∗ OTF2_Collectives_Bcast)(void ∗userData, OTF2_CollectiveContext ∗commContext, void ∗data, uint32_t numberElements, OTF2_Type type, uint32_t root)

Performs an broadcast collective on the given communication context.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.3 typedef OTF2_CallbackCode(∗ OTF2_Collectives_-CreateLocalComm)(void ∗userData, OTF2_CollectiveContext ∗∗localCommContext, OTF2_CollectiveContext ∗globalCommContext, uint32_t globalRank, uint32_t globalSize, uint32_t localRank, uint32_t localSize, uint32_t fileNumber, uint32_t numberOfFiles)

Create a new disjoint partitioning of the the *globalCommContext* communication context. *numberOfFiles* denotes the number of the partitions. *fileNumber* denotes in which of the partitions this OTF2_Archive should belong. *localSize* is the size of this partition and *localRank* the rank of this OTF2_Archive in the partition.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.4 typedef OTF2_CallbackCode(∗ OTF2_Collectives_-FreeLocalComm)(void ∗userData, OTF2_CollectiveContext ∗localCommContext)

Destroys the communication context previous created by the *OTF2_Collectives_-CreateLocalComm* callback.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.5 typedef OTF2_CallbackCode(∗ OTF2_Collectives_Gather)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, void ∗outData, uint32_t numberElements, OTF2_Type type, uint32_t root)

Performs an gather collective on the given communication context where each ranks contribute the same number of elements. *outData* is only valid at rank *root*.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.6 typedef OTF2_CallbackCode(∗ OTF2_Collectives_Gatherv)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, uint32_t inElements, void ∗outData, const uint32_t ∗outElements, OTF2_Type type, uint32_t root)

Performs an gather collective on the given communication context where each ranks contribute different number of elements. *outData* and *outElements* are only valid at rank *root*.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.7 typedef OTF2_CallbackCode(∗ OTF2_Collectives_GetRank)(void ∗userData, OTF2_CollectiveContext ∗commContext, uint32_t ∗rank)

Returns the rank of this OTF2_Archive objects in this communication context. A number between 0 and one less of the size of the communication context.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.8 typedef OTF2_CallbackCode($*$ OTF2_Collectives_GetSize)(void $*$userData, OTF2_CollectiveContext $*$commContext, uint32_t $*$size)

Returns the number of OTF2_Archive objects operating in this communication context.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.9 typedef void($*$ OTF2_Collectives_Release)(void $*$userData, OTF2_-CollectiveContext $*$globalCommContext, OTF2_CollectiveContext $*$localCommContext)

Optionally called in *OTF2_Archive_Close* or *OTF2_Reader_Close* respectively.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.10 typedef OTF2_CallbackCode($*$ OTF2_Collectives_Scatter)(void $*$userData, OTF2_CollectiveContext $*$commContext, const void $*$inData, void $*$outData, uint32_t numberElements, OTF2_Type type, uint32_t root)

Performs an scatter collective on the given communication context where each ranks contribute the same number of elements. *inData* is only valid at rank *root*.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.128.2.11 typedef OTF2_CallbackCode(∗ OTF2_Collectives_Scatterv)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, const uint32_t ∗inElements, void ∗outData, uint32_t outElements, OTF2_Type type, uint32_t root)

Performs an scatter collective on the given communication context where each ranks contribute different number of elements. *inData* and *inElements* are only valid at rank *root*.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.129 Operating OTF2 in a multi-threads context

Data Structures

- struct OTF2_LockingCallbacks

  Struct which holds all collective callbacks.

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_Locking_Create)(void ∗userData, OTF2_-Lock ∗lock)

  Creates a new locking object.
- typedef OTF2_CallbackCode(∗ OTF2_Locking_Destroy)(void ∗userData, OTF2_Lock lock)

  Destroys a locking object.
- typedef OTF2_CallbackCode(∗ OTF2_Locking_Lock)(void ∗userData, OTF2_-Lock lock)

  Locks a locking object.
- typedef void(∗ OTF2_Locking_Release)(void ∗userData)

  Optionally called in OTF2_Archive_Close or OTF2_Reader_Close respectively.
- typedef OTF2_CallbackCode(∗ OTF2_Locking_Unlock)(void ∗userData, OTF2_Lock lock)

  Unlocks a locking object.

C.129.1 Detailed Description

The OTF2 objects *OTF2_Archive* and *OTF2_Reader* including all derived objects from these are by default not thread safe. That means it is undefined behavior to operate any of these objects concurrently by multiple threads. Note that two independent *OTF2_Archive* or *OTF2_Reader* objects and there derived objects can be operatated by multiple threads concurrently though.

It is necessary to register the following locking callbacks to make a *OTF2_Archive* and *OTF2_Reader* and their derived objects thread safe. The created locking objects should have normal locking semantics, no recursive or nesting capability is needed.

OTF2 provides two locking callbacks implementations for Pthread and OpenMP. See the header files *otf2/OTF2_Pthread_Locks.h* and *otf2/OTF2_OpenMP_Locks.h*. For a usage of these headers have a look into the installed usage examples **`otf2_pthread_writer_example.c`** and **`otf2_openmp_writer_example.c`**.

If any locking callback returns !*OTF2_CALLBACK_SUCCESS*, than OTF2 returns to the caller the error *OTF2_ERROR_LOCKING_CALLBACK*.

C.129.2 Typedef Documentation

C.129.2.1 typedef OTF2_CallbackCode(∗ OTF2_Locking_Create)(void ∗userData, OTF2_Lock ∗lock)

Creates a new locking object.

Parameters

userData	Value from paramter *userData* passed to *OTF2_Archive_SetLockingCallbacks* or *OTF2_Reader_SetLockingCallbacks* respectively.
lock[out]	Reference to pointer to new lock object.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.129.2.2 typedef OTF2_CallbackCode(∗ OTF2_Locking_Destroy)(void ∗userData, OTF2_Lock lock)

Destroys a locking object.

Parameters

userData	Value from paramter *userData* passed to *OTF2_Archive_-SetLockingCallbacks* or *OTF2_Reader_SetLockingCallbacks* respectively.
lock	Lock object to destroy.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.129.2.3 typedef OTF2_CallbackCode(∗ OTF2_Locking_Lock)(void ∗userData, OTF2_Lock lock)

Locks a locking object.

Parameters

userData	Value from paramter *userData* passed to *OTF2_Archive_-SetLockingCallbacks* or *OTF2_Reader_SetLockingCallbacks* respectively.
lock	Lock object to lock.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.129.2.4 typedef void(∗ OTF2_Locking_Release)(void ∗userData)

Optionally called in *OTF2_Archive_Close* or *OTF2_Reader_Close* respectively.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.129.2.5 typedef OTF2_CallbackCode(∗ OTF2_Locking_Unlock)(void ∗userData, OTF2_Lock lock)

Unlocks a locking object.

Parameters

userData	Value from paramter *userData* passed to *OTF2_Archive_-SetLockingCallbacks* or *OTF2_Reader_SetLockingCallbacks* respectively.
lock	Lock object to unlock.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_ERROR*.

C.130 Usage in reading mode - MPI example

This is a example of how to use the OTF2 reading interface with MPI. It shows how to define and register callbacks and how to use the provided MPI collective callbacks to read all events of a given OTF2 archive in parallel. This example is available as source code in the file **otf2_mpi_reader_example.c**.

We start with inclusion of some standard headers.

```
#include <stdlib.h>
#include <stdio.h>
#include <inttypes.h>
```

And than include the MPI and OTF2 header.

```
#include <mpi.h>
```

C.130 Usage in reading mode - MPI example

```
#include <otf2/otf2.h>
```

Now prepare the inclusion of the <otf2/OTF2_MPI_Collectives.h> header. As it is an header-only interface, it needs some information about the used MPI environment. In particular the MPI datatypes which match the C99 types `uint64_t` and `int64_t`. In case you have an MPI 3.0 conforming MPI implementation you can skip this. If not, provide #define's for the following macros prior the #include statement. In this example, we asume an LP64 platform.

```
#if MPI_VERSION < 3
#define OTF2_MPI_UINT64_T MPI_UNSIGNED_LONG
#define OTF2_MPI_INT64_T  MPI_LONG
#endif
```

After this preparatory step, we can include the <otf2/OTF2_MPI_Collectives.h> header.

```
#include <otf2/OTF2_MPI_Collectives.h>
```

The following section until desribing `main` is the same as in the Usage in reading mode - a simple example.

Define an event callback for entering and leaving a region.

```
static OTF2_CallbackCode
Enter_print( OTF2_LocationRef    location,
             OTF2_TimeStamp      time,
             void*               userData,
             OTF2_AttributeList* attributes,
             OTF2_RegionRef      region )
{
    printf( "Entering region %u at location %" PRIu64 " at time %" PRIu64 ".\n",
            region, location, time );

    return OTF2_CALLBACK_SUCCESS;
}

static OTF2_CallbackCode
Leave_print( OTF2_LocationRef    location,
             OTF2_TimeStamp      time,
             void*               userData,
             OTF2_AttributeList* attributes,
             OTF2_RegionRef      region )
{
    printf( "Leaving region %u at location %" PRIu64 " at time %" PRIu64 ".\n",
            region, location, time );

    return OTF2_CALLBACK_SUCCESS;
}
```

The global definition file provides all location IDs that are included in the OTF2 trace archive. When reading the global defintions these location IDs must be collected and stored by the user. Probably, the easiest way to do that is to use a C++ container.

```
struct vector
{
    size_t   capacity;
    size_t   size;
    uint64_t members[];
};

static OTF2_CallbackCode
GlobDefLocation_Register( void*                 userData,
                          OTF2_LocationRef      location,
                          OTF2_StringRef        name,
                          OTF2_LocationType     locationType,
                          uint64_t              numberOfEvents,
                          OTF2_LocationGroupRef locationGroup )
{
    struct vector* locations = userData;

    if ( locations->size == locations->capacity )
    {
        return OTF2_CALLBACK_INTERRUPT;
    }

    locations->members[ locations->size++ ] = location;

    return OTF2_CALLBACK_SUCCESS;
}
```

Now everything is prepared to begin with the main program.

```
int
main( int    argc,
      char** argv )
{
```

First initialize the MPI envirionment and query the size and rank.

```
    MPI_Init( &argc, &argv );
    int size;
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    int rank;
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
```

Create a new reader handle. The path to the OTF2 anchor file must be provided as argument.

```
    OTF2_Reader* reader = OTF2_Reader_Open( "ArchivePath/ArchiveName.otf2" );
```

Now we provide the OTF2 reader object the MPI collectives.

```
    OTF2_MPI_Reader_SetCollectiveCallbacks( reader, MPI_COMM_WORLD );
```

OTF2 provides an API to query the number of locations prior reading the global definitions. We use this to pre-allocate the storage for all locations.

```
    uint64_t number_of_locations;
    OTF2_Reader_GetNumberOfLocations( reader,
                                      &number_of_locations );
    struct vector* locations = malloc( sizeof( *locations )
                                       + number_of_locations
                                       * sizeof( *locations->members ) );
    locations->capacity = number_of_locations;
    locations->size     = 0;
```

All ranks need to read the global definitions to know the list of locations in the trace. Get a global definition reader with the above reader handle as argument.

```
    OTF2_GlobalDefReader* global_def_reader = OTF2_Reader_GetGlobalDefReader( rea
      der );
```

Register the above defined global definition callbacks. All other definition callbacks will be deactivated. And instruct the reader to pass the *locations* object to each call of the callbacks.

```
    OTF2_GlobalDefReaderCallbacks* global_def_callbacks =
      OTF2_GlobalDefReaderCallbacks_New();
    OTF2_GlobalDefReaderCallbacks_SetLocationCallback( global_def_callbacks,
                                                       &GlobDefLocation_Register
      );
    OTF2_Reader_RegisterGlobalDefCallbacks( reader,
                                            global_def_reader,
                                            global_def_callbacks,
                                            locations );
    OTF2_GlobalDefReaderCallbacks_Delete( global_def_callbacks );
```

Read all global definitions. Everytime a location definition is read, the previosly registered callback is triggered. In `definitions_read` the number of read definitions is returned.

```
    uint64_t definitions_read = 0;
    OTF2_Reader_ReadAllGlobalDefinitions( reader,
                                          global_def_reader,
                                          &definitions_read );
```

After reading all global definitions all location IDs are stored in the generic container `ListOfLocations`. After that, the locations that are suppossed to be read are selected. We distribute the locations round-robin to all ranks in `MPI_COMM_WORLD`. We need also to remember, whether this rank actually reads any locations.

```
uint64_t number_of_locations_to_read = 0;
for ( size_t i = 0; i < locations->size; i++ )
{
    if ( locations->members[ i ] % size != rank )
    {
        continue;
    }
    number_of_locations_to_read++;
    OTF2_Reader_SelectLocation( reader, locations->members[ i ] );
}
```

When the locations are selected the according event and definition files can be opened. Note that the local definition files are optional, thus we need to remember the success of this call.

```
bool successful_open_def_files =
    OTF2_Reader_OpenDefFiles( reader ) == OTF2_SUCCESS;
OTF2_Reader_OpenEvtFiles( reader );
```

When the files are opened the event and defintion reader handle can be requested. We distribute the locations round-robin to all ranks in `MPI_COMM_WORLD`. To apply mapping tables stored in the local definitions, the local defintions must be read. Though the existence of these files are optional. The call to *OTF2_Reader_GetEvtReader* is mandatory, but the result is unused.

```
for ( size_t i = 0; i < locations->size; i++ )
{
    if ( locations->members[ i ] % size != rank )
    {
        continue;
    }

    if ( successful_open_def_files )
    {
        OTF2_DefReader* def_reader =
            OTF2_Reader_GetDefReader( reader, locations->members[ i ] );
        if ( def_reader )
        {
            uint64_t def_reads = 0;
            OTF2_Reader_ReadAllLocalDefinitions( reader,
                                                 def_reader,
                                                 &def_reads );
            OTF2_Reader_CloseDefReader( reader, def_reader );
        }
    }
```

```
        OTF2_EvtReader* evt_reader =
            OTF2_Reader_GetEvtReader( reader, locations->members[ i ] );
    }
```

The definition files can now be closed, if it was successfully opened in the first place.

```
    if ( successful_open_def_files )
    {
        OTF2_Reader_CloseDefFiles( reader );
    }
```

Only these ranks which actually read events for locations, can now open a new global event reader. This global reader automatically contains all previously opened local event readers.

```
    if ( number_of_locations_to_read > 0 )
    {
        OTF2_GlobalEvtReader* global_evt_reader = OTF2_Reader_GetGlobalEvtReader(
       reader );
```

Register the above defined global event callbacks. All other global event callbacks will be deactivated.

```
        OTF2_GlobalEvtReaderCallbacks* event_callbacks =
      OTF2_GlobalEvtReaderCallbacks_New();
        OTF2_GlobalEvtReaderCallbacks_SetEnterCallback( event_callbacks,
                                                        &Enter_print );
        OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback( event_callbacks,
                                                        &Leave_print );
        OTF2_Reader_RegisterGlobalEvtCallbacks( reader,
                                                global_evt_reader,
                                                event_callbacks,
                                                NULL );
        OTF2_GlobalEvtReaderCallbacks_Delete( event_callbacks );
```

Read all events in the OTF2 archive. The events are automatically ordered by the time they occured in the trace. Everytime an enter or leave event is read, the previously registered callbacks are triggered. In `events_read` the number of read events is returned.

```
        uint64_t events_read = 0;
        OTF2_Reader_ReadAllGlobalEvents( reader,
                                         global_evt_reader,
                                         &events_read );
```

The global event reader can now be closed and the event files too.

```
        OTF2_Reader_CloseGlobalEvtReader( reader, global_evt_reader );
```

As the call to *OTF2_Reader_CloseEvtFiles* is an collective operation all ranks need to call this, not only those which read events.

```
    }
    OTF2_Reader_CloseEvtFiles( reader );
```

At the end, close the reader and exit. All opened event and definition readers are closed automatically. Free resources and finalize the MPI envirionment.

```
    OTF2_Reader_Close( reader );

    free( locations );

    MPI_Finalize();

    return EXIT_SUCCESS;
}
```

To compile your program use a command like the following. Note that we need to activate the C99 standard explicitly for GCC.

```
    mpicc -std=c99 `otf2-config --cflags` \
            -c otf2_mpi_reader_example.c \
            -o otf2_mpi_reader_example.o
```

Now you can link your program with:

```
    mpicc otf2_mpi_reader_example.o \
            `otf2-config --ldflags` \
            `otf2-config --libs` \
            -o otf2_mpi_reader_example
```

C.131 Usage in writing mode - MPI example

This is a short example of how to use the OTF2 writing interface with MPI. This example is available as source code in the file **`otf2_mpi_writer_example.c`**.

We start with inclusion of some standard headers.

```
#include <stdlib.h>
#include <stdio.h>
#include <inttypes.h>
```

And than include the MPI and OTF2 header.

```
#include <mpi.h>

#include <otf2/otf2.h>
```

Now prepare the inclusion of the <otf2/OTF2_MPI_Collectives.h> header. As it is an header-only interface, it needs some information about the used MPI environment. In particular the MPI datatypes which match the C99 types `uint64_t` and `int64_t`. In case you have an MPI 3.0 conforming MPI implementation you can skip this. If not, provide #define's for the following macros prior the #include statement. In this example, we asume an LP64 platform.

```
#if MPI_VERSION < 3
#define OTF2_MPI_UINT64_T MPI_UNSIGNED_LONG
#define OTF2_MPI_INT64_T  MPI_LONG
#endif
```

After this preparatory step, we can include the <otf2/OTF2_MPI_Collectives.h> header.

```
#include <otf2/OTF2_MPI_Collectives.h>
```

We use `MPI_Wtime` to get timestamps for our events but need to convert the seconds to an integral value. We use a nano second resolution.

```
static OTF2_TimeStamp
get_time( void )
{
    double t = MPI_Wtime() * 1e9;
    return ( uint64_t )t;
}
```

Define a pre and post flush callback. If no memory is left in OTF2's internal memory buffer or the writer handle is closed a memory buffer flushing routine is triggered. The pre flush callback is triggered right before a buffer flush. It needs to return either OTF2_FLUSH to flush the recorded data to a file or OTF2_NO_-FLUSH to supress flushing data to a file. The post flush callback is triggered right after a memory buffer flush. It has to return a current timestamp which is recorded to mark the time spend in a buffer flush. The callbacks are passed via a struct to OTF2.

```
static OTF2_FlushType
pre_flush( void*            userData,
           OTF2_FileType    fileType,
           OTF2_LocationRef location,
           void*            callerData,
           bool             final )
{
    return OTF2_FLUSH;
}

static OTF2_TimeStamp
post_flush( void*            userData,
            OTF2_FileType    fileType,
            OTF2_LocationRef location )
{
    return get_time();
}

static OTF2_FlushCallbacks flush_callbacks =
{
    .otf2_pre_flush  = pre_flush,
    .otf2_post_flush = post_flush
};
```

Now everything is prepared to begin with the main program.

```
int
main( int    argc,
      char** argv )
{
```

First initialize the MPI envirionment and query the size and rank.

```
    MPI_Init( &argc, &argv );
    int size;
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    int rank;
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
```

Create new archive handle.

```
    OTF2_Archive* archive = OTF2_Archive_Open( "ArchivePath",
                                               "ArchiveName",
                                               OTF2_FILEMODE_WRITE,
                                               1024 * 1024 /* event chunk size */
      ,
                                               4 * 1024 * 1024 /* def chunk size
      */,
                                               OTF2_SUBSTRATE_POSIX,
                                               OTF2_COMPRESSION_NONE );
```

Set the previously defined flush callbacks.

```
    OTF2_Archive_SetFlushCallbacks( archive, &flush_callbacks, NULL );
```

Now we provide the OTF2 archive object the MPI collectives. As all ranks in MPI_COMM_WORLD write into the archive, we use this communicator as the global one. We set the local communicator to MPI_COMM_NULL, as we don't care about file optimization here.

```
    OTF2_MPI_Archive_SetCollectiveCallbacks( archive,
                                             MPI_COMM_WORLD,
                                             MPI_COMM_NULL );
```

Now we can create the event files. Though physical files aren't created yet.

```
    OTF2_Archive_OpenEvtFiles( archive );
```

Each rank now requests an event writer with its rank number as the location id.

```
    OTF2_EvtWriter* evt_writer = OTF2_Archive_GetEvtWriter( archive,
                                                            rank );
```

We note the start time in each rank, this is later used to determine the global epoch.

```
    uint64_t epoch_start = get_time();
```

Write an enter and a leave record for region 0 to the local event writer.

```
    OTF2_EvtWriter_Enter( evt_writer,
                          NULL,
                          get_time(),
                          0 /* region */ );
```

We also record an MPI_Barrier in the trace. For this we generate an event before we do the MPI call.

```
    OTF2_EvtWriter_MpiCollectiveBegin( evt_writer,
                                       NULL,
                                       get_time() );
```

Now we can do the MPI_Barrier call.

```
    MPI_Barrier( MPI_COMM_WORLD );
```

After we passed the `MPI_Barrier`. we can note the end of the collective operation inside the event stream.

```
OTF2_EvtWriter_MpiCollectiveEnd( evt_writer,
                                 NULL,
                                 get_time(),
                                 OTF2_COLLECTIVE_OP_BARRIER,
                                 0 /* communicator */,
                                 OTF2_UNDEFINED_UINT32 /* root */,
                                 0 /* bytes provided */,
                                 0 /* bytes obtained */ );
```

Finally we leave the region again with the leave region.

```
OTF2_EvtWriter_Leave( evt_writer,
                      NULL,
                      get_time(),
                      0 /* region */ );
```

The event recording is now done, note the end time in each rank.

```
uint64_t epoch_end = get_time();
```

Now close the event writer, before closing the event files collectivly.

```
OTF2_Archive_CloseEvtWriter( archive, evt_writer );
```

After we wrote all of the events we close the event files again.

```
OTF2_Archive_CloseEvtFiles( archive );
```

We now collect all of the `epoch_start` and `epoch_end` timestamps by calculating the minimum and maximize and provide these to the root rank.

```
uint64_t global_epoch_start;
MPI_Reduce( &epoch_start,
            &global_epoch_start,
            1, OTF2_MPI_UINT64_T, MPI_MIN,
            0, MPI_COMM_WORLD );

uint64_t global_epoch_end;
MPI_Reduce( &epoch_end,
            &global_epoch_end,
            1, OTF2_MPI_UINT64_T, MPI_MAX,
            0, MPI_COMM_WORLD );
```

C.131 Usage in writing mode - MPI example

Only the root rank will write the global definitions, thus only he requests an writer object from the archive.

```
    if ( 0 == rank )
    {
        OTF2_GlobalDefWriter* global_def_writer =
      OTF2_Archive_GetGlobalDefWriter( archive );
```

We need to define the clock used for this trace and the overall timestamp range.

```
        OTF2_GlobalDefWriter_WriteClockProperties( global_def_writer,
                                                   1000000000,
                                                   global_epoch_start,
                                                   global_epoch_end - global_epoc
      h_start + 1 );
```

Now we can start writing the referenced definitions, starting with the strings.

```
        OTF2_GlobalDefWriter_WriteString( global_def_writer, 0, "" );
        OTF2_GlobalDefWriter_WriteString( global_def_writer, 1, "Master Thread" )
      ;
        OTF2_GlobalDefWriter_WriteString( global_def_writer, 2, "MPI_Barrier" );
        OTF2_GlobalDefWriter_WriteString( global_def_writer, 3, "PMPI_Barrier" );

        OTF2_GlobalDefWriter_WriteString( global_def_writer, 4, "barrier" );
        OTF2_GlobalDefWriter_WriteString( global_def_writer, 5, "MyHost" );
        OTF2_GlobalDefWriter_WriteString( global_def_writer, 6, "node" );
        OTF2_GlobalDefWriter_WriteString( global_def_writer, 7, "MPI" );
        OTF2_GlobalDefWriter_WriteString( global_def_writer, 8, "MPI_COMM_WORLD"
      );
```

Write definition for the code region which was just entered and left to the global definition writer.

```
        OTF2_GlobalDefWriter_WriteRegion( global_def_writer,
                                          0 /* id */,
                                          2 /* region name  */,
                                          3 /* alternative name */,
                                          4 /* description */,
                                          OTF2_REGION_ROLE_BARRIER,
                                          OTF2_PARADIGM_MPI,
                                          OTF2_REGION_FLAG_NONE,
                                          7 /* source file */,
                                          0 /* begin lno */,
                                          0 /* end lno */ );
```

Write the system tree to the global definition writer.

```
    OTF2_GlobalDefWriter_WriteSystemTreeNode( global_def_writer,
                                              0 /* id */,
                                              5 /* name */,
                                              6 /* class */,

  OTF2_UNDEFINED_SYSTEM_TREE_NODE /* parent */ );
```

For each rank we define a new loation group and one location. We provide also a unique string for each location group.

```
    for ( int r = 0; r < size; r++ )
    {
        char process_name[ 32 ];
        sprintf( process_name, "MPI Rank %d", r );
        OTF2_GlobalDefWriter_WriteString( global_def_writer,
                                          9 + r,
                                          process_name );

        OTF2_GlobalDefWriter_WriteLocationGroup( global_def_writer,
                                                 r /* id */,
                                                 9 + r /* name */,

  OTF2_LOCATION_GROUP_TYPE_PROCESS,
                                                 0 /* system tree */ );

        OTF2_GlobalDefWriter_WriteLocation( global_def_writer,
                                            r /* id */,
                                            1 /* name */,
                                            OTF2_LOCATION_TYPE_CPU_THREAD,
                                            4 /* # events */,
                                            r /* location group */ );
    }
```

The last step is to define the MPI communicator. This is a three-step process. First we define that this trace actually recorded in the MPI paradigm and enumerate all locations which participate in this paradigm. As we used the MPI ranks directly as the location id, the array with the locations is the identity.

```
    uint64_t comm_locations[ size ];
    for ( int r = 0; r < size; r++ )
    {
        comm_locations[ r ] = r;
    }
    OTF2_GlobalDefWriter_WriteGroup( global_def_writer,
                                     0 /* id */,
                                     7 /* name */,
                                     OTF2_GROUP_TYPE_COMM_LOCATIONS,
                                     OTF2_PARADIGM_MPI,
                                     OTF2_GROUP_FLAG_NONE,
                                     size,
                                     comm_locations );
```

Now we can define sub-groups of the previously defined list of communication. locations. For `MPI_COMM_WORLD` this is the whole group here. Note the these sub-groups are created by using indices into the list of communication locations, and not by enumrateing location ids again. But in this example the sub-group is the identity again.

```
        OTF2_GlobalDefWriter_WriteGroup( global_def_writer,
                                         1 /* id */,
                                         0 /* name */,
                                         OTF2_GROUP_TYPE_COMM_GROUP,
                                         OTF2_PARADIGM_MPI,
                                         OTF2_GROUP_FLAG_NONE,
                                         size,
                                         comm_locations );
```

Finally we can write the definition of the `MPI_COMM_WORLD` communcator. This finalizes the writing of the global definitions and we can also close the writer object.

```
        OTF2_GlobalDefWriter_WriteComm( global_def_writer,
                                        0 /* id */,
                                        8 /* name */,
                                        1 /* group */,
                                        OTF2_UNDEFINED_COMM /* parent */ );

        OTF2_Archive_CloseGlobalDefWriter( archive,
                                           global_def_writer );
    }
```

All the other ranks wait inside this barrier so that root can write the global definitions.

```
    MPI_Barrier( MPI_COMM_WORLD );
```

At the end, close the archive, finalize the MPI envirionment, and exit.

```
    OTF2_Archive_Close( archive );

    MPI_Finalize();

    return EXIT_SUCCESS;
}
```

To compile your program use a command like the following. Note that we need to activate the C99 standard explicitly for GCC.

```
    mpicc -std=c99 `otf2-config --cflags` \
            -c otf2_mpi_writer_example.c \
            -o otf2_mpi_writer_example.o
```

Now you can link your program with:

```
    mpicc otf2_mpi_writer_example.o \
            `otf2-config --ldflags` \
            `otf2-config --libs` \
            -o otf2_mpi_writer_example
```

C.132 Usage in reading mode - a simple example

This is a short example of how to use the OTF2 reading interface. It shows how to define and register callbacks and how to use the reader interface to read all events of a given OTF2 archive. This example is available as source code in the file **otf2_reader_example.c**.

First include the OTF2 header.

```
#include <otf2/otf2.h>
```

For this example some additional include statements are necessary.

```
#include <stdlib.h>
#include <stdio.h>
#include <inttypes.h>
```

Define an event callback for entering and leaving a region.

```
static OTF2_CallbackCode
Enter_print( OTF2_LocationRef    location,
             OTF2_TimeStamp      time,
             void*               userData,
             OTF2_AttributeList* attributes,
             OTF2_RegionRef      region )
{
    printf( "Entering region %u at location %" PRIu64 " at time %" PRIu64 ".\n",
            region, location, time );

    return OTF2_CALLBACK_SUCCESS;
}

static OTF2_CallbackCode
Leave_print( OTF2_LocationRef    location,
             OTF2_TimeStamp      time,
             void*               userData,
             OTF2_AttributeList* attributes,
             OTF2_RegionRef      region )
{
    printf( "Leaving region %u at location %" PRIu64 " at time %" PRIu64 ".\n",
            region, location, time );

    return OTF2_CALLBACK_SUCCESS;
}
```

The global definition file provides all location IDs that are included in the OTF2 trace archive. When reading the global defintions these location IDs must be collected and stored by the user. Probably, the easiest way to do that is to use a C++ container.

```
struct vector
{
    size_t   capacity;
    size_t   size;
    uint64_t members[];
};

static OTF2_CallbackCode
GlobDefLocation_Register( void*                 userData,
                          OTF2_LocationRef      location,
                          OTF2_StringRef        name,
                          OTF2_LocationType     locationType,
                          uint64_t              numberOfEvents,
                          OTF2_LocationGroupRef locationGroup )
{
    struct vector* locations = userData;

    if ( locations->size == locations->capacity )
    {
        return OTF2_CALLBACK_INTERRUPT;
    }

    locations->members[ locations->size++ ] = location;

    return OTF2_CALLBACK_SUCCESS;
}
```

Now everything is prepared to begin with the main program.

```
int
main( int    argc,
      char** argv )
{
```

Create a new reader handle. The path to the OTF2 anchor file must be provided as argument.

```
    OTF2_Reader* reader = OTF2_Reader_Open( "ArchivePath/ArchiveName.otf2" );
```

We will operate in an serial context.

```
OTF2_Reader_SetSerialCollectiveCallbacks( reader );
```

OTF2 provides an API to query the number of locations prior reading the global definitions. We use this to pre-allocate the storage for all locations.

```
uint64_t number_of_locations;
OTF2_Reader_GetNumberOfLocations( reader,
                                  &number_of_locations );
struct vector* locations = malloc( sizeof( *locations )
                                   + number_of_locations
                                   * sizeof( *locations->members ) );
locations->capacity = number_of_locations;
locations->size     = 0;
```

Get the global definition reader from the reader handle.

```
OTF2_GlobalDefReader* global_def_reader = OTF2_Reader_GetGlobalDefReader( rea
  der );
```

Register the above defined global definition callbacks. All other definition callbacks will be deactivated. And instruct the reader to pass the *locations* object to each call of the callbacks.

```
OTF2_GlobalDefReaderCallbacks* global_def_callbacks =
  OTF2_GlobalDefReaderCallbacks_New();
OTF2_GlobalDefReaderCallbacks_SetLocationCallback( global_def_callbacks,
                                                   &GlobDefLocation_Register
  );
OTF2_Reader_RegisterGlobalDefCallbacks( reader,
                                        global_def_reader,
                                        global_def_callbacks,
                                        locations );
OTF2_GlobalDefReaderCallbacks_Delete( global_def_callbacks );
```

Read all global definitions. Everytime a location definition is read, the previosly registered callback is triggered. In `definitions_read` the number of read definitions is returned.

```
uint64_t definitions_read = 0;
OTF2_Reader_ReadAllGlobalDefinitions( reader,
                                      global_def_reader,
                                      &definitions_read );
```

After reading all global definitions all location IDs are stored in the vector `locations`. After that, the locations that are suppossed to be read are selected. In this example all.

```
    for ( size_t i = 0; i < locations->size; i++ )
    {
        OTF2_Reader_SelectLocation( reader, locations->members[ i ] );
    }
```

When the locations are selected the according event and definition files can be opened. Note that the local definition files are optional, thus we need to remember the success of this call.

```
    bool successful_open_def_files =
        OTF2_Reader_OpenDefFiles( reader ) == OTF2_SUCCESS;
    OTF2_Reader_OpenEvtFiles( reader );
```

When the files are opened the event and defintion reader handle can be requested. In this example for all. To apply mapping tables stored in the local definitions, the local defintions must be read. Though the existence of these files are optional. The call to *OTF2_Reader_GetEvtReader* is mandatory, but the result is unused.

```
    for ( size_t i = 0; i < locations->size; i++ )
    {
        if ( successful_open_def_files )
        {
            OTF2_DefReader* def_reader =
                OTF2_Reader_GetDefReader( reader, locations->members[ i ] );
            if ( def_reader )
            {
                uint64_t def_reads = 0;
                OTF2_Reader_ReadAllLocalDefinitions( reader,
                                                     def_reader,
                                                     &def_reads );
                OTF2_Reader_CloseDefReader( reader, def_reader );
            }
        }
        OTF2_EvtReader* evt_reader =
            OTF2_Reader_GetEvtReader( reader, locations->members[ i ] );
    }
```

The definition files can now be closed, if it was successfully opened in the first place.

```
    if ( successful_open_def_files )
    {
        OTF2_Reader_CloseDefFiles( reader );
    }
```

Open a new global event reader. This global reader automatically contains all previously opened local event readers.

```
    OTF2_GlobalEvtReader* global_evt_reader = OTF2_Reader_GetGlobalEvtReader( rea
      der );
```

Register the above defined global event callbacks. All other global event callbacks will be deactivated.

```
    OTF2_GlobalEvtReaderCallbacks* event_callbacks =
      OTF2_GlobalEvtReaderCallbacks_New();
    OTF2_GlobalEvtReaderCallbacks_SetEnterCallback( event_callbacks,
                                                    &Enter_print );
    OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback( event_callbacks,
                                                    &Leave_print );
    OTF2_Reader_RegisterGlobalEvtCallbacks( reader,
                                            global_evt_reader,
                                            event_callbacks,
                                            NULL );
    OTF2_GlobalEvtReaderCallbacks_Delete( event_callbacks );
```

Read all events in the OTF2 archive. The events are automatically ordered by the time they occured in the trace. Everytime an enter or leave event is read, the previously registered callbacks are triggered. In `events_read` the number of read events is returned.

```
    uint64_t events_read = 0;
    OTF2_Reader_ReadAllGlobalEvents( reader,
                                     global_evt_reader,
                                     &events_read );
```

The global event reader can now be closed and the event files too.

```
    OTF2_Reader_CloseGlobalEvtReader( reader, global_evt_reader );
    OTF2_Reader_CloseEvtFiles( reader );
```

At the end, close the reader and exit. All opened event and definition readers are closed automatically.

```
    OTF2_Reader_Close( reader );

    free( locations );

    return EXIT_SUCCESS;
}
```

To compile your program use a command like the following. Note that we need to activate the C99 standard explicitly for GCC.

```
gcc -std=c99 `otf2-config --cflags` \
        -c otf2_reader_example.c \
        -o otf2_reader_example.o
```

Now you can link your program with:

```
gcc otf2_reader_example.o \
        `otf2-config --ldflags` \
        `otf2-config --libs` \
        -o otf2_reader_example
```

Appendix D

Data Structure Documentation

D.1 OTF2_AttributeValue Union Reference

Value container for an attributes.

`#include <otf2/OTF2_AttributeValue.h>`

Data Fields

- OTF2_AttributeRef attributeRef

  References a Attribute definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_ATTRIBUTE is available.
- OTF2_CallingContextRef callingContextRef

  References a CallingContext definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_CALLING_CONTEXT is available.
- OTF2_CommRef commRef

  References a Comm definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_COMM is available.
- float float32

  Arbitrary value of type float.
- double float64

  Arbitrary value of type double.
- OTF2_GroupRef groupRef

  References a Group definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_GROUP is available.
- int16_t int16

  Arbitrary value of type int16_t.

- int32_t int32

  Arbitrary value of type int32_t.
- int64_t int64

  Arbitrary value of type int64_t.
- int8_t int8

  Arbitrary value of type int8_t.
- OTF2_InterruptGeneratorRef interruptGeneratorRef

  References a InterruptGenerator definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_INTERRUPT_GENERATOR is available.
- OTF2_LocationRef locationRef

  References a Location definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_LOCATION is available.
- OTF2_MetricRef metricRef

  References a MetricClass, or a MetricInstance definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_METRIC is available.
- OTF2_ParameterRef parameterRef

  References a Parameter definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_PARAMETER is available.
- OTF2_RegionRef regionRef

  References a Region definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_REGION is available.
- OTF2_RmaWinRef rmaWinRef

  References a RmaWin definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_RMA_WIN is available.
- OTF2_SourceCodeLocationRef sourceCodeLocationRef

  References a SourceCodeLocation definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_SOURCE_CODE_LOCATION is available.
- OTF2_StringRef stringRef

  References a String definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_STRING is available.
- uint16_t uint16

  Arbitrary value of type uint16_t.
- uint32_t uint32

  Arbitrary value of type uint32_t.
- uint64_t uint64

  Arbitrary value of type uint64_t.
- uint8_t uint8

  Arbitrary value of type uint8_t.

D.1.1 Detailed Description

Value container for an attributes.

For definition references (*OTF2_MappingType*) use the same data type as the definition.

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

- otf2/OTF2_AttributeValue.h

D.2 OTF2_CollectiveCallbacks Struct Reference

Struct which holds all collective callbacks.

```
#include <otf2/OTF2_Callbacks.h>
```

D.2.1 Detailed Description

Struct which holds all collective callbacks.

Since

Version 1.3

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

- otf2/OTF2_Callbacks.h

D.3 OTF2_CollectiveContext Struct Reference

Collective context which wraps an MPI communicator.

```
#include <otf2/OTF2_MPI_Collectives.h>
```

D.3.1 Detailed Description

Collective context which wraps an MPI communicator.

User provided type for collective groups.

Since

Version 1.3

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

- otf2/OTF2_MPI_Collectives.h

D.4 OTF2_FlushCallbacks Struct Reference

Structure holding the flush callbacks.

```
#include <otf2/OTF2_Callbacks.h>
```

Data Fields

- OTF2_PostFlushCallback otf2_post_flush

  Callback which is called after a flush.
- OTF2_PreFlushCallback otf2_pre_flush

  Callback which is called prior a flush.

D.4.1 Detailed Description

Structure holding the flush callbacks.

To be used in a call to *OTF2_Archive_SetFlushCallbacks*.

otf2_post_flush callback may be NULL to suppress writing a BufferFlush record.

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

- otf2/OTF2_Callbacks.h

D.5 OTF2_Lock Struct Reference

The OpenMP locking object type.

```
#include <otf2/OTF2_OpenMP_Locks.h>
```

D.5.1 Detailed Description

The OpenMP locking object type.

Opaque type for a locking object.

The Pthread locking object type.

Since

Version 1.5

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

- otf2/OTF2_OpenMP_Locks.h
- otf2/OTF2_Pthread_Locks.h

D.6 OTF2_LockingCallbacks Struct Reference

Struct which holds all collective callbacks.

```
#include <otf2/OTF2_Callbacks.h>
```

D.6.1 Detailed Description

Struct which holds all collective callbacks.

Since

Version 1.5

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

- otf2/OTF2_Callbacks.h

D.7 OTF2_MemoryCallbacks Struct Reference

Structure holding the memory callbacks.

```
#include <otf2/OTF2_Callbacks.h>
```

Data Fields

- OTF2_MemoryAllocate otf2_allocate

  Callback which is called to allocate a new chunk.
- OTF2_MemoryFreeAll otf2_free_all

  Callback which is called to release all previous allocated chunks.

D.7.1 Detailed Description

Structure holding the memory callbacks.

To be used in a call to *OTF2_Archive_SetMemoryCallbacks*.

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

- otf2/OTF2_Callbacks.h

D.8 OTF2_MetricValue Union Reference

Metric value.

`#include <otf2/OTF2_Events.h>`

D.8.1 Detailed Description

Metric value.

Wrapper for enum *OTF2_MetricValue_union*.

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

- otf2/OTF2_Events.h

D.9 OTF2_MPI_UserData Struct Reference

User data structure, which will be used by the MPI collectives.

`#include <otf2/OTF2_MPI_Collectives.h>`

D.9.1 Detailed Description

User data structure, which will be used by the MPI collectives.

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

- otf2/OTF2_MPI_Collectives.h

D.10 OTF2_Pthread_UserData Struct Reference

User data structure, which will be used by the Pthread locks.

D.10 OTF2_Pthread_UserData Struct Reference

```
#include <otf2/OTF2_Pthread_Locks.h>
```

D.10.1 Detailed Description

User data structure, which will be used by the Pthread locks.

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

- otf2/OTF2_Pthread_Locks.h

Appendix E

File Documentation

E.1 otf2/OTF2_ErrorCodes.h File Reference

Error codes and error handling.

```
#include <errno.h>
#include <stdint.h>
#include <stdarg.h>
```

Typedefs

- typedef OTF2_ErrorCode(∗ OTF2_ErrorCallback)(void ∗userData, const char ∗file, uint64_t line, const char ∗function, OTF2_ErrorCode errorCode, const char ∗msgFormatString, va_list va)

Enumerations

- enum OTF2_ErrorCode {

  OTF2_DEPRECATED = -3,

  OTF2_ABORT = -2,

  OTF2_WARNING = -1,

  OTF2_SUCCESS = 0,

  OTF2_ERROR_INVALID = 1,

  OTF2_ERROR_E2BIG,

  OTF2_ERROR_EACCES,

  OTF2_ERROR_EADDRNOTAVAIL,

OTF2_ERROR_EAFNOSUPPORT,

OTF2_ERROR_EAGAIN,

OTF2_ERROR_EALREADY,

OTF2_ERROR_EBADF,

OTF2_ERROR_EBADMSG,

OTF2_ERROR_EBUSY,

OTF2_ERROR_ECANCELED,

OTF2_ERROR_ECHILD,

OTF2_ERROR_ECONNREFUSED,

OTF2_ERROR_ECONNRESET,

OTF2_ERROR_EDEADLK,

OTF2_ERROR_EDESTADDRREQ,

OTF2_ERROR_EDOM,

OTF2_ERROR_EDQUOT,

OTF2_ERROR_EEXIST,

OTF2_ERROR_EFAULT,

OTF2_ERROR_EFBIG,

OTF2_ERROR_EINPROGRESS,

OTF2_ERROR_EINTR,

OTF2_ERROR_EINVAL,

OTF2_ERROR_EIO,

OTF2_ERROR_EISCONN,

OTF2_ERROR_EISDIR,

OTF2_ERROR_ELOOP,

OTF2_ERROR_EMFILE,

OTF2_ERROR_EMLINK,

OTF2_ERROR_EMSGSIZE,

OTF2_ERROR_EMULTIHOP,

OTF2_ERROR_ENAMETOOLONG,

OTF2_ERROR_ENETDOWN,

OTF2_ERROR_ENETRESET,

OTF2_ERROR_ENETUNREACH,

OTF2_ERROR_ENFILE,

OTF2_ERROR_ENOBUFS,

OTF2_ERROR_ENODATA,

OTF2_ERROR_ENODEV,

OTF2_ERROR_ENOENT,

OTF2_ERROR_ENOEXEC,

OTF2_ERROR_ENOLCK,

OTF2_ERROR_ENOLINK,

OTF2_ERROR_ENOMEM,

OTF2_ERROR_ENOMSG,

OTF2_ERROR_ENOPROTOOPT,

OTF2_ERROR_ENOSPC,

OTF2_ERROR_ENOSR,

OTF2_ERROR_ENOSTR,

OTF2_ERROR_ENOSYS,

OTF2_ERROR_ENOTCONN,

OTF2_ERROR_ENOTDIR,

OTF2_ERROR_ENOTEMPTY,

OTF2_ERROR_ENOTSOCK,

OTF2_ERROR_ENOTSUP,

OTF2_ERROR_ENOTTY,

OTF2_ERROR_ENXIO,

OTF2_ERROR_EOPNOTSUPP,

OTF2_ERROR_EOVERFLOW,

OTF2_ERROR_EPERM,

OTF2_ERROR_EPIPE,

OTF2_ERROR_EPROTO,

OTF2_ERROR_EPROTONOSUPPORT,

OTF2_ERROR_EPROTOTYPE,

OTF2_ERROR_ERANGE,

OTF2_ERROR_EROFS,

OTF2_ERROR_ESPIPE,

OTF2_ERROR_ESRCH,

OTF2_ERROR_ESTALE,

OTF2_ERROR_ETIME,

OTF2_ERROR_ETIMEDOUT,

OTF2_ERROR_ETXTBSY,

OTF2_ERROR_EWOULDBLOCK,

OTF2_ERROR_EXDEV,

OTF2_ERROR_END_OF_FUNCTION,

OTF2_ERROR_INVALID_CALL,

OTF2_ERROR_INVALID_ARGUMENT,

OTF2_ERROR_INVALID_RECORD,

OTF2_ERROR_INVALID_DATA,

OTF2_ERROR_INVALID_SIZE_GIVEN,

OTF2_ERROR_UNKNOWN_TYPE,

OTF2_ERROR_INTEGRITY_FAULT,

OTF2_ERROR_MEM_FAULT,

OTF2_ERROR_MEM_ALLOC_FAILED,

OTF2_ERROR_PROCESSED_WITH_FAULTS,

OTF2_ERROR_INDEX_OUT_OF_BOUNDS,

OTF2_ERROR_INVALID_LINENO,

OTF2_ERROR_END_OF_BUFFER,

OTF2_ERROR_FILE_INTERACTION,

OTF2_ERROR_FILE_CAN_NOT_OPEN,

OTF2_ERROR_INTERRUPTED_BY_CALLBACK,

OTF2_ERROR_PROPERTY_NAME_INVALID,

OTF2_ERROR_PROPERTY_EXISTS,

OTF2_ERROR_PROPERTY_NOT_FOUND,

OTF2_ERROR_PROPERTY_VALUE_INVALID,

OTF2_ERROR_FILE_COMPRESSION_NOT_SUPPORTED,

OTF2_ERROR_DUPLICATE_MAPPING_TABLE,

OTF2_ERROR_INVALID_FILE_MODE_TRANSITION,

OTF2_ERROR_COLLECTIVE_CALLBACK,

OTF2_ERROR_FILE_SUBSTRATE_NOT_SUPPORTED,

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE,

OTF2_ERROR_LOCKING_CALLBACK,

OTF2_ERROR_HINT_INVALID,
OTF2_ERROR_HINT_LOCKED,
OTF2_ERROR_HINT_INVALID_VALUE }

Functions

- const char ∗ OTF2_Error_GetDescription (OTF2_ErrorCode errorCode)
- const char ∗ OTF2_Error_GetName (OTF2_ErrorCode errorCode)
- OTF2_ErrorCallback OTF2_Error_RegisterCallback (OTF2_ErrorCallback errorCallbackIn, void ∗userData)

E.1.1 Detailed Description

Error codes and error handling.

E.1.2 Typedef Documentation

E.1.2.1 typedef OTF2_ErrorCode(∗ OTF2_ErrorCallback)(void ∗userData, const char ∗file, uint64_t line, const char ∗function, OTF2_ErrorCode errorCode, const char ∗msgFormatString, va_list va)

Signature of error handler callback functions. The error handler can be set with OTF2_Error_RegisterCallback.

Parameters

userData	: Data passed to this function as given by the registry call.
file	: Name of the source-code file where the error appeared
line	: Line number in the source-code where the error appeared
function	: Name of the function where the error appeared
errorCode	: Error Code
msgFormat-String	: Format string like it is used at the printf family.
va	: Variable argument list

Returns

Should return the errorCode

E.1.3 Enumeration Type Documentation

E.1.3.1 enum OTF2_ErrorCode

This is the list of error codes for OTF2.

Enumerator:

OTF2_DEPRECATED Special marker for error messages which indicates an deprecation.

OTF2_ABORT Special marker when the application will be aborted.

OTF2_WARNING Special marker for error messages which are only warnings.

OTF2_SUCCESS Operation successful

OTF2_ERROR_INVALID Invalid error code

Should only be used internally and not as an actual error code.

OTF2_ERROR_E2BIG The list of arguments is to long

OTF2_ERROR_EACCES Not enough rights

OTF2_ERROR_EADDRNOTAVAIL Address is not available

OTF2_ERROR_EAFNOSUPPORT Address family is not supported

OTF2_ERROR_EAGAIN Resource temporaly not available

OTF2_ERROR_EALREADY Connection is already processed

OTF2_ERROR_EBADF Invalid file pointer

OTF2_ERROR_EBADMSG Invalid message

OTF2_ERROR_EBUSY Resource or device is busy

OTF2_ERROR_ECANCELED Operation was aborted

OTF2_ERROR_ECHILD No child process available

OTF2_ERROR_ECONNREFUSED Connection was refused

OTF2_ERROR_ECONNRESET Connection was reset

OTF2_ERROR_EDEADLK Resolved deadlock

OTF2_ERROR_EDESTADDRREQ Destination address was expected

OTF2_ERROR_EDOM Domain error

OTF2_ERROR_EDQUOT Reserved

OTF2_ERROR_EEXIST File does already exist

OTF2_ERROR_EFAULT Invalid Address

OTF2_ERROR_EFBIG File is to big

OTF2_ERROR_EINPROGRESS Operation is work in progress

OTF2_ERROR_EINTR Interuption of an operating system call

OTF2_ERROR_EINVAL Invalid argument

OTF2_ERROR_EIO Generic I/O error

OTF2_ERROR_EISCONN Socket is already connected

OTF2_ERROR_EISDIR Target is a directory

OTF2_ERROR_ELOOP To many layers of symbolic links

OTF2_ERROR_EMFILE To many opened files

OTF2_ERROR_EMLINK To many links

OTF2_ERROR_EMSGSIZE Message buffer is to small

OTF2_ERROR_EMULTIHOP Reserved

OTF2_ERROR_ENAMETOOLONG Filename is to long

OTF2_ERROR_ENETDOWN Network is down

OTF2_ERROR_ENETRESET Connection was reset from the network

OTF2_ERROR_ENETUNREACH Network is not reachable

OTF2_ERROR_ENFILE To much opened files

OTF2_ERROR_ENOBUFS No buffer space available

OTF2_ERROR_ENODATA No more data left in the queue

OTF2_ERROR_ENODEV This device does not support this function

OTF2_ERROR_ENOENT File or Directory does not exist

OTF2_ERROR_ENOEXEC Cannot execute binary

OTF2_ERROR_ENOLCK Locking failed

OTF2_ERROR_ENOLINK Reserved

OTF2_ERROR_ENOMEM Not enough main memory available

OTF2_ERROR_ENOMSG Message has not the expected type

OTF2_ERROR_ENOPROTOOPT This protocol is not available

OTF2_ERROR_ENOSPC No space left on device

OTF2_ERROR_ENOSR No stream available

OTF2_ERROR_ENOSTR This is not a stream

OTF2_ERROR_ENOSYS Requested function is not implemented

OTF2_ERROR_ENOTCONN Socket is not connected

OTF2_ERROR_ENOTDIR This is not an directory

OTF2_ERROR_ENOTEMPTY This directory is not empty

OTF2_ERROR_ENOTSOCK No socket

OTF2_ERROR_ENOTSUP This operation is not supported

OTF2_ERROR_ENOTTY This IOCTL is not supported by the device

OTF2_ERROR_ENXIO Device is not yet configured

OTF2_ERROR_EOPNOTSUPP Operation is not supported by this socket

OTF2_ERROR_EOVERFLOW Value is to long for the datatype

OTF2_ERROR_EPERM Operation is not permitted

OTF2_ERROR_EPIPE Broken pipe

OTF2_ERROR_EPROTO Protocoll error

OTF2_ERROR_EPROTONOSUPPORT Protocoll is not supported

OTF2_ERROR_EPROTOTYPE Wrong protocoll type for this socket

OTF2_ERROR_ERANGE Value is out of range

OTF2_ERROR_EROFS Filesystem is read only

OTF2_ERROR_ESPIPE This seek is not allowed

OTF2_ERROR_ESRCH No matching process found

OTF2_ERROR_ESTALE Reserved

OTF2_ERROR_ETIME Timout in file stream or IOCTL

OTF2_ERROR_ETIMEDOUT Connection timed out

OTF2_ERROR_ETXTBSY File couldn't be executed while it is opened

OTF2_ERROR_EWOULDBLOCK Operation would be blocking

OTF2_ERROR_EXDEV Invalid link between devices

OTF2_ERROR_END_OF_FUNCTION Unintentional reached end of function

OTF2_ERROR_INVALID_CALL Function call not allowed in current state

OTF2_ERROR_INVALID_ARGUMENT Parameter value out of range

OTF2_ERROR_INVALID_RECORD Invalid definition or event record

OTF2_ERROR_INVALID_DATA Invalid or inconsistent record data

OTF2_ERROR_INVALID_SIZE_GIVEN The given size cannot be used

OTF2_ERROR_UNKNOWN_TYPE The given type is not known

OTF2_ERROR_INTEGRITY_FAULT The structural integrity is not given

OTF2_ERROR_MEM_FAULT This could not be done with the given memory

OTF2_ERROR_MEM_ALLOC_FAILED Memory allocation failed

OTF2_ERROR_PROCESSED_WITH_FAULTS An error appeared when data was processed

OTF2_ERROR_INDEX_OUT_OF_BOUNDS Index out of bounds

OTF2_ERROR_INVALID_LINENO Invalid source code line number

OTF2_ERROR_END_OF_BUFFER End of buffer/file reached

OTF2_ERROR_FILE_INTERACTION Invalid file operation

OTF2_ERROR_FILE_CAN_NOT_OPEN Unable to open file

OTF2_ERROR_INTERRUPTED_BY_CALLBACK Record reading interrupted by reader callback

OTF2_ERROR_PROPERTY_NAME_INVALID Property name does not conform to the naming scheme

OTF2_ERROR_PROPERTY_EXISTS Property already exists

OTF2_ERROR_PROPERTY_NOT_FOUND Property not found found in this archive

OTF2_ERROR_PROPERTY_VALUE_INVALID Property value does not have the expected value

OTF2_ERROR_FILE_COMPRESSION_NOT_SUPPORTED Missing library support for requested compression mode

OTF2_ERROR_DUPLICATE_MAPPING_TABLE Multiple definitions for the same mapping type

OTF2_ERROR_INVALID_FILE_MODE_TRANSITION File mode transition not permitted

OTF2_ERROR_COLLECTIVE_CALLBACK Collective callback failed

OTF2_ERROR_FILE_SUBSTRATE_NOT_SUPPORTED Missing library support for requested file substrate

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE The type of the attribute does not match the expected one.

OTF2_ERROR_LOCKING_CALLBACK Locking callback failed

OTF2_ERROR_HINT_INVALID The hint is not valid for the current operation mode of OTF2.

OTF2_ERROR_HINT_LOCKED The hint was either already set by the user or at least once queried from OTF2.

OTF2_ERROR_HINT_INVALID_VALUE Invalid value for hint.

E.1.4 Function Documentation

E.1.4.1 const char∗ OTF2_Error_GetDescription (OTF2_ErrorCode *errorCode*)

Returns the description of an error code.

Parameters

errorCode	: Error Code

Returns

Returns the description of a known error code.

E.1.4.2 const char* OTF2_Error_GetName (OTF2_ErrorCode *errorCode*)

Returns the name of an error code.

Parameters

errorCode	: Error Code

Returns

Returns the name of a known error code, and "INVALID_ERROR" for invalid or unknown error IDs.

E.1.4.3 OTF2_ErrorCallback OTF2_Error_RegisterCallback (OTF2_ErrorCallback *errorCallbackIn*, void * *userData*)

Register a programmers callback function for error handling.

Parameters

errorCallbackIn	: Fucntion will be called instead of printing a default message to standard error.
userData	: Data pointer passed to the callback.

Returns

Function pointer to the former error handling function.

E.2 otf2/otf2.h File Reference

Main include file for applications using OTF2.

```
#include <otf2/OTF2_Reader.h>
```

E.2.1 Detailed Description

Main include file for applications using OTF2.

E.3 otf2/OTF2_Archive.h File Reference

Writing interface for OTF2 archives.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Callbacks.h>
#include <otf2/OTF2_DefWriter.h>
#include <otf2/OTF2_DefReader.h>
#include <otf2/OTF2_EvtWriter.h>
#include <otf2/OTF2_EvtReader.h>
#include <otf2/OTF2_SnapWriter.h>
#include <otf2/OTF2_SnapReader.h>
#include <otf2/OTF2_GlobalDefWriter.h>
#include <otf2/OTF2_GlobalDefReader.h>
#include <otf2/OTF2_GlobalEvtReader.h>
#include <otf2/OTF2_GlobalSnapReader.h>
#include <otf2/OTF2_Thumbnail.h>
#include <otf2/OTF2_MarkerWriter.h>
#include <otf2/OTF2_MarkerReader.h>
```

Defines

- #define OTF2_CHUNK_SIZE_DEFINITIONS_DEFAULT ($4 * 1024 * 1024$)

  Default size for OTF2's internal event chunk memory handling.
- #define OTF2_CHUNK_SIZE_EVENTS_DEFAULT ($1024 * 1024$)

  Default size for OTF2's internal event chunk memory handling.

Typedefs

- typedef struct OTF2_Archive_struct OTF2_Archive

Keeps all meta-data for an OTF2 archive.

Functions

- OTF2_ErrorCode OTF2_Archive_Close (OTF2_Archive ∗archive)

  Close an opened archive.
- OTF2_ErrorCode OTF2_Archive_CloseDefFiles (OTF2_Archive ∗archive)

  Closes the local definitions file container.
- OTF2_ErrorCode OTF2_Archive_CloseDefReader (OTF2_Archive ∗archive, OTF2_DefReader ∗reader)

  Close an opened local definition reader.
- OTF2_ErrorCode OTF2_Archive_CloseDefWriter (OTF2_Archive ∗archive, OTF2_DefWriter ∗writer)

  Close an opened local definition writer.
- OTF2_ErrorCode OTF2_Archive_CloseEvtFiles (OTF2_Archive ∗archive)

  Closes the events file container.
- OTF2_ErrorCode OTF2_Archive_CloseEvtReader (OTF2_Archive ∗archive, OTF2_EvtReader ∗reader)

  Close an opened local event reader.
- OTF2_ErrorCode OTF2_Archive_CloseEvtWriter (OTF2_Archive ∗archive, OTF2_EvtWriter ∗writer)

  Close an opened local event writer.
- OTF2_ErrorCode OTF2_Archive_CloseGlobalDefReader (OTF2_Archive ∗archive, OTF2_GlobalDefReader ∗globalDefReader)

  Closes the global definition reader.
- OTF2_ErrorCode OTF2_Archive_CloseGlobalDefWriter (OTF2_Archive ∗archive, OTF2_GlobalDefWriter ∗writer)

  Close an opened global definition writer.
- OTF2_ErrorCode OTF2_Archive_CloseGlobalEvtReader (OTF2_Archive ∗archive, OTF2_GlobalEvtReader ∗globalEvtReader)

  Closes the global event reader.
- OTF2_ErrorCode OTF2_Archive_CloseGlobalSnapReader (OTF2_Archive ∗archive, OTF2_GlobalSnapReader ∗globalSnapReader)

  Close the opened global snapshot reader.
- OTF2_ErrorCode OTF2_Archive_CloseMarkerReader (OTF2_Archive ∗archive, OTF2_MarkerReader ∗markerReader)

  Closes the marker reader.

- OTF2_ErrorCode OTF2_Archive_CloseMarkerWriter (OTF2_Archive ∗archive, OTF2_MarkerWriter ∗writer)

  Close an opened marker writer.
- OTF2_ErrorCode OTF2_Archive_CloseSnapFiles (OTF2_Archive ∗archive)

  Closes the snapshots file container.
- OTF2_ErrorCode OTF2_Archive_CloseSnapReader (OTF2_Archive ∗archive, OTF2_SnapReader ∗reader)

  Close an opened local snap reader.
- OTF2_ErrorCode OTF2_Archive_CloseSnapWriter (OTF2_Archive ∗archive, OTF2_SnapWriter ∗writer)

  Close an opened local snap writer.
- OTF2_ErrorCode OTF2_Archive_CloseThumbReader (OTF2_Archive ∗archive, OTF2_ThumbReader ∗reader)

  Close an opened thumbnail reader.
- OTF2_ErrorCode OTF2_Archive_GetChunkSize (OTF2_Archive ∗archive, uint64_t ∗chunkSizeEvents, uint64_t ∗chunkSizeDefs)

  Get the chunksize.
- OTF2_ErrorCode OTF2_Archive_GetCompression (OTF2_Archive ∗archive, OTF2_Compression ∗compression)

  Get compression mode (none or zlib)
- OTF2_ErrorCode OTF2_Archive_GetCreator (OTF2_Archive ∗archive, char ∗∗creator)

  Get creator information.
- OTF2_DefReader ∗ OTF2_Archive_GetDefReader (OTF2_Archive ∗archive, OTF2_LocationRef location)

  Get a local definition reader.
- OTF2_DefWriter ∗ OTF2_Archive_GetDefWriter (OTF2_Archive ∗archive, OTF2_LocationRef location)

  Get a local definition writer.
- OTF2_ErrorCode OTF2_Archive_GetDescription (OTF2_Archive ∗archive, char ∗∗description)

  Get description.
- OTF2_EvtReader ∗ OTF2_Archive_GetEvtReader (OTF2_Archive ∗archive, OTF2_LocationRef location)

  Get a local event reader.
- OTF2_EvtWriter ∗ OTF2_Archive_GetEvtWriter (OTF2_Archive ∗archive, OTF2_LocationRef location)

  Get a local event writer.

- OTF2_ErrorCode OTF2_Archive_GetFileSubstrate (OTF2_Archive ∗archive, OTF2_FileSubstrate ∗substrate)

  Get the file substrate (posix, sion, none)
- OTF2_GlobalDefReader ∗ OTF2_Archive_GetGlobalDefReader (OTF2_Archive ∗archive)

  Get a global definition reader.
- OTF2_GlobalDefWriter ∗ OTF2_Archive_GetGlobalDefWriter (OTF2_Archive ∗archive)

  Get a global definition writer.
- OTF2_GlobalEvtReader ∗ OTF2_Archive_GetGlobalEvtReader (OTF2_Archive ∗archive)

  Get a global event reader.
- OTF2_GlobalSnapReader ∗ OTF2_Archive_GetGlobalSnapReader (OTF2_-Archive ∗archive)

  Get a global snap reader.
- OTF2_ErrorCode OTF2_Archive_GetMachineName (OTF2_Archive ∗archive, char ∗∗machineName)

  Get machine name.
- OTF2_MarkerReader ∗ OTF2_Archive_GetMarkerReader (OTF2_Archive ∗archive)

  Get a marker reader.
- OTF2_MarkerWriter ∗ OTF2_Archive_GetMarkerWriter (OTF2_Archive ∗archive)

  Get a marker writer.
- OTF2_ErrorCode OTF2_Archive_GetNumberOfGlobalDefinitions (OTF2_-Archive ∗archive, uint64_t ∗numberOfDefinitions)

  Get the number of global definitions.
- OTF2_ErrorCode OTF2_Archive_GetNumberOfLocations (OTF2_Archive ∗archive, uint64_t ∗numberOfLocations)

  Get the number of locations.
- OTF2_ErrorCode OTF2_Archive_GetNumberOfSnapshots (OTF2_Archive ∗archive, uint32_t ∗number)

  Get the number of snapshots.
- OTF2_ErrorCode OTF2_Archive_GetNumberOfThumbnails (OTF2_Archive ∗archive, uint32_t ∗number)

  Get the number of thumbnails.
- OTF2_ErrorCode OTF2_Archive_GetProperty (OTF2_Archive ∗archive, const char ∗name, char ∗∗value)

  Get the value of the named trace file property.

- OTF2_ErrorCode OTF2_Archive_GetPropertyNames (OTF2_Archive ∗archive, uint32_t ∗numberOfProperties, char ∗∗∗names)

  Get the names of all trace file properties.
- OTF2_SnapReader ∗ OTF2_Archive_GetSnapReader (OTF2_Archive ∗archive, OTF2_LocationRef location)

  Get a local snap reader.
- OTF2_SnapWriter ∗ OTF2_Archive_GetSnapWriter (OTF2_Archive ∗archive, OTF2_LocationRef location)

  Get a local snap writer.
- OTF2_ThumbReader ∗ OTF2_Archive_GetThumbReader (OTF2_Archive ∗archive, uint32_t number)

  Get a thumb reader.
- OTF2_ThumbWriter ∗ OTF2_Archive_GetThumbWriter (OTF2_Archive ∗archive, const char ∗name, const char ∗description, OTF2_ThumbnailType type, uint32_-t numberOfSamples, uint32_t numberOfMetrics, const uint64_t ∗refsToDefs)

  Get a thumb writer.
- OTF2_ErrorCode OTF2_Archive_GetTraceId (OTF2_Archive ∗archive, uint64_-t ∗id)

  Get the identifier of the trace file.
- OTF2_ErrorCode OTF2_Archive_GetVersion (OTF2_Archive ∗archive, uint8_-t ∗major, uint8_t ∗minor, uint8_t ∗bugfix)

  Get format version.
- OTF2_Archive ∗ OTF2_Archive_Open (const char ∗archivePath, const char ∗archiveName, const OTF2_FileMode fileMode, const uint64_t chunkSizeEvents, const uint64_t chunkSizeDefs, const OTF2_FileSubstrate fileSubstrate, const OTF2_Compression compression)

  Create a new archive.
- OTF2_ErrorCode OTF2_Archive_OpenDefFiles (OTF2_Archive ∗archive)

  Open the local definitions file container.
- OTF2_ErrorCode OTF2_Archive_OpenEvtFiles (OTF2_Archive ∗archive)

  Open the events file container.
- OTF2_ErrorCode OTF2_Archive_OpenSnapFiles (OTF2_Archive ∗archive)

  Open the snapshots file container.
- OTF2_ErrorCode OTF2_Archive_SelectLocation (OTF2_Archive ∗archive, OTF2_LocationRef location)

  Select a location to be read.

- OTF2_ErrorCode OTF2_Archive_SetBoolProperty (OTF2_Archive ∗archive, const char ∗name, bool value, bool overwrite)

  Add or remove a boolean trace file property to this archive.
- OTF2_ErrorCode OTF2_Archive_SetCollectiveCallbacks (OTF2_Archive ∗archive, const OTF2_CollectiveCallbacks ∗collectiveCallbacks, void ∗collectiveData, OTF2_CollectiveContext ∗globalCommContext, OTF2_CollectiveContext ∗localCommContext)

  Set the collective callbacks for the archive.
- OTF2_ErrorCode OTF2_Archive_SetCreator (OTF2_Archive ∗archive, const char ∗creator)

  Set creator.
- OTF2_ErrorCode OTF2_Archive_SetDescription (OTF2_Archive ∗archive, const char ∗description)

  Set a description.
- OTF2_ErrorCode OTF2_Archive_SetFlushCallbacks (OTF2_Archive ∗archive, const OTF2_FlushCallbacks ∗flushCallbacks, void ∗flushData)

  Set the flush callbacks for the archive.
- OTF2_ErrorCode OTF2_Archive_SetHint (OTF2_Archive ∗archive, OTF2_-Hint hint, void ∗value)

  Set the hint in the archive to the given value.
- OTF2_ErrorCode OTF2_Archive_SetLockingCallbacks (OTF2_Archive ∗archive, const OTF2_LockingCallbacks ∗lockingCallbacks, void ∗lockingData)

  Set the locking callbacks for the archive.
- OTF2_ErrorCode OTF2_Archive_SetMachineName (OTF2_Archive ∗archive, const char ∗machineName)

  Set machine name.
- OTF2_ErrorCode OTF2_Archive_SetMemoryCallbacks (OTF2_Archive ∗archive, const OTF2_MemoryCallbacks ∗memoryCallbacks, void ∗memoryData)

  Set the memory callbacks for the archive.
- OTF2_ErrorCode OTF2_Archive_SetNumberOfSnapshots (OTF2_Archive ∗archive, uint32_t number)

  Set the number of snapshots.
- OTF2_ErrorCode OTF2_Archive_SetProperty (OTF2_Archive ∗archive, const char ∗name, const char ∗value, bool overwrite)

  Add or remove a trace file property to this archive.
- OTF2_ErrorCode OTF2_Archive_SetSerialCollectiveCallbacks (OTF2_Archive ∗archive)

  Convenient function to set the collective callbacks to an serial implementation.
- OTF2_ErrorCode OTF2_Archive_SwitchFileMode (OTF2_Archive ∗archive, OTF2_FileMode newFileMode)

  Switch file mode of the archive.

E.3.1 Detailed Description

Writing interface for OTF2 archives.

E.3.2 Define Documentation

E.3.2.1 #define OTF2_CHUNK_SIZE_DEFINITIONS_DEFAULT (4 ∗ 1024 ∗ 1024)

Default size for OTF2's internal event chunk memory handling.

If you are not sure which chunk size is the best to use, use this default value.

E.3.2.2 #define OTF2_CHUNK_SIZE_EVENTS_DEFAULT (1024 ∗ 1024)

Default size for OTF2's internal event chunk memory handling.

If you are not sure which chunk size is the best to use, use this default value.

E.3.3 Typedef Documentation

E.3.3.1 typedef struct OTF2_Archive_struct OTF2_Archive

Keeps all meta-data for an OTF2 archive.

An OTF2 archive handle keeps all runtime information about an OTF2 archive. It is the central handle to get and set information about the archive and to request event and definition writer handles.

E.3.4 Function Documentation

E.3.4.1 OTF2_ErrorCode OTF2_Archive_Close (OTF2_Archive ∗ *archive*)

Close an opened archive.

Closes an opened archive and releases the associated resources. Closes also all opened writer and reader handles. Does nothing if NULL is passed.

Parameters

archive	Archive handle.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.2 OTF2_ErrorCode OTF2_Archive_CloseDefFiles (OTF2_Archive ∗ *archive*)

Closes the local definitions file container.

This function is an collective operation.

Parameters

archive	Archive handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.3 OTF2_ErrorCode OTF2_Archive_CloseDefReader (OTF2_Archive ∗ *archive*, OTF2_DefReader ∗ *reader*)

Close an opened local definition reader.

Parameters

archive	Archive handle.
reader	Reader handle to be closed.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.4 OTF2_ErrorCode OTF2_Archive_CloseDefWriter (OTF2_Archive ∗ *archive*, OTF2_DefWriter ∗ *writer*)

Close an opened local definition writer.

Parameters

archive	Archive handle.
writer	Writer handle to be closed.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.5 OTF2_ErrorCode OTF2_Archive_CloseEvtFiles (OTF2_Archive * *archive*)

Closes the events file container.

This function is an collective operation.

Parameters

archive	Archive handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.6 OTF2_ErrorCode OTF2_Archive_CloseEvtReader (OTF2_Archive * *archive*, OTF2_EvtReader * *reader*)

Close an opened local event reader.

Parameters

archive	Archive handle.
reader	Reader handle to be closed.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.7 OTF2_ErrorCode OTF2_Archive_CloseEvtWriter (OTF2_Archive * *archive*, OTF2_EvtWriter * *writer*)

Close an opened local event writer.

Parameters

archive	Archive handle.
writer	Writer handle to be closed.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.8 OTF2_ErrorCode OTF2_Archive_CloseGlobalDefReader (OTF2_Archive ∗ *archive*, OTF2_GlobalDefReader ∗ *globalDefReader*)

Closes the global definition reader.

Parameters

archive	Archive handle.
globalDef-Reader	The global definition reader.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.9 OTF2_ErrorCode OTF2_Archive_CloseGlobalDefWriter (OTF2_Archive ∗ *archive*, OTF2_GlobalDefWriter ∗ *writer*)

Close an opened global definition writer.

Only the master archive can call this function.

Parameters

archive	Archive handle.
writer	Writer handle to be closed.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if the archive or writer argument is invalid

OTF2_ERROR_INVALID_CALL if the archive is not in master mode

E.3.4.10 OTF2_ErrorCode OTF2_Archive_CloseGlobalEvtReader (OTF2_Archive ∗ *archive*, OTF2_GlobalEvtReader ∗ *globalEvtReader*)

Closes the global event reader.

This closes also all local event readers.

Parameters

archive	Archive handle.

glob-alEvtReader	The global event reader.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.11 OTF2_ErrorCode OTF2_Archive_CloseGlobalSnapReader (OTF2_Archive * *archive*, OTF2_GlobalSnapReader * *globalSnapReader*)

Close the opened global snapshot reader.

Parameters

archive	Archive handle.
global-SnapReader	Reader handle to be closed.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.12 OTF2_ErrorCode OTF2_Archive_CloseMarkerReader (OTF2_Archive * *archive*, OTF2_MarkerReader * *markerReader*)

Closes the marker reader.

Parameters

archive	Archive handle.
marker-Reader	The marker reader.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.13 OTF2_ErrorCode OTF2_Archive_CloseMarkerWriter (OTF2_Archive ∗ *archive*, OTF2_MarkerWriter ∗ *writer*)

Close an opened marker writer.

Parameters

archive	Archive handle.
writer	Writer handle to be closed.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.14 OTF2_ErrorCode OTF2_Archive_CloseSnapFiles (OTF2_Archive ∗ *archive*)

Closes the snapshots file container.

This function is an collective operation.

Parameters

archive	Archive handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.15 OTF2_ErrorCode OTF2_Archive_CloseSnapReader (OTF2_Archive ∗ *archive*, OTF2_SnapReader ∗ *reader*)

Close an opened local snap reader.

Parameters

archive	Archive handle.
reader	Reader handle to be closed.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.3.4.16 OTF2_ErrorCode OTF2_Archive_CloseSnapWriter (OTF2_Archive ∗ *archive*, OTF2_SnapWriter ∗ *writer*)

Close an opened local snap writer.

Parameters

archive	Archive handle.
writer	Writer handle to be closed.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.17 OTF2_ErrorCode OTF2_Archive_CloseThumbReader (OTF2_Archive ∗ *archive*, OTF2_ThumbReader ∗ *reader*)

Close an opened thumbnail reader.

Parameters

archive	Archive handle.
reader	Reader handle to be closed.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.18 OTF2_ErrorCode OTF2_Archive_GetChunkSize (OTF2_Archive ∗ *archive*, uint64_t ∗ *chunkSizeEvents*, uint64_t ∗ *chunkSizeDefs*)

Get the chunksize.

Parameters

	archive	Archive handle.
out	*chunk-SizeEvents*	Chunk size for event files.
out	*chunk-SizeDefs*	Chunk size for definition files.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.19 OTF2_ErrorCode OTF2_Archive_GetCompression (OTF2_Archive ∗ *archive*, OTF2_Compression ∗ *compression*)

Get compression mode (none or zlib)

Parameters

	archive	Archive handle.
out	*compres-sion*	Returned compression mode.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.20 OTF2_ErrorCode OTF2_Archive_GetCreator (OTF2_Archive ∗ *archive*, char ∗∗ *creator*)

Get creator information.

Parameters

	archive	Archive handle.
out	*creator*	Returned creator. Allocated with *malloc*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.21 OTF2_DefReader∗ OTF2_Archive_GetDefReader (OTF2_Archive ∗ *archive*, OTF2_LocationRef *location*)

Get a local definition reader.

Parameters

archive	Archive handle.
location	Location ID of the requested reader handle.

Returns

Returns a local definition reader handle if successful, NULL if an error occurs.

E.3.4.22 OTF2_DefWriter∗ OTF2_Archive_GetDefWriter (OTF2_Archive ∗ *archive*, OTF2_LocationRef *location*)

Get a local definition writer.

Parameters

archive	Archive handle.
location	Location ID of the requested writer handle.

Returns

Returns a local definition writer handle if successful, NULL if an error occurs.

E.3.4.23 OTF2_ErrorCode OTF2_Archive_GetDescription (OTF2_Archive ∗ *archive*, char ∗∗ *description*)

Get description.

Parameters

	archive	Archive handle.
`out`	*description*	Returned description. Allocated with *malloc*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.24 OTF2_EvtReader∗ OTF2_Archive_GetEvtReader (OTF2_Archive ∗ *archive*, OTF2_LocationRef *location*)

Get a local event reader.

Parameters

archive	Archive handle.
location	Location ID of the requested reader handle.

Returns

Returns a local event reader handle if successful, NULL if an error occurs.

E.3.4.25 OTF2_EvtWriter∗ OTF2_Archive_GetEvtWriter (OTF2_Archive ∗ *archive*, OTF2_LocationRef *location*)

Get a local event writer.

Parameters

archive	Archive handle.
location	Location ID of the requested writer handle.

Returns

Returns a local event writer handle if successful, NULL if an error occurs.

E.3.4.26 OTF2_ErrorCode OTF2_Archive_GetFileSubstrate (OTF2_Archive ∗ *archive*, OTF2_FileSubstrate ∗ *substrate*)

Get the file substrate (posix, sion, none)

Parameters

	archive	Archive handle.
out	*substrate*	Returned file substrate.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.27 OTF2_GlobalDefReader∗ OTF2_Archive_GetGlobalDefReader (OTF2_Archive ∗ *archive*)

Get a global definition reader.

Only the master archive can call this function.

Parameters

archive	Archive handle.

Returns

Returns a global definition reader handle if successful, NULL if an error occurs.

E.3.4.28 OTF2_GlobalDefWriter∗ OTF2_Archive_GetGlobalDefWriter (OTF2_Archive ∗ *archive*)

Get a global definition writer.

Parameters

archive	Archive handle.

Returns

Returns a global definition writer handle if successful, NULL if an error occurs.

E.3.4.29 OTF2_GlobalEvtReader∗ OTF2_Archive_GetGlobalEvtReader (OTF2_Archive ∗ *archive*)

Get a global event reader.

Parameters

archive	Archive handle.

Returns

Returns a global event reader handle if successful, NULL if an error occurs.

E.3.4.30 OTF2_GlobalSnapReader∗ OTF2_Archive_GetGlobalSnapReader (OTF2_Archive ∗ *archive*)

Get a global snap reader.

Parameters

archive	Archive handle.

Since

Version 1.2

Returns

Returns a global snap reader handle if successful, NULL if an error occurs.

E.3.4.31 OTF2_ErrorCode OTF2_Archive_GetMachineName (OTF2_Archive ∗ *archive*, char ∗∗ *machineName*)

Get machine name.

Parameters

	archive	Archive handle.
out	*machineName*	Returned machine name. Allocated with *malloc*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.32 OTF2_MarkerReader∗ OTF2_Archive_GetMarkerReader (OTF2_Archive ∗ *archive*)

Get a marker reader.

Parameters

archive	Archive handle.

Since

Version 1.2

Returns

Returns a marker reader handle if successful, NULL if an error occurs.

E.3.4.33 OTF2_MarkerWriter∗ OTF2_Archive_GetMarkerWriter (OTF2_Archive ∗ *archive*)

Get a marker writer.

Parameters

archive	Archive handle.

Since

Version 1.2

Returns

Returns a marker writer handle if successful, NULL if an error occurs.

E.3.4.34 OTF2_ErrorCode OTF2_Archive_GetNumberOfGlobalDefinitions (OTF2_Archive ∗ *archive,* uint64_t ∗ *numberOfDefinitions*)

Get the number of global definitions.

Parameters

	archive	Archive handle.
out	*numberOfDefinitions*	Return pointer to the number of global definitions.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.35 OTF2_ErrorCode OTF2_Archive_GetNumberOfLocations (OTF2_Archive * *archive*, uint64_t * *numberOfLocations*)

Get the number of locations.

Parameters

	archive	Archive handle.
out	*numberOfLocations*	Return pointer to the number of locations.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.36 OTF2_ErrorCode OTF2_Archive_GetNumberOfSnapshots (OTF2_Archive * *archive*, uint32_t * *number*)

Get the number of snapshots.

Parameters

archive	Archive handle.
number	Snapshot number.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.37 OTF2_ErrorCode OTF2_Archive_GetNumberOfThumbnails (OTF2_Archive * *archive*, uint32_t * *number*)

Get the number of thumbnails.

Parameters

archive	Archive handle.
number	Thumb number.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.38 OTF2_ErrorCode OTF2_Archive_GetProperty (OTF2_Archive ∗ *archive*, const char ∗ *name*, char ∗∗ *value*)

Get the value of the named trace file property.

Parameters

	archive	Archive handle.
	name	Name of the property.
out	*value*	Returned value of the property. Allocated with *malloc*.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_PROPERTY_NOT_FOUND if the named property was not found

E.3.4.39 OTF2_ErrorCode OTF2_Archive_GetPropertyNames (OTF2_Archive ∗ *archive*, uint32_t ∗ *numberOfProperties*, char ∗∗∗ *names*)

Get the names of all trace file properties.

Parameters

	archive	Archive handle.
out	*numberOf-Properties*	Returned number of trace file properties.
out	*names*	Returned list of property names. Allocated with *malloc*. To release memory, just pass `*names` to *free*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.40 OTF2_SnapReader∗ OTF2_Archive_GetSnapReader (OTF2_Archive ∗ *archive*, OTF2_LocationRef *location*)

Get a local snap reader.

Parameters

archive	Archive handle.
location	Location ID of the requested snap handle.

Since

Version 1.2

Returns

Returns a local snap handle if successful, NULL if an error occurs.

E.3.4.41 OTF2_SnapWriter∗ OTF2_Archive_GetSnapWriter (OTF2_Archive ∗ *archive*, OTF2_LocationRef *location*)

Get a local snap writer.

Parameters

archive	Archive handle.
location	Location ID of the requested writer handle.

Since

Version 1.2

Returns

Returns a local event writer handle if successful, NULL if an error occurs.

E.3.4.42 OTF2_ThumbReader∗ OTF2_Archive_GetThumbReader (OTF2_Archive ∗ *archive*, uint32_t *number*)

Get a thumb reader.

Parameters

archive	Archive handle.
number	Thumbnail number.

Since

Version 1.2

Returns

Returns a global definition writer handle if successful, NULL if an error occurs.

E.3.4.43 OTF2_ThumbWriter∗ OTF2_Archive_GetThumbWriter (OTF2_Archive ∗ *archive,* const char ∗ *name,* const char ∗ *description,* OTF2_ThumbnailType *type,* uint32_t *numberOfSamples,* uint32_t *numberOfMetrics,* const uint64_t ∗ *refsToDefs*)

Get a thumb writer.

Parameters

archive	Archive handle.
name	Name of thumb.
description	Description of thumb.
type	Type of thumb.
numberOf-Samples	Number of samples.
numberOf-Metrics	Number of metrics.
refsToDefs	*numberOfMetrics* references to defintion matching the thumbnail type.

Since

Version 1.2

Returns

Returns a thumb writer handle if successful, NULL if an error occurs.

E.3.4.44 OTF2_ErrorCode OTF2_Archive_GetTraceId (OTF2_Archive ∗ *archive,* uint64_t ∗ *id*)

Get the identifier of the trace file.

Note

This call is only allowed when the archive was opened with mode OTF2_-FILEMODE_READ.

Parameters

	archive	Archive handle.
out	*id*	Trace identifier.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.45 OTF2_ErrorCode OTF2_Archive_GetVersion (OTF2_Archive ∗ *archive*, uint8_t ∗ *major*, uint8_t ∗ *minor*, uint8_t ∗ *bugfix*)

Get format version.

Parameters

	archive	Archive handle
out	*major*	Major version number
out	*minor*	Minor version number
out	*bugfix*	Bugfix revision

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.46 OTF2_Archive∗ OTF2_Archive_Open (const char ∗ *archivePath*, const char ∗ *archiveName*, const OTF2_FileMode *fileMode*, const uint64_t *chunkSizeEvents*, const uint64_t *chunkSizeDefs*, const OTF2_FileSubstrate *fileSubstrate*, const OTF2_Compression *compression*)

Create a new archive.

Creates a new archive handle that keeps all meta data about the archive on runtime.

Parameters

archivePath	Path to the archive i.e. the directory where the anchor file is located.
archive-Name	Name of the archive. It is used to generate sub pathes e.g. 'archive-Name.otf2'.
fileMode	Determines if in reading or writing mode. Available values are *OTF2_-FILEMODE_WRITE* or *OTF2_FILEMODE_READ*.

chunk-SizeEvents	Requested size of OTF2's internal event chunks in writing mode. Available values are from 256kB to 16MB. The event chunk size affects performance as well as total memory usage. A value satisfying both is about 1MB. If you are not sure which chunk size is the best to use, use *OTF2_CHUNK_SIZE_EVENTS_DEFAULT*. In reading mode this value is ignored because the correct chunk size is extracted from the anchor file.
chunk-SizeDefs	Requested size of OTF2's internal definition chunks in writing mode. Available values are from 256kB to 16MB. The definition chunk size affects performance as well as total memory usage. In addition, the definition chunk size must be big enough to carry the largest possible definition record. Therefore, the definition chunk size must be at least 10 times the number of locations. A value satisfying these requirements is about 4MB. If you are not sure which chunk size is the best to use, use *OTF2_CHUNK_SIZE_DEFINITIONS_DEFAULT*. In reading mode this value is ignored because the correct chunk size is extracted from the anchor file.
fileSub-strate	Determines which file substrate should be used in writing mode. Available values are *OTF2_SUBSTRATE_POSIX* to use the standard Posix interface, *OTF2_SUBSTRATE_SION* to use an installed SION library to store multiple logical files into fewer or one physical file, and *OTF2_-SUBSTRATE_NONE* to supress file writing at all. In reading mode this value is ignored because the correct file substrated is extracted from the anchor file.
compres-sion	Determines if compression is used to reduce the size of data in files. Available values are *OTF2_COMPRESSION_ZLIB* to use an installed zlib and *OTF2_COMPRESSION_NONE* to disable compression. In reading mode this value is ignored because the correct file compression is extracted from the anchor file.

Returns

Returns an archive handle if successful, NULL otherwise.

E.3.4.47 OTF2_ErrorCode OTF2_Archive_OpenDefFiles (OTF2_Archive ∗ *archive*)

Open the local definitions file container.

This function is an collective operation.

Parameters

archive	Archive handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.48 OTF2_ErrorCode OTF2_Archive_OpenEvtFiles (OTF2_Archive ∗ *archive*)

Open the events file container.

This function is an collective operation.

Parameters

archive	Archive handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.49 OTF2_ErrorCode OTF2_Archive_OpenSnapFiles (OTF2_Archive ∗ *archive*)

Open the snapshots file container.

This function is an collective operation.

Parameters

archive	Archive handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.50 OTF2_ErrorCode OTF2_Archive_SelectLocation (OTF2_Archive ∗ *archive*, OTF2_LocationRef *location*)

Select a location to be read.

Parameters

archive	Archive handle.
location	Location ID.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.51 OTF2_ErrorCode OTF2_Archive_SetBoolProperty (OTF2_Archive ∗ *archive*, const char ∗ *name*, bool *value*, bool *overwrite*)

Add or remove a boolean trace file property to this archive.

Note

This call is only allowed when the archive was opened with mode *OTF2_FILEMODE_WRITE*.

Parameters

archive	Archive handle.
name	Name of the trace file property (case insensitive, [A-Z0-9_]).
value	Boolean value of property (e.g. true or false).
overwrite	If true a previous trace file property with the same name `name` will be overwritten.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_PROPERTY_NAME_INVALID if property name does not conform to the naming scheme

OTF2_ERROR_PROPERTY_NOT_FOUND if property was not found, but requested to remove

OTF2_ERROR_PROPERTY_EXISTS if property exists but overwrite was not set

E.3.4.52 OTF2_ErrorCode OTF2_Archive_SetCollectiveCallbacks (OTF2_Archive ∗ *archive,* const OTF2_CollectiveCallbacks ∗ *collectiveCallbacks,* void ∗ *collectiveData,* OTF2_CollectiveContext ∗ *globalCommContext,* OTF2_CollectiveContext ∗ *localCommContext*)

Set the collective callbacks for the archive.

This function is an collective operation.

Parameters

archive	Archive handle.
collective-Callbacks	Struct holding the collective callback functions.
collective-Data	Data passed to the collective callbacks in the `userData` argument.
global-CommCon-text	Global communication context.
local-CommCon-text	Local communication context.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.53 OTF2_ErrorCode OTF2_Archive_SetCreator (OTF2_Archive ∗ *archive,* const char ∗ *creator*)

Set creator.

Sets information about the creator of the trace archive. This value is optional. It only needs to be set for an archive handle marked as 'master' or does not need to be set at all.

Parameters

archive	Archive handle.
creator	Creator information.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.54 OTF2_ErrorCode OTF2_Archive_SetDescription (OTF2_Archive ∗ *archive*, const char ∗ *description*)

Set a description.

Sets a description for a trace archive. This value is optional. It only needs to be set for an archive handle marked as 'master' or does not need to be set at all.

Parameters

archive	Archive handle.
description	Description.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.55 OTF2_ErrorCode OTF2_Archive_SetFlushCallbacks (OTF2_Archive ∗ *archive*, const OTF2_FlushCallbacks ∗ *flushCallbacks*, void ∗ *flushData*)

Set the flush callbacks for the archive.

Parameters

archive	Archive handle.
flushCall-backs	Struct holding the flush callback functions.
flushData	Data passed to the flush callbacks in the `userData` argument.

Returns

OTF2_ErrorCode, or error code.

E.3.4.56 OTF2_ErrorCode OTF2_Archive_SetHint (OTF2_Archive ∗ *archive*, OTF2_Hint *hint*, void ∗ *value*)

Set the *hint* in the *archive* to the given *value*.

Hints can only be set once and only before OTF2 itself uses the hint the first time.

Parameters

archive	Archive handle.
hint	Name of the hint.
value	Reference to the hint value.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT in case of NULL pointers for *archive* or *value*, or an unknown *hint* value

OTF2_ERROR_HINT_INVALID in case the hint is not valid for this handle

OTF2_ERROR_HINT_LOCKED in case the hint was already set or was queried at least once by the handle

OTF2_ERROR_HINT_INVALID_VALUE in case the provided value is invalid for this hint

E.3.4.57 OTF2_ErrorCode OTF2_Archive_SetLockingCallbacks (OTF2_Archive ∗ *archive*, const OTF2_LockingCallbacks ∗ *lockingCallbacks*, void ∗ *lockingData*)

Set the locking callbacks for the archive.

Can be called any time, but only once. Before this call no thread-safety is guaranteed.

Parameters

archive	Archive handle.
locking-Callbacks	Struct holding the locking callback functions.
lockingData	Data passed to the locking callbacks in the `userData` argument.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT in case of NULL pointers for *archive* or *lockingCallbacks*, or mandatory callbacks in *lockingCallbacks* are missing

OTF2_ERROR_INVALID_CALL in case there were locking callbacks already set

E.3.4.58 OTF2_ErrorCode OTF2_Archive_SetMachineName (OTF2_Archive ∗ *archive*, const char ∗ *machineName*)

Set machine name.

Sets the name for the machine the trace was recorded. This value is optional. It only needs to be set for an archive handle marked as 'master' or does not need to be set at all.

Parameters

archive	Archive handle.
machine-Name	Machine name.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.59 OTF2_ErrorCode OTF2_Archive_SetMemoryCallbacks (OTF2_Archive ∗ *archive*, const OTF2_MemoryCallbacks ∗ *memoryCallbacks*, void ∗ *memoryData*)

Set the memory callbacks for the archive.

Parameters

archive	Archive handle.
memo-ryCallbacks	Struct holding the memory callback functions.
memory-Data	Data passed to the memory callbacks in the `userData` argument.

Returns

OTF2_ErrorCode, or error code.

E.3.4.60 OTF2_ErrorCode OTF2_Archive_SetNumberOfSnapshots (OTF2_Archive ∗ *archive*, uint32_t *number*)

Set the number of snapshots.

Parameters

archive	Archive handle.
number	Snapshot number.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.61 OTF2_ErrorCode OTF2_Archive_SetProperty (OTF2_Archive ∗ *archive*, const char ∗ *name*, const char ∗ *value*, bool *overwrite*)

Add or remove a trace file property to this archive.

Removing a trace file property is done by passing "" in the `value` parameter. The `overwrite` parameter is ignored than.

Note

This call is only allowed when the archive was opened with mode *OTF2_-FILEMODE_WRITE*.

Parameters

archive	Archive handle.
name	Name of the trace file property (case insensitive, [A-Z0-9_]).
value	Value of property.
overwrite	If true a previous trace file property with the same name `name` will be overwritten.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_PROPERTY_NAME_INVALID if property name does not conform to the naming scheme

OTF2_ERROR_PROPERTY_NOT_FOUND if property was not found, but requested to remove

OTF2_ERROR_PROPERTY_EXISTS if property exists but overwrite was not set

E.3.4.62 OTF2_ErrorCode OTF2_Archive_SetSerialCollectiveCallbacks (OTF2_Archive ∗ *archive*)

Convenient function to set the collective callbacks to an serial implementation.

Parameters

archive	Archive handle.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.3.4.63 OTF2_ErrorCode OTF2_Archive_SwitchFileMode (OTF2_Archive ∗ *archive,* OTF2_FileMode *newFileMode*)

Switch file mode of the archive.

Currently only a switch from *OTF2_FILEMODE_READ* to *OTF2_FILEMODE_-WRITE* is permitted. Currently it is also only permitted when operating on an OTF2 archive with the *OTF2_SUBSTRATE_POSIX* file substrate.

Parameters

archive	Archive handle.
newFile-Mode	New *OTF2_FileMode* to switch to.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.4 otf2/OTF2_AttributeList.h File Reference

This layer enables dynamic appending of arbitrary attributes to any type of event record.

```
#include <stdint.h>
#include <stdbool.h>
#include <otf2/OTF2_ErrorCodes.h>
```

```
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeValue.h>
```

Typedefs

- typedef struct OTF2_AttributeList_struct OTF2_AttributeList

  Attribute list handle.

Functions

- OTF2_ErrorCode OTF2_AttributeList_AddAttribute (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_Type type, OTF2_AttributeValue attributeValue)

  Add an attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddAttributeRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_AttributeRef attributeRef)

  Add an OTF2_TYPE_ATTRIBUTE attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddCallingContextRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_CallingContextRef callingContextRef)

  Add an OTF2_TYPE_CALLING_CONTEXT attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddCommRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_CommRef commRef)

  Add an OTF2_TYPE_COMM attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddDouble (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, double float64Value)

  Add an OTF2_TYPE_DOUBLE attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddFloat (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, float float32Value)

  Add an OTF2_TYPE_FLOAT attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddGroupRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_GroupRef groupRef)

  Add an OTF2_TYPE_GROUP attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddInt16 (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, int16_t int16Value)

  Add an OTF2_TYPE_INT16 attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddInt32 (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, int32_t int32Value)

Add an OTF2_TYPE_INT32 attribute to an attribute list.

- OTF2_ErrorCode OTF2_AttributeList_AddInt64 (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, int64_t int64Value)

  Add an OTF2_TYPE_INT64 attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddInt8 (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, int8_t int8Value)

  Add an OTF2_TYPE_INT8 attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddInterruptGeneratorRef (OTF2_-AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_InterruptGeneratorRef interruptGeneratorRef)

  Add an OTF2_TYPE_INTERRUPT_GENERATOR attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddLocationRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_LocationRef locationRef)

  Add an OTF2_TYPE_LOCATION attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddMetricRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_MetricRef metricRef)

  Add an OTF2_TYPE_METRIC attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddParameterRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_ParameterRef parameterRef)

  Add an OTF2_TYPE_PARAMETER attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddRegionRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_RegionRef regionRef)

  Add an OTF2_TYPE_REGION attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddRmaWinRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_RmaWinRef rmaWinRef)

  Add an OTF2_TYPE_RMA_WIN attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddSourceCodeLocationRef (OTF2_-AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_SourceCodeLocationRef sourceCodeLocationRef)

  Add an OTF2_TYPE_SOURCE_CODE_LOCATION attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddString (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_StringRef stringRef)

  Add an OTF2_STRING attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddStringRef (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_StringRef stringRef)

  Add an OTF2_TYPE_STRING attribute to an attribute list.

- OTF2_ErrorCode OTF2_AttributeList_AddUint16 (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, uint16_t uint16Value)

  Add an OTF2_TYPE_UINT16 attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddUint32 (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, uint32_t uint32Value)

  Add an OTF2_TYPE_UINT32 attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddUint64 (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, uint64_t uint64Value)

  Add an OTF2_TYPE_UINT64 attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_AddUint8 (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, uint8_t uint8Value)

  Add an OTF2_TYPE_UINT8 attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_Delete (OTF2_AttributeList ∗attributeList)

  Delete an attribute list handle.
- OTF2_ErrorCode OTF2_AttributeList_GetAttributeByID (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_Type ∗type, OTF2_AttributeValue ∗attributeValue)

  Get an attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetAttributeByIndex (const OTF2_-AttributeList ∗attributeList, uint32_t index, OTF2_AttributeRef ∗attribute, OTF2_Type ∗type, OTF2_AttributeValue ∗attributeValue)

  Get an attribute from an attribute list by attribute index.
- OTF2_ErrorCode OTF2_AttributeList_GetAttributeRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_AttributeRef ∗attributeRef)

  Get an OTF2_TYPE_ATTRIBUTE attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetCallingContextRef (const OTF2_-AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_CallingContextRef ∗callingContextRef)

  Get an OTF2_TYPE_CALLING_CONTEXT attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetCommRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_CommRef ∗commRef)

  Get an OTF2_TYPE_COMM attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetDouble (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, double ∗float64Value)

  Get an OTF2_TYPE_DOUBLE attribute from an attribute list by attribute ID.

- OTF2_ErrorCode OTF2_AttributeList_GetFloat (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, float ∗float32Value)

  Get an OTF2_TYPE_FLOAT attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetGroupRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_GroupRef ∗groupRef)

  Get an OTF2_TYPE_GROUP attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetInt16 (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, int16_t ∗int16Value)

  Get an OTF2_TYPE_INT16 attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetInt32 (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, int32_t ∗int32Value)

  Get an OTF2_TYPE_INT32 attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetInt64 (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, int64_t ∗int64Value)

  Get an OTF2_TYPE_INT64 attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetInt8 (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, int8_t ∗int8Value)

  Get an OTF2_TYPE_INT8 attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetInterruptGeneratorRef (const OTF2_-AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_InterruptGeneratorRef ∗interruptGeneratorRef)

  Get an OTF2_TYPE_INTERRUPT_GENERATOR attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetLocationRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_LocationRef ∗locationRef)

  Get an OTF2_TYPE_LOCATION attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetMetricRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_MetricRef ∗metricRef)

  Get an OTF2_TYPE_METRIC attribute from an attribute list by attribute ID.
- uint32_t OTF2_AttributeList_GetNumberOfElements (const OTF2_AttributeList ∗attributeList)

  Get the number of entries in an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_GetParameterRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_ParameterRef ∗parameterRef)

  Get an OTF2_TYPE_PARAMETER attribute from an attribute list by attribute ID.

- OTF2_ErrorCode OTF2_AttributeList_GetRegionRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_RegionRef ∗regionRef)

  Get an OTF2_TYPE_REGION attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetRmaWinRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_RmaWinRef ∗rmaWinRef)

  Get an OTF2_TYPE_RMA_WIN attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetSourceCodeLocationRef (const OTF2_-AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_SourceCodeLocationRef ∗sourceCodeLocationRef)

  Get an OTF2_TYPE_SOURCE_CODE_LOCATION attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetString (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_StringRef ∗stringRef)

  Add an OTF2_STRING attribute to an attribute list.
- OTF2_ErrorCode OTF2_AttributeList_GetStringRef (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, OTF2_StringRef ∗stringRef)

  Get an OTF2_TYPE_STRING attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetUint16 (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, uint16_t ∗uint16Value)

  Get an OTF2_TYPE_UINT16 attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetUint32 (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, uint32_t ∗uint32Value)

  Get an OTF2_TYPE_UINT32 attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetUint64 (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, uint64_t ∗uint64Value)

  Get an OTF2_TYPE_UINT64 attribute from an attribute list by attribute ID.
- OTF2_ErrorCode OTF2_AttributeList_GetUint8 (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute, uint8_t ∗uint8Value)

  Get an OTF2_TYPE_UINT8 attribute from an attribute list by attribute ID.
- OTF2_AttributeList ∗ OTF2_AttributeList_New (void)

  Create a new attribute list handle.
- OTF2_ErrorCode OTF2_AttributeList_PopAttribute (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef ∗attribute, OTF2_Type ∗type, OTF2_AttributeValue ∗attributeValue)

  Get first attribute from an attribute list and remove it.
- OTF2_ErrorCode OTF2_AttributeList_RemoveAllAttributes (OTF2_AttributeList ∗attributeList)

  Remove all attributes from an attribute list.

- OTF2_ErrorCode OTF2_AttributeList_RemoveAttribute (OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute)

  Remove an attribute from an attribute list.
- bool OTF2_AttributeList_TestAttributeByID (const OTF2_AttributeList ∗attributeList, OTF2_AttributeRef attribute)

  Test if an attribute is in the attribute list.

E.4.1 Detailed Description

This layer enables dynamic appending of arbitrary attributes to any type of event record.

Source Template:

template/OTF2_AttributeList.tmpl.h

E.4.2 Function Documentation

E.4.2.1 OTF2_ErrorCode OTF2_AttributeList_AddAttribute (OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* OTF2_Type *type,* OTF2_AttributeValue *attributeValue*)

Add an attribute to an attribute list.

Adds an attribute to an attribute list. If the attribute already exists, it fails and returns an error.

Parameters

attributeList	Attribute list handle.
attribute	Reference to attribute definition.
type	Type of the attribute.
attribute-Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.2 OTF2_ErrorCode OTF2_AttributeList_AddAttributeRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_AttributeRef *attributeRef*)

Add an OTF2_TYPE_ATTRIBUTE attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
attributeRef	Reference to *Attribute* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.3 OTF2_ErrorCode OTF2_AttributeList_AddCallingContextRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_CallingContextRef *callingContextRef*)

Add an OTF2_TYPE_CALLING_CONTEXT attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
callingContextRef	Reference to *CallingContext* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.4 OTF2_ErrorCode OTF2_AttributeList_AddCommRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_CommRef *commRef*)

Add an OTF2_TYPE_COMM attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
commRef	Reference to *Comm* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.5 OTF2_ErrorCode OTF2_AttributeList_AddDouble (OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* double *float64Value*)

Add an OTF2_TYPE_DOUBLE attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
float64Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.6 OTF2_ErrorCode OTF2_AttributeList_AddFloat (OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* float *float32Value*)

Add an OTF2_TYPE_FLOAT attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
float32Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.7 OTF2_ErrorCode OTF2_AttributeList_AddGroupRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_GroupRef *groupRef*)

Add an OTF2_TYPE_GROUP attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
groupRef	Reference to *Group* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.8 OTF2_ErrorCode OTF2_AttributeList_AddInt16 (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, int16_t *int16Value*)

Add an OTF2_TYPE_INT16 attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
int16Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.9 OTF2_ErrorCode OTF2_AttributeList_AddInt32 (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, int32_t *int32Value*)

Add an OTF2_TYPE_INT32 attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
int32Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.10 OTF2_ErrorCode OTF2_AttributeList_AddInt64 (OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* int64_t *int64Value*)

Add an OTF2_TYPE_INT64 attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
int64Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.11 OTF2_ErrorCode OTF2_AttributeList_AddInt8 (OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* int8_t *int8Value*)

Add an OTF2_TYPE_INT8 attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
int8Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.12 OTF2_ErrorCode OTF2_AttributeList_AddInterruptGeneratorRef (OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* OTF2_InterruptGeneratorRef *interruptGeneratorRef*)

Add an OTF2_TYPE_INTERRUPT_GENERATOR attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
interrupt-Generator-Ref	Reference to *InterruptGenerator* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.13 OTF2_ErrorCode OTF2_AttributeList_AddLocationRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_LocationRef *locationRef*)

Add an OTF2_TYPE_LOCATION attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
locationRef	Reference to *Location* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.14 OTF2_ErrorCode OTF2_AttributeList_AddMetricRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_MetricRef *metricRef*)

Add an OTF2_TYPE_METRIC attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
metricRef	Reference to *Metric* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.15 OTF2_ErrorCode OTF2_AttributeList_AddParameterRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_ParameterRef *parameterRef*)

Add an OTF2_TYPE_PARAMETER attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
parameter-Ref	Reference to *Parameter* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.16 OTF2_ErrorCode OTF2_AttributeList_AddRegionRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_RegionRef *regionRef*)

Add an OTF2_TYPE_REGION attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
regionRef	Reference to *Region* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.17 OTF2_ErrorCode OTF2_AttributeList_AddRmaWinRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_RmaWinRef *rmaWinRef*)

Add an OTF2_TYPE_RMA_WIN attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
rmaWinRef	Reference to *RmaWin* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.18 OTF2_ErrorCode OTF2_AttributeList_AddSourceCodeLocationRef (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_SourceCodeLocationRef *sourceCodeLocationRef*)

Add an OTF2_TYPE_SOURCE_CODE_LOCATION attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
source-CodeLoca-tionRef	Reference to *SourceCodeLocation* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.19 OTF2_ErrorCode OTF2_AttributeList_AddString (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_StringRef *stringRef*)

Add an OTF2_STRING attribute to an attribute list.

Deprecated

Use *OTF2_AttributeList_AddStringRef()* instead.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
stringRef	Reference to String definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.20 OTF2_ErrorCode OTF2_AttributeList_AddStringRef (OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, OTF2_StringRef *stringRef*)

Add an OTF2_TYPE_STRING attribute to an attribute list.

Convenience function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
stringRef	Reference to *String* definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.21 OTF2_ErrorCode OTF2_AttributeList_AddUint16 (OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, uint16_t *uint16Value*)

Add an OTF2_TYPE_UINT16 attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
uint16Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.22 OTF2_ErrorCode OTF2_AttributeList_AddUint32 (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, uint32_t *uint32Value*)

Add an OTF2_TYPE_UINT32 attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
uint32Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.23 OTF2_ErrorCode OTF2_AttributeList_AddUint64 (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, uint64_t *uint64Value*)

Add an OTF2_TYPE_UINT64 attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
uint64Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.24 OTF2_ErrorCode OTF2_AttributeList_AddUint8 (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, uint8_t *uint8Value*)

Add an OTF2_TYPE_UINT8 attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

attributeList	Attribute list handle.
attribute	Reference to *Attribute* definition.
uint8Value	Value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.25 OTF2_ErrorCode OTF2_AttributeList_Delete (OTF2_AttributeList ∗ *attributeList*)

Delete an attribute list handle.

Deletes an attribute list handle and releases all associated resources.

Parameters

attributeList	Attribute list handle.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.26 OTF2_ErrorCode OTF2_AttributeList_GetAttributeByID (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *attributeValue*)

Get an attribute from an attribute list by attribute ID.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
`out`	*type*	Returned type of the attribute.
`out`	*attribute-Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.27 OTF2_ErrorCode OTF2_AttributeList_GetAttributeByIndex (const OTF2_AttributeList ∗ *attributeList,* uint32_t *index,* OTF2_AttributeRef ∗ *attribute,* OTF2_Type ∗ *type,* OTF2_AttributeValue ∗ *attributeValue*)

Get an attribute from an attribute list by attribute index.

Parameters

	attributeList	Attribute list handle.
	index	Position of the attribute in the attribute list.
out	*attribute*	Returned attribute reference.
out	*type*	Returned type of the attribute.
out	*attribute-Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.28 OTF2_ErrorCode OTF2_AttributeList_GetAttributeRef (const OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* OTF2_AttributeRef ∗ *attributeRef*)

Get an OTF2_TYPE_ATTRIBUTE attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*attributeRef*	Returned attribute value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.29 OTF2_ErrorCode OTF2_AttributeList_GetCallingContextRef (const OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* OTF2_CallingContextRef ∗ *callingContextRef*)

Get an OTF2_TYPE_CALLING_CONTEXT attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*callingContextRef*	Returned callingContext value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.30 OTF2_ErrorCode OTF2_AttributeList_GetCommRef (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_CommRef ∗ *commRef*)

Get an OTF2_TYPE_COMM attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*commRef*	Returned comm value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.31 OTF2_ErrorCode OTF2_AttributeList_GetDouble (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, double ∗ *float64Value*)

Get an OTF2_TYPE_DOUBLE attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*float64Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.32 OTF2_ErrorCode OTF2_AttributeList_GetFloat (const OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, float * *float32Value*)

Get an OTF2_TYPE_FLOAT attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*float32Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.33 OTF2_ErrorCode OTF2_AttributeList_GetGroupRef (const OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, OTF2_GroupRef * *groupRef*)

Get an OTF2_TYPE_GROUP attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*groupRef*	Returned group value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.34 OTF2_ErrorCode OTF2_AttributeList_GetInt16 (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, int16_t ∗ *int16Value*)

Get an OTF2_TYPE_INT16 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*int16Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.35 OTF2_ErrorCode OTF2_AttributeList_GetInt32 (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, int32_t ∗ *int32Value*)

Get an OTF2_TYPE_INT32 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*int32Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.36 OTF2_ErrorCode OTF2_AttributeList_GetInt64 (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, int64_t ∗ *int64Value*)

Get an OTF2_TYPE_INT64 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*int64Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.37 OTF2_ErrorCode OTF2_AttributeList_GetInt8 (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, int8_t ∗ *int8Value*)

Get an OTF2_TYPE_INT8 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*int8Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.38 OTF2_ErrorCode OTF2_AttributeList_GetInterruptGeneratorRef (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_InterruptGeneratorRef ∗ *interruptGeneratorRef*)

Get an OTF2_TYPE_INTERRUPT_GENERATOR attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*interrupt-Generator-Ref*	Returned interruptGenerator value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.39 OTF2_ErrorCode OTF2_AttributeList_GetLocationRef (const OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, OTF2_LocationRef * *locationRef*)

Get an OTF2_TYPE_LOCATION attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*locationRef*	Returned location value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.40 OTF2_ErrorCode OTF2_AttributeList_GetMetricRef (const OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, OTF2_MetricRef * *metricRef*)

Get an OTF2_TYPE_METRIC attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*metricRef*	Returned metric value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.41 uint32_t OTF2_AttributeList_GetNumberOfElements (const OTF2_AttributeList * *attributeList*)

Get the number of entries in an attribute list.

Parameters

attributeList	Attribute list handle.

Returns

Returns the number of elements in the list. Returns zero if the list does not exist.

E.4.2.42 OTF2_ErrorCode OTF2_AttributeList_GetParameterRef (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_ParameterRef ∗ *parameterRef*)

Get an OTF2_TYPE_PARAMETER attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*parameter-Ref*	Returned parameter value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.43 OTF2_ErrorCode OTF2_AttributeList_GetRegionRef (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_RegionRef ∗ *regionRef*)

Get an OTF2_TYPE_REGION attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*regionRef*	Returned region value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.44 OTF2_ErrorCode OTF2_AttributeList_GetRmaWinRef (const OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, OTF2_RmaWinRef * *rmaWinRef*)

Get an OTF2_TYPE_RMA_WIN attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*rmaWinRef*	Returned rmaWin value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.45 OTF2_ErrorCode OTF2_AttributeList_GetSourceCodeLocationRef (const OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, OTF2_SourceCodeLocationRef * *sourceCodeLocationRef*)

Get an OTF2_TYPE_SOURCE_CODE_LOCATION attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*source-CodeLoca-tionRef*	Returned sourceCodeLocation value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.46 OTF2_ErrorCode OTF2_AttributeList_GetString (const OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, OTF2_StringRef * *stringRef*)

Add an OTF2_STRING attribute to an attribute list.

Deprecated

Use *OTF2_AttributeList_GetStringRef()* instead.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*stringRef*	Returned string value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.47 OTF2_ErrorCode OTF2_AttributeList_GetStringRef (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, OTF2_StringRef ∗ *stringRef*)

Get an OTF2_TYPE_STRING attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to attribute definition.
out	*stringRef*	Returned string value.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.48 OTF2_ErrorCode OTF2_AttributeList_GetUint16 (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*, uint16_t ∗ *uint16Value*)

Get an OTF2_TYPE_UINT16 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.

	attribute	Reference to Attribute definition.
out	*uint16Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.49 OTF2_ErrorCode OTF2_AttributeList_GetUint32 (const OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* uint32_t ∗ *uint32Value*)

Get an OTF2_TYPE_UINT32 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*uint32Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.50 OTF2_ErrorCode OTF2_AttributeList_GetUint64 (const OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* uint64_t ∗ *uint64Value*)

Get an OTF2_TYPE_UINT64 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*uint64Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.51 OTF2_ErrorCode OTF2_AttributeList_GetUint8 (const OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef *attribute,* uint8_t ∗ *uint8Value*)

Get an OTF2_TYPE_UINT8 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	attributeList	Attribute list handle.
	attribute	Reference to Attribute definition.
out	*uint8Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.52 OTF2_AttributeList∗ OTF2_AttributeList_New (void)

Create a new attribute list handle.

Returns

Returns a handle to the attribute list if successful, NULL otherwise.

E.4.2.53 OTF2_ErrorCode OTF2_AttributeList_PopAttribute (OTF2_AttributeList ∗ *attributeList,* OTF2_AttributeRef ∗ *attribute,* OTF2_Type ∗ *type,* OTF2_AttributeValue ∗ *attributeValue*)

Get first attribute from an attribute list and remove it.

Returns the first entry in the attribute list and removes it from the list.

Parameters

	attributeList	Attribute list handle.
out	*attribute*	Returned attribute reference.
out	*type*	Returned type of the attribute.
out	*attribute-Value*	Returned value of the attribute.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.54 OTF2_ErrorCode OTF2_AttributeList_RemoveAllAttributes (OTF2_AttributeList ∗ *attributeList*)

Remove all attributes from an attribute list.

Parameters

attributeList	Attribute list handle.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.55 OTF2_ErrorCode OTF2_AttributeList_RemoveAttribute (OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*)

Remove an attribute from an attribute list.

Parameters

attributeList	Attribute list handle.
attribute	Reference to Attribute definition.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.4.2.56 bool OTF2_AttributeList_TestAttributeByID (const OTF2_AttributeList ∗ *attributeList*, OTF2_AttributeRef *attribute*)

Test if an attribute is in the attribute list.

Parameters

attributeList	Attribute list handle.
attribute	Reference to Attribute definition.

Returns

True if the id is in the list, else false.

E.5 otf2/OTF2_AttributeValue.h File Reference

Declares the *OTF2_AttributeValue* and provides convenience functions to convert from and to OTF2 enum values.

```
#include <stdint.h>
#include <stdbool.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_Events.h>
```

Data Structures

- union OTF2_AttributeValue

  Value container for an attributes.

Functions

- OTF2_ErrorCode OTF2_AttributeValue_GetBoolean (OTF2_Type type, OTF2_-AttributeValue value, OTF2_Boolean ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_Boolean. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetCartPeriodicity (OTF2_Type type, OTF2_AttributeValue value, OTF2_CartPeriodicity ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_CartPeriodicity. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetCollectiveOp (OTF2_Type type, OTF2_AttributeValue value, OTF2_CollectiveOp ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_CollectiveOp. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetFileSubstrate (OTF2_Type type, OTF2_AttributeValue value, OTF2_FileSubstrate ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_FileSubstrate. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_GetFileType (OTF2_Type type, OTF2_-AttributeValue value, OTF2_FileType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_FileType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetGroupFlag (OTF2_Type type, OTF2_AttributeValue value, OTF2_GroupFlag ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_GroupFlag. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetGroupType (OTF2_Type type, OTF2_AttributeValue value, OTF2_GroupType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_GroupType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetLocationGroupType (OTF2_Type type, OTF2_AttributeValue value, OTF2_LocationGroupType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_LocationGroupType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetLocationType (OTF2_Type type, OTF2_AttributeValue value, OTF2_LocationType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_LocationType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetLockType (OTF2_Type type, OTF2_-AttributeValue value, OTF2_LockType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_LockType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetMappingType (OTF2_Type type, OTF2_AttributeValue value, OTF2_MappingType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MappingType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetMeasurementMode (OTF2_Type type, OTF2_AttributeValue value, OTF2_MeasurementMode ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MeasurementMode. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetMetricBase (OTF2_Type type, OTF2_AttributeValue value, OTF2_MetricBase ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MetricBase. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetMetricMode (OTF2_Type type, OTF2_AttributeValue value, OTF2_MetricMode ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MetricMode. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_GetMetricOccurrence (OTF2_Type type, OTF2_AttributeValue value, OTF2_MetricOccurrence ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MetricOccurrence. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetMetricScope (OTF2_Type type, OTF2_AttributeValue value, OTF2_MetricScope ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MetricScope. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetMetricTiming (OTF2_Type type, OTF2_AttributeValue value, OTF2_MetricTiming ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MetricTiming. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetMetricType (OTF2_Type type, OTF2_AttributeValue value, OTF2_MetricType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MetricType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetMetricValueProperty (OTF2_-Type type, OTF2_AttributeValue value, OTF2_MetricValueProperty ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_MetricValueProperty. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetParadigm (OTF2_Type type, OTF2_-AttributeValue value, OTF2_Paradigm ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_Paradigm. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetParadigmClass (OTF2_Type type, OTF2_AttributeValue value, OTF2_ParadigmClass ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_ParadigmClass. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetParadigmProperty (OTF2_Type type, OTF2_AttributeValue value, OTF2_ParadigmProperty ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_ParadigmProperty. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetParameterType (OTF2_Type type, OTF2_AttributeValue value, OTF2_ParameterType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_ParameterType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetRecorderKind (OTF2_Type type, OTF2_AttributeValue value, OTF2_RecorderKind ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_RecorderKind. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_GetRegionFlag (OTF2_Type type, OTF2_AttributeValue value, OTF2_RegionFlag ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_RegionFlag. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetRegionRole (OTF2_Type type, OTF2_AttributeValue value, OTF2_RegionRole ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_RegionRole. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetRmaAtomicType (OTF2_Type type, OTF2_AttributeValue value, OTF2_RmaAtomicType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_RmaAtomicType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetRmaSyncLevel (OTF2_Type type, OTF2_AttributeValue value, OTF2_RmaSyncLevel ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_RmaSyncLevel. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetRmaSyncType (OTF2_Type type, OTF2_AttributeValue value, OTF2_RmaSyncType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_RmaSyncType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetSystemTreeDomain (OTF2_Type type, OTF2_AttributeValue value, OTF2_SystemTreeDomain ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_SystemTreeDomain. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetThumbnailType (OTF2_Type type, OTF2_AttributeValue value, OTF2_ThumbnailType ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_ThumbnailType. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_GetType (OTF2_Type type, OTF2_- AttributeValue value, OTF2_Type ∗enumValue)

  Converts a OTF2_Type and OTF2_AttributeValue pair to the appropriate value for the enum OTF2_Type. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetBoolean (OTF2_Boolean enum- Value, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetCartPeriodicity (OTF2_CartPeriodicity enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetCollectiveOp (OTF2_CollectiveOp enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetFileSubstrate (OTF2_FileSubstrate enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetFileType (OTF2_FileType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetGroupFlag (OTF2_GroupFlag enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetGroupType (OTF2_GroupType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetLocationGroupType (OTF2_LocationGroupType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetLocationType (OTF2_LocationType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetLockType (OTF2_LockType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetMappingType (OTF2_MappingType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetMeasurementMode (OTF2_MeasurementMode enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetMetricBase (OTF2_MetricBase enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetMetricMode (OTF2_MetricMode enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetMetricOccurrence (OTF2_MetricOccurrence enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetMetricScope (OTF2_MetricScope enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetMetricTiming (OTF2_MetricTiming enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetMetricType (OTF2_MetricType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetMetricValueProperty (OTF2_MetricValueProperty enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetParadigm (OTF2_Paradigm enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetParadigmClass (OTF2_ParadigmClass enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetParadigmProperty (OTF2_ParadigmProperty enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetParameterType (OTF2_ParameterType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

- OTF2_ErrorCode OTF2_AttributeValue_SetRecorderKind (OTF2_RecorderKind enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetRegionFlag (OTF2_RegionFlag enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetRegionRole (OTF2_RegionRole enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetRmaAtomicType (OTF2_RmaAtomicType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetRmaSyncLevel (OTF2_RmaSyncLevel enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetRmaSyncType (OTF2_RmaSyncType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetSystemTreeDomain (OTF2_SystemTreeDomain enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetThumbnailType (OTF2_ThumbnailType enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.
- OTF2_ErrorCode OTF2_AttributeValue_SetType (OTF2_Type enumValue, OTF2_Type ∗type, OTF2_AttributeValue ∗value)

  Set OTF2_Type and OTF2_AttributeValue to the appropriate values for the given enum entry. No value range checking done.

E.5.1 Detailed Description

Declares the *OTF2_AttributeValue* and provides convenience functions to convert from and to OTF2 enum values.

Source Template:

template/OTF2_AttributeValue.tmpl.h

E.5.2 Function Documentation

E.5.2.1 OTF2_ErrorCode OTF2_AttributeValue_GetBoolean (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_Boolean ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_Boolean*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
`out`	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_Boolean*

E.5.2.2 OTF2_ErrorCode OTF2_AttributeValue_GetCartPeriodicity (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_CartPeriodicity ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_CartPeriodicity*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
`out`	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_CartPeriodicity*

E.5.2.3 OTF2_ErrorCode OTF2_AttributeValue_GetCollectiveOp (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_CollectiveOp ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_CollectiveOp*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_CollectiveOp*

E.5.2.4 OTF2_ErrorCode OTF2_AttributeValue_GetFileSubstrate (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_FileSubstrate ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_FileSubstrate*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_FileSubstrate*

E.5.2.5 OTF2_ErrorCode OTF2_AttributeValue_GetFileType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_FileType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_FileType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_FileType*

E.5.2.6 OTF2_ErrorCode OTF2_AttributeValue_GetGroupFlag (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_GroupFlag ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_GroupFlag*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_GroupFlag*

E.5.2.7 OTF2_ErrorCode OTF2_AttributeValue_GetGroupType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_GroupType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_GroupType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_GroupType*

E.5.2.8 OTF2_ErrorCode OTF2_AttributeValue_GetLocationGroupType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_LocationGroupType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_LocationGroupType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_LocationGroupType*

E.5.2.9 OTF2_ErrorCode OTF2_AttributeValue_GetLocationType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_LocationType * *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_LocationType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_LocationType*

E.5.2.10 OTF2_ErrorCode OTF2_AttributeValue_GetLockType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_LockType * *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_LockType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_LockType*

E.5.2.11 OTF2_ErrorCode OTF2_AttributeValue_GetMappingType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MappingType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MappingType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MappingType*

E.5.2.12 OTF2_ErrorCode OTF2_AttributeValue_GetMeasurementMode (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MeasurementMode ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MeasurementMode*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MeasurementMode*

E.5.2.13 OTF2_ErrorCode OTF2_AttributeValue_GetMetricBase (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MetricBase ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MetricBase*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
`out`	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MetricBase*

E.5.2.14 OTF2_ErrorCode OTF2_AttributeValue_GetMetricMode (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MetricMode ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MetricMode*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
`out`	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MetricMode*

E.5.2.15 OTF2_ErrorCode OTF2_AttributeValue_GetMetricOccurrence (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MetricOccurrence * *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MetricOccurrence*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MetricOccurrence*

E.5.2.16 OTF2_ErrorCode OTF2_AttributeValue_GetMetricScope (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MetricScope * *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MetricScope*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MetricScope*

E.5.2.17 OTF2_ErrorCode OTF2_AttributeValue_GetMetricTiming (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MetricTiming ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MetricTiming*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MetricTiming*

E.5.2.18 OTF2_ErrorCode OTF2_AttributeValue_GetMetricType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MetricType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MetricType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MetricType*

E.5.2.19 OTF2_ErrorCode OTF2_AttributeValue_GetMetricValueProperty (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_MetricValueProperty ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_MetricValueProperty*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_MetricValueProperty*

E.5.2.20 OTF2_ErrorCode OTF2_AttributeValue_GetParadigm (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_Paradigm ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_Paradigm*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_Paradigm*

E.5.2.21 OTF2_ErrorCode OTF2_AttributeValue_GetParadigmClass (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_ParadigmClass * *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_ParadigmClass*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
`out`	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_ParadigmClass*

E.5.2.22 OTF2_ErrorCode OTF2_AttributeValue_GetParadigmProperty (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_ParadigmProperty * *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_ParadigmProperty*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
`out`	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_ParadigmProperty*

E.5.2.23 OTF2_ErrorCode OTF2_AttributeValue_GetParameterType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_ParameterType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_ParameterType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_ParameterType*

E.5.2.24 OTF2_ErrorCode OTF2_AttributeValue_GetRecorderKind (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_RecorderKind ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_RecorderKind*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_RecorderKind*

E.5.2.25 OTF2_ErrorCode OTF2_AttributeValue_GetRegionFlag (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_RegionFlag ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_RegionFlag*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_RegionFlag*

E.5.2.26 OTF2_ErrorCode OTF2_AttributeValue_GetRegionRole (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_RegionRole ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_RegionRole*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_RegionRole*

E.5.2.27 OTF2_ErrorCode OTF2_AttributeValue_GetRmaAtomicType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_RmaAtomicType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_RmaAtomicType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_RmaAtomicType*

E.5.2.28 OTF2_ErrorCode OTF2_AttributeValue_GetRmaSyncLevel (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_RmaSyncLevel ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_RmaSyncLevel*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_RmaSyncLevel*

E.5.2.29 OTF2_ErrorCode OTF2_AttributeValue_GetRmaSyncType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_RmaSyncType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_RmaSyncType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
`out`	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_RmaSyncType*

E.5.2.30 OTF2_ErrorCode OTF2_AttributeValue_GetSystemTreeDomain (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_SystemTreeDomain ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_SystemTreeDomain*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
`out`	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_SystemTreeDomain*

E.5.2.31 OTF2_ErrorCode OTF2_AttributeValue_GetThumbnailType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_ThumbnailType ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_ThumbnailType*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_ThumbnailType*

E.5.2.32 OTF2_ErrorCode OTF2_AttributeValue_GetType (OTF2_Type *type*, OTF2_AttributeValue *value*, OTF2_Type ∗ *enumValue*)

Converts a *OTF2_Type* and *OTF2_AttributeValue* pair to the appropriate value for the enum *OTF2_Type*. No value range checking done.

Parameters

	type	Given type.
	value	Given value.
out	*enumValue*	Converted enum value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if *enumValue* a NULL pointer

OTF2_ERROR_INVALID_ATTRIBUTE_TYPE if *type* does not match the base type of the enum *OTF2_Type*

E.5.2.33 OTF2_ErrorCode OTF2_AttributeValue_SetBoolean (OTF2_Boolean *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_Boolean*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.34 OTF2_ErrorCode OTF2_AttributeValue_SetCartPeriodicity (OTF2_CartPeriodicity *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_CartPeriodicity*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.35 OTF2_ErrorCode OTF2_AttributeValue_SetCollectiveOp (OTF2_CollectiveOp *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_CollectiveOp*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.36 OTF2_ErrorCode OTF2_AttributeValue_SetFileSubstrate (OTF2_FileSubstrate *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_FileSubstrate*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.37 OTF2_ErrorCode OTF2_AttributeValue_SetFileType (OTF2_FileType *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_FileType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.38 OTF2_ErrorCode OTF2_AttributeValue_SetGroupFlag (OTF2_GroupFlag *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_GroupFlag*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.39 OTF2_ErrorCode OTF2_AttributeValue_SetGroupType (OTF2_GroupType *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_GroupType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.40 OTF2_ErrorCode OTF2_AttributeValue_SetLocationGroupType (OTF2_LocationGroupType *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_LocationGroupType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.41 OTF2_ErrorCode OTF2_AttributeValue_SetLocationType (OTF2_LocationType *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_LocationType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.42 OTF2_ErrorCode OTF2_AttributeValue_SetLockType (OTF2_LockType *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_LockType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.43 OTF2_ErrorCode OTF2_AttributeValue_SetMappingType (OTF2_MappingType *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MappingType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.44 OTF2_ErrorCode OTF2_AttributeValue_SetMeasurementMode (OTF2_MeasurementMode *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MeasurementMode*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.45 OTF2_ErrorCode OTF2_AttributeValue_SetMetricBase (OTF2_MetricBase *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MetricBase*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.46 OTF2_ErrorCode OTF2_AttributeValue_SetMetricMode (OTF2_MetricMode *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MetricMode*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.47 OTF2_ErrorCode OTF2_AttributeValue_SetMetricOccurrence (OTF2_MetricOccurrence *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MetricOccurrence*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.48 OTF2_ErrorCode OTF2_AttributeValue_SetMetricScope (OTF2_MetricScope *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MetricScope*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.49 OTF2_ErrorCode OTF2_AttributeValue_SetMetricTiming (OTF2_MetricTiming *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MetricTiming*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.50 OTF2_ErrorCode OTF2_AttributeValue_SetMetricType (OTF2_MetricType *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MetricType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.51 OTF2_ErrorCode OTF2_AttributeValue_SetMetricValueProperty (OTF2_MetricValueProperty *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_MetricValueProperty*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.52 OTF2_ErrorCode OTF2_AttributeValue_SetParadigm (OTF2_Paradigm *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_Paradigm*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.53 OTF2_ErrorCode OTF2_AttributeValue_SetParadigmClass (OTF2_ParadigmClass *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_ParadigmClass*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.54 OTF2_ErrorCode OTF2_AttributeValue_SetParadigmProperty (OTF2_ParadigmProperty *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
`out`	*type*	Matching *OTF2_Type* for the enum *OTF2_ParadigmProperty*.
`out`	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.55 OTF2_ErrorCode OTF2_AttributeValue_SetParameterType (OTF2_ParameterType *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
`out`	*type*	Matching *OTF2_Type* for the enum *OTF2_ParameterType*.
`out`	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.56 OTF2_ErrorCode OTF2_AttributeValue_SetRecorderKind (OTF2_RecorderKind *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_RecorderKind*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.57 OTF2_ErrorCode OTF2_AttributeValue_SetRegionFlag (OTF2_RegionFlag *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_RegionFlag*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.58 OTF2_ErrorCode OTF2_AttributeValue_SetRegionRole (OTF2_RegionRole *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_RegionRole*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.59 OTF2_ErrorCode OTF2_AttributeValue_SetRmaAtomicType (OTF2_RmaAtomicType *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_RmaAtomicType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.60 OTF2_ErrorCode OTF2_AttributeValue_SetRmaSyncLevel (OTF2_RmaSyncLevel *enumValue*, OTF2_Type ∗ *type*, OTF2_AttributeValue ∗ *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_RmaSyncLevel*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.61 OTF2_ErrorCode OTF2_AttributeValue_SetRmaSyncType (OTF2_RmaSyncType *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_RmaSyncType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.62 OTF2_ErrorCode OTF2_AttributeValue_SetSystemTreeDomain (OTF2_SystemTreeDomain *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_SystemTreeDomain*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.63 OTF2_ErrorCode OTF2_AttributeValue_SetThumbnailType (OTF2_ThumbnailType *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_ThumbnailType*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.5.2.64 OTF2_ErrorCode OTF2_AttributeValue_SetType (OTF2_Type *enumValue*, OTF2_Type * *type*, OTF2_AttributeValue * *value*)

Set *OTF2_Type* and *OTF2_AttributeValue* to the appropriate values for the given enum entry. No value range checking done.

Parameters

	enumValue	The enum value to be converted.
out	*type*	Matching *OTF2_Type* for the enum *OTF2_Type*.
out	*value*	Matching *OTF2_AttributeValue* for the *enumValue* value.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT if either *type* or *value* are NULL pointer

E.6 otf2/OTF2_Callbacks.h File Reference

This header file provides all user callbacks.

```
#include <stdbool.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
```

Data Structures

- struct OTF2_CollectiveCallbacks

  Struct which holds all collective callbacks.
- struct OTF2_FlushCallbacks

  Structure holding the flush callbacks.
- struct OTF2_LockingCallbacks

  Struct which holds all collective callbacks.
- struct OTF2_MemoryCallbacks

  Structure holding the memory callbacks.

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Barrier)(void ∗userData, OTF2_CollectiveContext ∗commContext)

  Performs an barrier collective on the given communication context.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Bcast)(void ∗userData, OTF2_CollectiveContext ∗commContext, void ∗data, uint32_t numberElements, OTF2_Type type, uint32_t root)

  Performs an broadcast collective on the given communication context.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_CreateLocalComm)(void ∗userData, OTF2_CollectiveContext ∗∗localCommContext, OTF2_CollectiveContext ∗globalCommContext, uint32_t globalRank, uint32_t globalSize, uint32_t localRank, uint32_t localSize, uint32_t fileNumber, uint32_t numberOfFiles)

  Create a new disjoint partitioning of the the globalCommContext communication context. numberOfFiles denotes the number of the partitions. fileNumber denotes in which of the partitions this OTF2_Archive should belong. localSize is the size of this partition and localRank the rank of this OTF2_Archive in the partition.
- typedef OTF2_CallbackCode(∗ OTF2_Collectives_FreeLocalComm)(void ∗userData, OTF2_CollectiveContext ∗localCommContext)

Destroys the communication context previous created by the OTF2_Collectives_-CreateLocalComm callback.

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Gather)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, void ∗outData, uint32_t numberElements, OTF2_Type type, uint32_t root)

  Performs an gather collective on the given communication context where each ranks contribute the same number of elements. outData is only valid at rank root.

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Gatherv)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, uint32_t inElements, void ∗outData, const uint32_t ∗outElements, OTF2_Type type, uint32_t root)

  Performs an gather collective on the given communication context where each ranks contribute different number of elements. outData and outElements are only valid at rank root.

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_GetRank)(void ∗userData, OTF2_CollectiveContext ∗commContext, uint32_t ∗rank)

  Returns the rank of this OTF2_Archive objects in this communication context. A number between 0 and one less of the size of the communication context.

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_GetSize)(void ∗userData, OTF2_CollectiveContext ∗commContext, uint32_t ∗size)

  Returns the number of OTF2_Archive objects operating in this communication context.

- typedef void(∗ OTF2_Collectives_Release)(void ∗userData, OTF2_CollectiveContext ∗globalCommContext, OTF2_CollectiveContext ∗localCommContext)

  Optionally called in OTF2_Archive_Close or OTF2_Reader_Close respectively.

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Scatter)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, void ∗outData, uint32_t numberElements, OTF2_Type type, uint32_t root)

  Performs an scatter collective on the given communication context where each ranks contribute the same number of elements. inData is only valid at rank root.

- typedef OTF2_CallbackCode(∗ OTF2_Collectives_Scatterv)(void ∗userData, OTF2_CollectiveContext ∗commContext, const void ∗inData, const uint32_t ∗inElements, void ∗outData, uint32_t outElements, OTF2_Type type, uint32_t root)

  Performs an scatter collective on the given communication context where each ranks contribute different number of elements. inData and inElements are only valid at rank root.

- typedef OTF2_CallbackCode(∗ OTF2_Locking_Create)(void ∗userData, OTF2_Lock ∗lock)

  Creates a new locking object.

- typedef OTF2_CallbackCode(∗ OTF2_Locking_Destroy)(void ∗userData, OTF2_Lock lock)

Destroys a locking object.

- typedef OTF2_CallbackCode(∗ OTF2_Locking_Lock)(void ∗userData, OTF2_-Lock lock)

  Locks a locking object.
- typedef void(∗ OTF2_Locking_Release)(void ∗userData)

  Optionally called in OTF2_Archive_Close or OTF2_Reader_Close respectively.
- typedef OTF2_CallbackCode(∗ OTF2_Locking_Unlock)(void ∗userData, OTF2_Lock lock)

  Unlocks a locking object.
- typedef void ∗(∗ OTF2_MemoryAllocate)(void ∗userData, OTF2_FileType fileType, OTF2_LocationRef location, void ∗∗perBufferData, uint64_t chunk-Size)

  Function pointer for allocating memory for chunks.
- typedef void(∗ OTF2_MemoryFreeAll)(void ∗userData, OTF2_FileType file-Type, OTF2_LocationRef location, void ∗∗perBufferData, bool final)

  Function pointer to release all allocated chunks.
- typedef OTF2_TimeStamp(∗ OTF2_PostFlushCallback)(void ∗userData, OTF2_-FileType fileType, OTF2_LocationRef location)

  Definition for the post flush callback.
- typedef OTF2_FlushType(∗ OTF2_PreFlushCallback)(void ∗userData, OTF2_-FileType fileType, OTF2_LocationRef location, void ∗callerData, bool fi-nal)

  Definition for the pre flush callback.

E.6.1 Detailed Description

This header file provides all user callbacks.

E.7 otf2/OTF2_Definitions.h File Reference

Data types used in the definition records.

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
```

Typedefs

- typedef uint8_t OTF2_CartPeriodicity

  Wrapper for enum OTF2_CartPeriodicity_enum.

- typedef uint32_t OTF2_GroupFlag

  Wrapper for enum OTF2_GroupFlag_enum.
- typedef uint8_t OTF2_GroupType

  Wrapper for enum OTF2_GroupType_enum.
- typedef uint8_t OTF2_LocationGroupType

  Wrapper for enum OTF2_LocationGroupType_enum.
- typedef uint8_t OTF2_LocationType

  Wrapper for enum OTF2_LocationType_enum.
- typedef uint8_t OTF2_MetricBase

  Wrapper for enum OTF2_MetricBase_enum.
- typedef uint8_t OTF2_MetricMode

  Wrapper for enum OTF2_MetricMode_enum.
- typedef uint8_t OTF2_MetricOccurrence

  Wrapper for enum OTF2_MetricOccurrence_enum.
- typedef uint8_t OTF2_MetricScope

  Wrapper for enum OTF2_MetricScope_enum.
- typedef uint8_t OTF2_MetricTiming

  Wrapper for enum OTF2_MetricTiming_enum.
- typedef uint8_t OTF2_MetricType

  Wrapper for enum OTF2_MetricType_enum.
- typedef uint8_t OTF2_MetricValueProperty

  Wrapper for enum OTF2_MetricValueProperty_enum.
- typedef uint8_t OTF2_ParameterType

  Wrapper for enum OTF2_ParameterType_enum.
- typedef uint8_t OTF2_RecorderKind

  Wrapper for enum OTF2_RecorderKind_enum.
- typedef uint32_t OTF2_RegionFlag

  Wrapper for enum OTF2_RegionFlag_enum.
- typedef uint8_t OTF2_RegionRole

  Wrapper for enum OTF2_RegionRole_enum.
- typedef uint8_t OTF2_SystemTreeDomain

  Wrapper for enum OTF2_SystemTreeDomain_enum.

Enumerations

- enum OTF2_CartPeriodicity_enum {

  OTF2_CART_PERIODIC_FALSE = 0,

  OTF2_CART_PERIODIC_TRUE = 1 }

Periodicity types of a cartesian topology dimension.

- enum OTF2_GroupFlag_enum {

  OTF2_GROUP_FLAG_NONE = 0,

  OTF2_GROUP_FLAG_GLOBAL_MEMBERS = (1 << 0) }

  List of possible flags to specify special characteristics of a Group.
- enum OTF2_GroupType_enum {

  OTF2_GROUP_TYPE_UNKNOWN = 0,

  OTF2_GROUP_TYPE_LOCATIONS = 1,

  OTF2_GROUP_TYPE_REGIONS = 2,

  OTF2_GROUP_TYPE_METRIC = 3,

  OTF2_GROUP_TYPE_COMM_LOCATIONS = 4,

  OTF2_GROUP_TYPE_COMM_GROUP = 5,

  OTF2_GROUP_TYPE_COMM_SELF = 6 }

  List of available group types.
- enum OTF2_LocationGroupType_enum {

  OTF2_LOCATION_GROUP_TYPE_UNKNOWN = 0,

  OTF2_LOCATION_GROUP_TYPE_PROCESS = 1 }

  List of possible definitions of type LocationGroup.
- enum OTF2_LocationType_enum {

  OTF2_LOCATION_TYPE_UNKNOWN = 0,

  OTF2_LOCATION_TYPE_CPU_THREAD = 1,

  OTF2_LOCATION_TYPE_GPU = 2,

  OTF2_LOCATION_TYPE_METRIC = 3 }

  List of possible definitions of type Location.
- enum OTF2_MetricBase_enum {

  OTF2_BASE_BINARY = 0,

  OTF2_BASE_DECIMAL = 1 }

  Metric base types.
- enum OTF2_MetricMode_enum {

  OTF2_METRIC_ACCUMULATED_START = OTF2_METRIC_VALUE_-ACCUMULATED | OTF2_METRIC_TIMING_START,

  OTF2_METRIC_ACCUMULATED_POINT = OTF2_METRIC_VALUE_-ACCUMULATED | OTF2_METRIC_TIMING_POINT,

  OTF2_METRIC_ACCUMULATED_LAST = OTF2_METRIC_VALUE_ACCUMULATED | OTF2_METRIC_TIMING_LAST, OTF2_METRIC_ACCUMULATED_NEXT = OTF2_METRIC_VALUE_-ACCUMULATED | OTF2_METRIC_TIMING_NEXT,

OTF2_METRIC_ABSOLUTE_POINT = OTF2_METRIC_VALUE_ABSOLUTE | OTF2_METRIC_TIMING_POINT,

OTF2_METRIC_ABSOLUTE_LAST = OTF2_METRIC_VALUE_ABSOLUTE | OTF2_METRIC_TIMING_LAST,

OTF2_METRIC_ABSOLUTE_NEXT = OTF2_METRIC_VALUE_ABSOLUTE | OTF2_METRIC_TIMING_NEXT,

OTF2_METRIC_RELATIVE_POINT = OTF2_METRIC_VALUE_RELATIVE | OTF2_METRIC_TIMING_POINT,

OTF2_METRIC_RELATIVE_LAST = OTF2_METRIC_VALUE_RELATIVE | OTF2_METRIC_TIMING_LAST,

OTF2_METRIC_RELATIVE_NEXT = OTF2_METRIC_VALUE_RELATIVE | OTF2_METRIC_TIMING_NEXT }

Metric mode is a combination of value property and timing information.

- enum OTF2_MetricOccurrence_enum {

  OTF2_METRIC_SYNCHRONOUS_STRICT = 0,

  OTF2_METRIC_SYNCHRONOUS = 1,

  OTF2_METRIC_ASYNCHRONOUS = 2 }

  Metric occurrence.

- enum OTF2_MetricScope_enum {

  OTF2_SCOPE_LOCATION = 0,

  OTF2_SCOPE_LOCATION_GROUP = 1,

  OTF2_SCOPE_SYSTEM_TREE_NODE = 2,

  OTF2_SCOPE_GROUP = 3 }

  List of available metric scopes.

- enum OTF2_MetricTiming_enum {

  OTF2_METRIC_TIMING_START = 0,

  OTF2_METRIC_TIMING_POINT = 1 << 4,

  OTF2_METRIC_TIMING_LAST = 2 << 4,

  OTF2_METRIC_TIMING_NEXT = 3 << 4,

  OTF2_METRIC_TIMING_MASK = 240 }

  Determines when the values have been collected or for which interval of time they are valid. Used for the upper half-byte of OTF2_MetricMode.

- enum OTF2_MetricType_enum {
  OTF2_METRIC_TYPE_OTHER = 0,
  OTF2_METRIC_TYPE_PAPI = 1,
  OTF2_METRIC_TYPE_RUSAGE = 2,
  OTF2_METRIC_TYPE_USER = 3 }

  List of available metric types.
- enum OTF2_MetricValueProperty_enum {
  OTF2_METRIC_VALUE_ACCUMULATED = 0,
  OTF2_METRIC_VALUE_ABSOLUTE = 1,
  OTF2_METRIC_VALUE_RELATIVE = 2,
  OTF2_METRIC_VALUE_MASK = 15 }

  Information about whether the metric value is accumulated, absolute, or relative. Used for the lower half-byte of OTF2_MetricMode.
- enum OTF2_ParameterType_enum {
  OTF2_PARAMETER_TYPE_STRING = 0,
  OTF2_PARAMETER_TYPE_INT64 = 1,
  OTF2_PARAMETER_TYPE_UINT64 = 2 }

  List of possible for definitions of type Parameter.
- enum OTF2_RecorderKind_enum {
  OTF2_RECORDER_KIND_UNKNOWN = 0,
  OTF2_RECORDER_KIND_ABSTRACT = 1,
  OTF2_RECORDER_KIND_CPU = 2,
  OTF2_RECORDER_KIND_GPU = 3 }

  List of possible kinds a MetricClass can be recorded by.
- enum OTF2_RegionFlag_enum {
  OTF2_REGION_FLAG_NONE = 0,
  OTF2_REGION_FLAG_DYNAMIC = (1 << 0),
  OTF2_REGION_FLAG_PHASE = (1 << 1) }

  List of possible flags to specify special characteristics of a Region.
- enum OTF2_RegionRole_enum {
  OTF2_REGION_ROLE_UNKNOWN = 0,
  OTF2_REGION_ROLE_FUNCTION = 1,
  OTF2_REGION_ROLE_WRAPPER = 2,
  OTF2_REGION_ROLE_LOOP = 3,
  OTF2_REGION_ROLE_CODE = 4,

OTF2_REGION_ROLE_PARALLEL = 5,

OTF2_REGION_ROLE_SECTIONS = 6,

OTF2_REGION_ROLE_SECTION = 7,

OTF2_REGION_ROLE_WORKSHARE = 8,

OTF2_REGION_ROLE_SINGLE = 9,

OTF2_REGION_ROLE_SINGLE_SBLOCK = 10,

OTF2_REGION_ROLE_MASTER = 11,

OTF2_REGION_ROLE_CRITICAL = 12,

OTF2_REGION_ROLE_CRITICAL_SBLOCK = 13,

OTF2_REGION_ROLE_ATOMIC = 14,

OTF2_REGION_ROLE_BARRIER = 15,

OTF2_REGION_ROLE_IMPLICIT_BARRIER = 16,

OTF2_REGION_ROLE_FLUSH = 17,

OTF2_REGION_ROLE_ORDERED = 18,

OTF2_REGION_ROLE_ORDERED_SBLOCK = 19,

OTF2_REGION_ROLE_TASK = 20,

OTF2_REGION_ROLE_TASK_CREATE = 21,

OTF2_REGION_ROLE_TASK_WAIT = 22,

OTF2_REGION_ROLE_COLL_ONE2ALL = 23,

OTF2_REGION_ROLE_COLL_ALL2ONE = 24,

OTF2_REGION_ROLE_COLL_ALL2ALL = 25,

OTF2_REGION_ROLE_COLL_OTHER = 26,

OTF2_REGION_ROLE_FILE_IO = 27,

OTF2_REGION_ROLE_POINT2POINT = 28,

OTF2_REGION_ROLE_RMA = 29,

OTF2_REGION_ROLE_DATA_TRANSFER = 30,

OTF2_REGION_ROLE_ARTIFICIAL = 31,

OTF2_REGION_ROLE_THREAD_CREATE = 32,

OTF2_REGION_ROLE_THREAD_WAIT = 33,

OTF2_REGION_ROLE_TASK_UNTIED = 34 }

List of possible roles of a Region.

- enum OTF2_SystemTreeDomain_enum {
  OTF2_SYSTEM_TREE_DOMAIN_MACHINE = 0,
  OTF2_SYSTEM_TREE_DOMAIN_SHARED_MEMORY = 1,
  OTF2_SYSTEM_TREE_DOMAIN_NUMA = 2,
  OTF2_SYSTEM_TREE_DOMAIN_SOCKET = 3,
  OTF2_SYSTEM_TREE_DOMAIN_CACHE = 4,
  OTF2_SYSTEM_TREE_DOMAIN_CORE = 5,
  OTF2_SYSTEM_TREE_DOMAIN_PU = 6 }

  List of available system tree node domains.

E.7.1 Detailed Description

Data types used in the definition records.

Source Template:

templates/OTF2_Definitions.tmpl.h

E.7.2 Enumeration Type Documentation

E.7.2.1 enum OTF2_CartPeriodicity_enum

Periodicity types of a cartesian topology dimension.

Since

Version 1.0

Enumerator:

OTF2_CART_PERIODIC_FALSE	Dimension is not periodic.
OTF2_CART_PERIODIC_TRUE	Dimension is periodic.

E.7.2.2 enum OTF2_GroupFlag_enum

List of possible flags to specify special characteristics of a Group.

Since

Version 1.2

Enumerator:

OTF2_GROUP_FLAG_NONE A group without special characterization.

OTF2_GROUP_FLAG_GLOBAL_MEMBERS No translation of ranks in event records needs to be done when a group of type *OTF2_GROUP_-TYPE_COMM_GROUP* has this flag. I.e., the ranks are indexes into the the *OTF2_GROUP_TYPE_COMM_LOCATIONS* group.

E.7.2.3 enum OTF2_GroupType_enum

List of available group types.

Since

Version 1.2

Enumerator:

OTF2_GROUP_TYPE_UNKNOWN Group of unknown type.

OTF2_GROUP_TYPE_LOCATIONS Group of locations.

OTF2_GROUP_TYPE_REGIONS Group of regions.

OTF2_GROUP_TYPE_METRIC Group of metrics.

OTF2_GROUP_TYPE_COMM_LOCATIONS List of locations which participated in the pardigm specified by the group definition. For example: In case of MPI, the size of this group should match the size of *MPI_-COMM_WORLD*. Each entry in the list is a *Location* reference, where the index of the entry is equal to the rank in *MPI_COMM_WORLD* (i.e., rank *i* corresponds to location *members[i]*).

Also, if this definition is present, the location group ids of locations with type *OTF2_LOCATION_TYPE_CPU_THREAD* should match the MPI rank.

This group needs to be defined, before any group of type *OTF2_GROUP_-TYPE_COMM_GROUP* and the same paradigm.

Note: This does not makes sense in local definitions.

OTF2_GROUP_TYPE_COMM_GROUP A sub-group of the corresponding group definition with type *OTF2_GROUP_TYPE_COMM_LOCATIONS* and the same paradigm. The sub-group is formed by listing the indexes of the *OTF2_GROUP_TYPE_COMM_LOCATIONS* group.

OTF2_GROUP_TYPE_COMM_SELF Special group type to efficiently handle self-like communicators (i.e., MPI_COMM_SELF and friends). At most one of this definition is allowed to exists per paradigm.

E.7.2.4 enum OTF2_LocationGroupType_enum

List of possible definitions of type LocationGroup.

Since

Version 1.0

Enumerator:

OTF2_LOCATION_GROUP_TYPE_UNKNOWN A location group of unknown type.

OTF2_LOCATION_GROUP_TYPE_PROCESS A process.

E.7.2.5 enum OTF2_LocationType_enum

List of possible definitions of type Location.

Since

Version 1.0

Enumerator:

OTF2_LOCATION_TYPE_UNKNOWN A location of unknown type.

OTF2_LOCATION_TYPE_CPU_THREAD A CPU thread.

OTF2_LOCATION_TYPE_GPU A GPU location.

OTF2_LOCATION_TYPE_METRIC A metric only location e.g. an external device.

E.7.2.6 enum OTF2_MetricBase_enum

Metric base types.

Since

Version 1.0

Enumerator:

OTF2_BASE_BINARY Binary base.

OTF2_BASE_DECIMAL Decimal base.

E.7.2.7 enum OTF2_MetricMode_enum

Metric mode is a combination of value property and timing information.

Since

Version 1.0

Enumerator:

OTF2_METRIC_ACCUMULATED_START Accumulated metric, ’START’ timing.

OTF2_METRIC_ACCUMULATED_POINT Accumulated metric, ’POINT’ timing.

OTF2_METRIC_ACCUMULATED_LAST Accumulated metric, ’LAST’ timing.

OTF2_METRIC_ACCUMULATED_NEXT Accumulated metric, ’NEXT’ timing.

OTF2_METRIC_ABSOLUTE_POINT Absolute metric, ’POINT’ timing.

OTF2_METRIC_ABSOLUTE_LAST Absolute metric, ’LAST’ timing.

OTF2_METRIC_ABSOLUTE_NEXT Absolute metric, ’NEXT’ timing.

OTF2_METRIC_RELATIVE_POINT Relative metric, ’POINT’ timing.

OTF2_METRIC_RELATIVE_LAST Relative metric, ’LAST’ timing.

OTF2_METRIC_RELATIVE_NEXT Relative metric, ’NEXT’ timing.

E.7.2.8 enum OTF2_MetricOccurrence_enum

Metric occurrence.

Since

Version 1.0

Enumerator:

OTF2_METRIC_SYNCHRONOUS_STRICT Metric occurs at every region enter and leave.

OTF2_METRIC_SYNCHRONOUS Metric occurs only at a region enter and leave, but does not need to occur at every enter/leave.

OTF2_METRIC_ASYNCHRONOUS Metric can occur at any place i.e. it is not related to region enter and leaves.

E.7.2.9 enum OTF2_MetricScope_enum

List of available metric scopes.

Since

Version 1.0

Enumerator:

OTF2_SCOPE_LOCATION Scope of a metric is another location.

OTF2_SCOPE_LOCATION_GROUP Scope of a metric is a location group.

OTF2_SCOPE_SYSTEM_TREE_NODE Scope of a metric is a system tree node.

OTF2_SCOPE_GROUP Scope of a metric is a generic group of locations.

E.7.2.10 enum OTF2_MetricTiming_enum

Determines when the values have been collected or for which interval of time they are valid. Used for the upper half-byte of OTF2_MetricMode.

Since

Version 1.0

Enumerator:

OTF2_METRIC_TIMING_START Metric value belongs to the time interval since the beginning of the measurement.

OTF2_METRIC_TIMING_POINT Metric value is only valid at a point in time but not necessarily for any interval of time.

OTF2_METRIC_TIMING_LAST Metric value is related to the time interval since the last counter sample of the same metric, i.e. the immediate past.

OTF2_METRIC_TIMING_NEXT Metric value is valid from now until the next counter sample, i.e. the future right ahead.

OTF2_METRIC_TIMING_MASK This mask can be used to get the upper half-byte in OTF2_MetricMode that is used to indicate metric timing information.

E.7.2.11 enum OTF2_MetricType_enum

List of available metric types.

Since

Version 1.0

Enumerator:

OTF2_METRIC_TYPE_OTHER Any metric of a type not explicitly listed below.

OTF2_METRIC_TYPE_PAPI PAPI counter.

OTF2_METRIC_TYPE_RUSAGE Resource usage counter.

OTF2_METRIC_TYPE_USER User metrics.

E.7.2.12 enum OTF2_MetricValueProperty_enum

Information about whether the metric value is accumulated, absolute, or relative. Used for the lower half-byte of OTF2_MetricMode.

Since

Version 1.0

Enumerator:

OTF2_METRIC_VALUE_ACCUMULATED Accumulated metric is monotonously increasing (i.e., PAPI counter for number of executed floating point operations).

OTF2_METRIC_VALUE_ABSOLUTE Absolute metric (i.e., temperature, rate, mean value, etc.).

OTF2_METRIC_VALUE_RELATIVE Relative metric.

OTF2_METRIC_VALUE_MASK This mask can be used to get lower half-byte in OTF2_MetricMode that is used to indicate metric value property.

E.7.2.13 enum OTF2_ParameterType_enum

List of possible for definitions of type Parameter.

Since

Version 1.0

Enumerator:

OTF2_PARAMETER_TYPE_STRING Parameter is of type string.

OTF2_PARAMETER_TYPE_INT64 Parameter is of type signed 8-byte integer.

OTF2_PARAMETER_TYPE_UINT64 Parameter is of type unsigned 8-byte integer.

E.7.2.14 enum OTF2_RecorderKind_enum

List of possible kinds a MetricClass can be recorded by.

Since

Version 1.2

Enumerator:

OTF2_RECORDER_KIND_UNKNOWN No specific kind of recorder.

OTF2_RECORDER_KIND_ABSTRACT The metric class will only be recorded via a *MetricInstance* definitions.

OTF2_RECORDER_KIND_CPU This metric class will only be recored by locations of type *OTF2_LOCATION_TYPE_CPU_THREAD*.

OTF2_RECORDER_KIND_GPU This metric class will only be recored by locations of type *OTF2_LOCATION_TYPE_GPU*.

E.7.2.15 enum OTF2_RegionFlag_enum

List of possible flags to specify special characteristics of a Region.

Since

Version 1.1

Enumerator:

OTF2_REGION_FLAG_NONE A region without special characterization.

OTF2_REGION_FLAG_DYNAMIC Each time this region is entered it will get an individual call path in the profile.

OTF2_REGION_FLAG_PHASE Each time this region is entered it will get an individual root node in the profile.

E.7.2.16 enum OTF2_RegionRole_enum

List of possible roles of a Region.

Since

Version 1.1

Enumerator:

OTF2_REGION_ROLE_UNKNOWN A region of unknown role.

OTF2_REGION_ROLE_FUNCTION An entire function/subroutine.

OTF2_REGION_ROLE_WRAPPER An API function wrapped by Score-P.

OTF2_REGION_ROLE_LOOP A loop in the code.

OTF2_REGION_ROLE_CODE An arbitrary section of code.

OTF2_REGION_ROLE_PARALLEL E.g. OpenMP "parallel" construct (structured block)

OTF2_REGION_ROLE_SECTIONS E.g. OpenMP "sections" construct.

OTF2_REGION_ROLE_SECTION Individual "section" inside an OpenMP "sections" construct.

OTF2_REGION_ROLE_WORKSHARE E.g. OpenMP "workshare" construct.

OTF2_REGION_ROLE_SINGLE E.g. OpenMP "single" construct.

OTF2_REGION_ROLE_SINGLE_SBLOCK E.g. OpenMP "single" construct (structured block)

OTF2_REGION_ROLE_MASTER E.g. OpenMP "master" construct.

OTF2_REGION_ROLE_CRITICAL E.g. OpenMP "critical" construct.

OTF2_REGION_ROLE_CRITICAL_SBLOCK E.g. OpenMP "critical" construct (structured block)

OTF2_REGION_ROLE_ATOMIC E.g. OpenMP "atomic" construct.

OTF2_REGION_ROLE_BARRIER Explicit barrier.

OTF2_REGION_ROLE_IMPLICIT_BARRIER Implicit barrier.

OTF2_REGION_ROLE_FLUSH E.g. OpenMP "flush" construct.

OTF2_REGION_ROLE_ORDERED E.g. OpenMP "ordered" construct.

OTF2_REGION_ROLE_ORDERED_SBLOCK E.g. OpenMP "ordered" construct (structured block)

OTF2_REGION_ROLE_TASK "task" construct (structured block)

OTF2_REGION_ROLE_TASK_CREATE "task" construct (creation)

OTF2_REGION_ROLE_TASK_WAIT "taskwait" construct

OTF2_REGION_ROLE_COLL_ONE2ALL Collective 1:N communication operation.

OTF2_REGION_ROLE_COLL_ALL2ONE Collective N:1 communication operation.

OTF2_REGION_ROLE_COLL_ALL2ALL Collective N:N communication operation.

OTF2_REGION_ROLE_COLL_OTHER Collective M:N communication operation.

OTF2_REGION_ROLE_FILE_IO Any file I/O operation.

OTF2_REGION_ROLE_POINT2POINT A point-to-point communication function.

OTF2_REGION_ROLE_RMA A remote memory access communication operation.

OTF2_REGION_ROLE_DATA_TRANSFER A data transfer operation in memory.

OTF2_REGION_ROLE_ARTIFICIAL An artificial region, mostly used by the monitor software.

Since

Version 1.2.

OTF2_REGION_ROLE_THREAD_CREATE A function which creates one thread.

Since

Version 1.3.

OTF2_REGION_ROLE_THREAD_WAIT A function which waits for the completion of one thread.

Since

Version 1.3.

OTF2_REGION_ROLE_TASK_UNTIED "untied task" construct (structured block)

Since

Version 1.5.

E.7.2.17 enum OTF2_SystemTreeDomain_enum

List of available system tree node domains.

Since

Version 1.2

Enumerator:

OTF2_SYSTEM_TREE_DOMAIN_MACHINE All nodes below a node with this attribute encompass a tightly coupled HPC system.

OTF2_SYSTEM_TREE_DOMAIN_SHARED_MEMORY All nodes below a node with this attribute encompass a system where processes can communicate via hardware shared memory.

OTF2_SYSTEM_TREE_DOMAIN_NUMA A numa domain. A set of processors around memory which the processors can directly access.

OTF2_SYSTEM_TREE_DOMAIN_SOCKET Socket, physical package, or chip. In the physical meaning, i.e. that you can add or remove physically.

OTF2_SYSTEM_TREE_DOMAIN_CACHE Cache. Can be L1i, L1d, L2, L3, ...

OTF2_SYSTEM_TREE_DOMAIN_CORE Core. A computation unit (may be shared by several logical processors).

OTF2_SYSTEM_TREE_DOMAIN_PU Processing Unit (An non-shared ALU, FPU, ...)

E.8 otf2/OTF2_DefReader.h File Reference

This is the local definition reader, which reads location dependend definitions, and can also be used to get the mapping information from the local definition file. Local definitions are always assigned to a location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_DefReaderCallbacks.h>
```

Functions

- OTF2_ErrorCode OTF2_DefReader_GetLocationID (const OTF2_DefReader ∗reader, OTF2_LocationRef ∗location)

  Get the location ID of this reader object.
- OTF2_ErrorCode OTF2_DefReader_ReadDefinitions (OTF2_DefReader ∗reader, uint64_t recordsToRead, uint64_t ∗recordsRead)

Reads the given number of records from the definition reader.

- OTF2_ErrorCode OTF2_DefReader_SetCallbacks (OTF2_DefReader ∗reader, const OTF2_DefReaderCallbacks ∗callbacks, void ∗userData)

  Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

E.8.1 Detailed Description

This is the local definition reader, which reads location dependend definitions, and can also be used to get the mapping information from the local definition file. Local definitions are always assigned to a location.

E.8.2 Function Documentation

E.8.2.1 OTF2_ErrorCode OTF2_DefReader_GetLocationID (const OTF2_DefReader ∗ *reader,* OTF2_LocationRef ∗ *location*)

Get the location ID of this reader object.

Parameters

reader	This given reader object will be deleted.
location	Pointer to the variable where the location ID is returned in.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.8.2.2 OTF2_ErrorCode OTF2_DefReader_ReadDefinitions (OTF2_DefReader ∗ *reader,* uint64_t *recordsToRead,* uint64_t ∗ *recordsRead*)

Reads the given number of records from the definition reader.

Parameters

	reader	The records of this reader will be read when the function is issued.
	record-sToRead	This variable tells the reader how much records it has to read.

out	*record-sRead*	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given record-sToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finnished his job by checking recordsRead < recordsToRead.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INTERRUPTED_BY_CALLBACK if an user supplied callback returned OTF2_CALLBACK_INTERRUPT

OTF2_ERROR_DUPLICATE_MAPPING_TABLE if an duplicate mapping table definition was read

otherwise the error code

E.8.2.3 OTF2_ErrorCode OTF2_DefReader_SetCallbacks (OTF2_DefReader ∗ *reader*, const OTF2_DefReaderCallbacks ∗ *callbacks*, void ∗ *userData*)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

reader	This given reader object will be setted up with new callback functions.
callbacks	Struct which holds a function pointer for each record type. *OTF2_-DefReaderCallbacks_New*.
userData	Data passed as argument *userData* to the record callbacks.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.9 otf2/OTF2_DefReaderCallbacks.h File Reference

This defines the callbacks for the definition reader.

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

```
#include <otf2/OTF2_ErrorCodes.h>
```

E.9 otf2/OTF2_DefReaderCallbacks.h File Reference

```
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_IdMap.h>
```

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Attribute)(void ∗userData, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_Type type)

  Function pointer definition for the callback which is triggered by a Attribute definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_CallingContext)(void ∗userData, OTF2_CallingContextRef self, uint64_t ip, OTF2_RegionRef region, uint32_t offsetLineNumber, OTF2_CallingContextRef parent)

  Function pointer definition for the callback which is triggered by a CallingContext definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Callpath)(void ∗userData, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region)

  Function pointer definition for the callback which is triggered by a Callpath definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Callsite)(void ∗userData, OTF2_CallsiteRef self, OTF2_StringRef sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion, OTF2_RegionRef leftRegion)

  Function pointer definition for the callback which is triggered by a Callsite definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_CartCoordinate)(void ∗userData, OTF2_CartTopologyRef cartTopology, uint32_t rank, uint8_t numberOfDimensions, const uint32_t ∗coordinates)

  Function pointer definition for the callback which is triggered by a CartCoordinate definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_CartDimension)(void ∗userData, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_t size, OTF2_CartPeriodicity cartPeriodicity)

  Function pointer definition for the callback which is triggered by a CartDimension definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_CartTopology)(void ∗userData, OTF2_CartTopologyRef self, OTF2_StringRef name, OTF2_CommRef communicator, uint8_t numberOfDimensions, const OTF2_CartDimensionRef ∗cartDimensions)

  Function pointer definition for the callback which is triggered by a CartTopology definition record.

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_ClockOffset)(void ∗userData, OTF2_TimeStamp time, int64_t offset, double standardDeviation)

  Function pointer definition for the callback which is triggered by a ClockOffset definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Comm)(void ∗userData, OTF2_CommRef self, OTF2_StringRef name, OTF2_GroupRef group, OTF2_CommRef parent)

  Function pointer definition for the callback which is triggered by a Comm definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Group)(void ∗userData, OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_t numberOfMembers, const uint64_t ∗members)

  Function pointer definition for the callback which is triggered by a Group definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_InterruptGenerator)(void ∗userData, OTF2_InterruptGeneratorRef self, OTF2_StringRef name, OTF2_StringRef unit, uint64_t period)

  Function pointer definition for the callback which is triggered by a InterruptGenerator definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Location)(void ∗userData, OTF2_LocationRef self, OTF2_StringRef name, OTF2_LocationType locationType, uint64_t numberOfEvents, OTF2_LocationGroupRef locationGroup)

  Function pointer definition for the callback which is triggered by a Location definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_LocationGroup)(void ∗userData, OTF2_LocationGroupRef self, OTF2_StringRef name, OTF2_LocationGroupType locationGroupType, OTF2_SystemTreeNodeRef systemTreeParent)

  Function pointer definition for the callback which is triggered by a LocationGroup definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_LocationGroupProperty)(void ∗userData, OTF2_LocationGroupRef locationGroup, OTF2_StringRef name, OTF2_StringRef value)

  Function pointer definition for the callback which is triggered by a LocationGroupProperty definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_LocationProperty)(void ∗userData, OTF2_LocationRef location, OTF2_StringRef name, OTF2_StringRef value)

  Function pointer definition for the callback which is triggered by a LocationProperty definition record.

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_MappingTable)(void ∗userData, OTF2_MappingType mappingType, const OTF2_IdMap ∗idMap)

  Function pointer definition for the callback which is triggered by a MappingTable definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_MetricClass)(void ∗userData, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_-MetricMemberRef ∗metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)

  Function pointer definition for the callback which is triggered by a MetricClass definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_MetricClassRecorder)(void ∗userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)

  Function pointer definition for the callback which is triggered by a MetricClass-Recorder definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_MetricInstance)(void ∗userData, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_-LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)

  Function pointer definition for the callback which is triggered by a MetricInstance definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_MetricMember)(void ∗userData, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_-StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_-t exponent, OTF2_StringRef unit)

  Function pointer definition for the callback which is triggered by a MetricMember definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Parameter)(void ∗userData, OTF2_ParameterRef self, OTF2_StringRef name, OTF2_ParameterType parameterType)

  Function pointer definition for the callback which is triggered by a Parameter definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Region)(void ∗userData, OTF2_RegionRef self, OTF2_StringRef name, OTF2_StringRef canonicalName, OTF2_StringRef description, OTF2_RegionRole regionRole, OTF2_-Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)

  Function pointer definition for the callback which is triggered by a Region definition record.
- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_RmaWin)(void ∗userData, OTF2_RmaWinRef self, OTF2_StringRef name, OTF2_CommRef comm)

Function pointer definition for the callback which is triggered by a RmaWin definition record.

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_SourceCodeLocation)(void ∗userData, OTF2_SourceCodeLocationRef self, OTF2_StringRef file, uint32_t lineNumber)

  Function pointer definition for the callback which is triggered by a SourceCode-Location definition record.

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_String)(void ∗userData, OTF2_StringRef self, const char ∗string)

  Function pointer definition for the callback which is triggered by a String definition record.

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_SystemTreeNode)(void ∗userData, OTF2_SystemTreeNodeRef self, OTF2_StringRef name, OTF2_StringRef className, OTF2_SystemTreeNodeRef parent)

  Function pointer definition for the callback which is triggered by a SystemTreeN-ode definition record.

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_SystemTreeNodeDomain)(void ∗userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_SystemTreeDomain systemTreeDomain)

  Function pointer definition for the callback which is triggered by a SystemTreeN-odeDomain definition record.

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_SystemTreeNodeProperty)(void ∗userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_StringRef name, OTF2_StringRef value)

  Function pointer definition for the callback which is triggered by a SystemTreeN-odeProperty definition record.

- typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Unknown)(void ∗userData)

  Function pointer definition for the callback which is triggered for an unknown definition.

- typedef struct OTF2_DefReaderCallbacks_struct OTF2_DefReaderCallbacks

  Opaque struct which holdes all definition record callbacks.

Functions

- void OTF2_DefReaderCallbacks_Clear (OTF2_DefReaderCallbacks ∗defReaderCallbacks)

  Clears a struct for the definition callbacks.

- void OTF2_DefReaderCallbacks_Delete (OTF2_DefReaderCallbacks ∗defReaderCallbacks)

Deallocates a struct for the definition callbacks.

- OTF2_DefReaderCallbacks ∗ OTF2_DefReaderCallbacks_New (void)

  Allocates a new struct for the definition callbacks.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetAttributeCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Attribute attributeCallback)

  Registers the callback for the Attribute definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallingContextCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_CallingContext callingContextCallback)

  Registers the callback for the CallingContext definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallpathCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Callpath callpathCallback)

  Registers the callback for the Callpath definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallsiteCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Callsite callsiteCallback)

  Registers the callback for the Callsite definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCartCoordinateCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_CartCoordinate cartCoordinateCallback)

  Registers the callback for the CartCoordinate definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCartDimensionCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_CartDimension cartDimensionCallback)

  Registers the callback for the CartDimension definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCartTopologyCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_CartTopology cartTopologyCallback)

  Registers the callback for the CartTopology definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetClockOffsetCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_ClockOffset clockOffsetCallback)

  Registers the callback for the ClockOffset definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCommCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Comm commCallback)

  Registers the callback for the Comm definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetGroupCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Group groupCallback)

  Registers the callback for the Group definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetInterruptGeneratorCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_-InterruptGenerator interruptGeneratorCallback)

  Registers the callback for the InterruptGenerator definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Location locationCallback)

  Registers the callback for the Location definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationGroupCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_LocationGroup locationGroupCallback)

  Registers the callback for the LocationGroup definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationGroupPropertyCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_-LocationGroupProperty locationGroupPropertyCallback)

  Registers the callback for the LocationGroupProperty definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationPropertyCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_-LocationProperty locationPropertyCallback)

  Registers the callback for the LocationProperty definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMappingTableCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_MappingTable mappingTableCallback)

  Registers the callback for the MappingTable definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricClassCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_MetricClass metricClassCallback)

  Registers the callback for the MetricClass definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricClassRecorderCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_-MetricClassRecorder metricClassRecorderCallback)

  Registers the callback for the MetricClassRecorder definition.
- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricInstanceCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_MetricInstance metricInstanceCallback)

  Registers the callback for the MetricInstance definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricMemberCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_MetricMember metricMemberCallback)

  Registers the callback for the MetricMember definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetParameterCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Parameter parameterCallback)

  Registers the callback for the Parameter definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetRegionCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Region regionCallback)

  Registers the callback for the Region definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetRmaWinCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_RmaWin rmaWinCallback)

  Registers the callback for the RmaWin definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSourceCodeLocationCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_-SourceCodeLocation sourceCodeLocationCallback)

  Registers the callback for the SourceCodeLocation definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetStringCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_String stringCallback)

  Registers the callback for the String definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodeCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_-SystemTreeNode systemTreeNodeCallback)

  Registers the callback for the SystemTreeNode definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodeDomainCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_-SystemTreeNodeDomain systemTreeNodeDomainCallback)

  Registers the callback for the SystemTreeNodeDomain definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodePropertyCallback (OTF2_DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_-SystemTreeNodeProperty systemTreeNodePropertyCallback)

  Registers the callback for the SystemTreeNodeProperty definition.

- OTF2_ErrorCode OTF2_DefReaderCallbacks_SetUnknownCallback (OTF2_-DefReaderCallbacks ∗defReaderCallbacks, OTF2_DefReaderCallback_Unknown unknownCallback)

  Registers the callback for an unknown definition.

E.9.1 Detailed Description

This defines the callbacks for the definition reader.

Source Template:

templates/OTF2_DefReaderCallbacks.tmpl.h

E.9.2 Typedef Documentation

E.9.2.1 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_Attribute)(void ∗userData, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_Type type)

Function pointer definition for the callback which is triggered by a *Attribute* definition record.

The attribute definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_DefReader_SetCallbacks*.
self	The unique identifier for this *Attribute* definition.
name	Name of the attribute. References a *String* definition.
description	Description of the attribute. References a *String* definition. Since version 1.4.
type	Type of the attribute value.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.2 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_CallingContext)(void ∗userData, OTF2_CallingContextRef self, uint64_t ip, OTF2_RegionRef region, uint32_t offsetLineNumber, OTF2_CallingContextRef parent)

Function pointer definition for the callback which is triggered by a *CallingContext* definition record.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *CallingContext* definition.
ip	Instruction pointer as the offset to the start of the function.
region	The region. References a *Region* definition.
offsetLi-neNumber	The line offset inside the region.
parent	Parent id of this context. References a *CallingContext* definition.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.3 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-Callpath)(void ∗userData, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region)

Function pointer definition for the callback which is triggered by a *Callpath* definition record.

The callpath definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *Callpath* definition.
parent	The parent of this callpath. References a *Callpath* definition.
region	The region of this callpath. References a *Region* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.4 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-Callsite)(void ∗userData, OTF2_CallsiteRef self, OTF2_StringRef sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion, OTF2_RegionRef leftRegion)

Function pointer definition for the callback which is triggered by a *Callsite* definition record.

The callsite definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *Callsite* definition.
sourceFile	The source file where this call was made. References a *String* definition.
lineNumber	Line number in the source file where this call was made.
enteredRe-gion	The region which was called. References a *Region* definition.
leftRegion	The region which made the call. References a *Region* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.5 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-CartCoordinate)(void ∗userData, OTF2_CartTopologyRef cartTopology, uint32_t rank, uint8_t numberOfDimensions, const uint32_t ∗coordinates)

Function pointer definition for the callback which is triggered by a *CartCoordinate* definition record.

Defines the coordinate of the location referenced by the given rank (w.r.t. the communicator associated to the topology) in the referenced topology.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
cartTopol-ogy	Parent *CartTopology* definition to which this one is a supplementary definition. References a *CartTopology* definition.

rank	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
numberOfDimensions	Number of dimensions.
coordinates	Coordinates, indexed by dimension.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.6 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_CartDimension)(void ∗userData, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_t size, OTF2_CartPeriodicity cartPeriodicity)

Function pointer definition for the callback which is triggered by a *CartDimension* definition record.

Each dimension in a Cartesian topology is composed of a global id, a name, its size, and whether it is periodic or not.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_DefReader_SetCallbacks*.
self	The unique identifier for this *CartDimension* definition.
name	The name of the cartesian topology dimension. References a *String* definition.
size	The size of the cartesian topology dimension.
cartPeriodicity	Periodicity of the cartesian topology dimension.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.7 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-CartTopology)(void ∗userData, OTF2_CartTopologyRef self, OTF2_StringRef name, OTF2_CommRef communicator, uint8_t numberOfDimensions, const OTF2_CartDimensionRef ∗cartDimensions)

Function pointer definition for the callback which is triggered by a *CartTopology* definition record.

Each topology is described by a global id, a reference to its name, a reference to a communicator, the number of dimensions, and references to those dimensions. The topology type is defined by the paradigm of the group referenced by the associated communicator.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *CartTopology* definition.
name	The name of the topology. References a *String* definition.
communi-cator	Communicator object used to create the topology. References a *Comm* definition.
num-berOfDi-mensions	Number of dimensions.
cartDimen-sions	The dimensions of this topology. References a *CartDimension* definition.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.8 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-ClockOffset)(void ∗userData, OTF2_TimeStamp time, int64_t offset, double standardDeviation)

Function pointer definition for the callback which is triggered by a *ClockOffset* definition record.

Clock offsets are used for clock corrections.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
time	Time when this offset was determined.
offset	The offset to the global clock which was determined at `time`.
standard-Deviation	A possible standard deviation, which can be used as a metric for the quality of the offset.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.9 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-Comm)(void ∗userData, OTF2_CommRef self, OTF2_StringRef name, OTF2_GroupRef group, OTF2_CommRef parent)

Function pointer definition for the callback which is triggered by a *Comm* definition record.

The communicator definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *Comm* definition.
name	The name given by calling MPI_Comm_set_name on this communicator. Or the empty name to indicate that no name was given. References a *String* definition.
group	The describing MPI group of this MPI communicator The group needs to be of type *OTF2_GROUP_TYPE_COMM_GROUP* or *OTF2_-GROUP_TYPE_COMM_SELF*. References a *Group* definition.
parent	The parent MPI communicator from which this communicator was created, if any. Use *OTF2_UNDEFINED_COMM* to indicate no parent. References a *Comm* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.10 typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_-Group)(void *userData, OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_t numberOfMembers, const uint64_t *members)

Function pointer definition for the callback which is triggered by a *Group* definition record.

The group definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *Group* definition.
name	Name of this group References a *String* definition.
groupType	The type of this group. Since version 1.2.
paradigm	The paradigm of this communication group. Since version 1.2.
groupFlags	Flags for this group. Since version 1.2.
num-berOfMem-bers	The number of members in this group.
members	The identifiers of the group members.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.11 typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_-InterruptGenerator)(void *userData, OTF2_InterruptGeneratorRef self, OTF2_StringRef name, OTF2_StringRef unit, uint64_t period)

Function pointer definition for the callback which is triggered by a *InterruptGenerator* definition record.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *InterruptGenerator* definition.
name	The name of this interrupt generator. References a *String* definition.
unit	The unit used by this interrupt generator for the period. References a *String* definition.
period	The period this interrupt generator generates interrupts.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.12 typedef OTF2_CallbackCode($*$ OTF2_DefReaderCallback_-Location)(void $*$userData, OTF2_LocationRef self, OTF2_StringRef name, OTF2_LocationType locationType, uint64_t numberOfEvents, OTF2_LocationGroupRef locationGroup)

Function pointer definition for the callback which is triggered by a *Location* definition record.

The location definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *Location* definition.
name	Name of the location References a *String* definition.
location-Type	Location type.
numberO-fEvents	Number of events this location has recorded.
location-Group	Location group which includes this location. References a *Location-Group* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.13 typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_-LocationGroup)(void *userData, OTF2_LocationGroupRef self, OTF2_StringRef name, OTF2_LocationGroupType locationGroupType, OTF2_SystemTreeNodeRef systemTreeParent)

Function pointer definition for the callback which is triggered by a *LocationGroup* definition record.

The location group definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *LocationGroup* definition.
name	Name of the group. References a *String* definition.
location-GroupType	Type of this group.
sys-temTreePar-ent	Parent of this location group in the system tree. References a *Sys-temTreeNode* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.14 typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_-LocationGroupProperty)(void *userData, OTF2_LocationGroupRef locationGroup, OTF2_StringRef name, OTF2_StringRef value)

Function pointer definition for the callback which is triggered by a *LocationGroup-Property* definition record.

An arbitrary key/value property for a *LocationGroup* definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
location-Group	Parent *LocationGroup* definition to which this one is a supplementary definition. References a *LocationGroup* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.15 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-LocationProperty)(void ∗userData, OTF2_LocationRef location, OTF2_StringRef name, OTF2_StringRef value)

Function pointer definition for the callback which is triggered by a *LocationProperty* definition record.

An arbitrary key/value property for a *Location* definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
location	Parent *Location* definition to which this one is a supplementary definition. References a *Location* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.16 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_MappingTable)(void ∗userData, OTF2_MappingType mappingType, const OTF2_IdMap ∗idMap)

Function pointer definition for the callback which is triggered by a *MappingTable* definition record.

Mapping tables are needed for situations where an ID is not globally known at measurement time. They are applied automatically at reading.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_DefReader_SetCallbacks*.
mappingType	Says to what type of ID the mapping table has to be applied.
idMap	Mapping table.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.17 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_MetricClass)(void ∗userData, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_MetricMemberRef ∗metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)

Function pointer definition for the callback which is triggered by a *MetricClass* definition record.

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_DefReader_SetCallbacks*.
self	The unique identifier for this *MetricClass* definition.
numberOfMetrics	Number of metrics within the set.

metricMembers	List of metric members. References a *MetricMember* definition.
metricOccurrence	Defines occurrence of a metric set.
recorderKind	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.18 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-MetricClassRecorder)(void ∗userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)

Function pointer definition for the callback which is triggered by a *MetricClass-Recorder* definition record.

The metric class recorder definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
metricClass	Parent *MetricClass* definition to which this one is a supplementary definition. References a *MetricClass* definition.
recorder	The location which recorded the referenced metric class. References a *Location* definition.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.19 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-MetricInstance)(void ∗userData, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)

Function pointer definition for the callback which is triggered by a *MetricInstance* definition record.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type *OTF2_METRIC_ASYNCHRONOUS*.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *MetricClass* definition.
metricClass	The instanced *MetricClass*. This metric class must be of kind *OTF2_-RECORDER_KIND_ABSTRACT*. References a *MetricClass* definition.
recorder	Recorder of the metric: location ID. References a *Location* definition.
metric-Scope	Defines type of scope: location, location group, system tree node, or a generic group of locations.
scope	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.20 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-MetricMember)(void ∗userData, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2_StringRef unit)

Function pointer definition for the callback which is triggered by a *MetricMember* definition record.

A metric is defined by a metric member definition. A metric member is always a member of a metric class. Therefore, a single metric is a special case of a metric class with only one member. It is not allowed to reference a metric member id in a metric event, but only metric class IDs.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *MetricMember* definition.
name	Name of the metric. References a *String* definition.
description	Description of the metric. References a *String* definition.
metricType	Metric type: PAPI, etc.
metricMode	Metric mode: accumulative, fix, relative, etc.
valueType	Type of the value. Only *OTF2_TYPE_INT64*, *OTF2_TYPE_UINT64*, and *OTF2_TYPE_DOUBLE* are valid types. If this metric member is recorded in an *Metric* event, than this type and the type in the event must match.
metricBase	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
exponent	The values inside the Metric events should be scaled by the factor base$^\wedge$exponent, to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be *OTF2_BASE_-BINARY* and the exponent 10. Than the writer does not need to scale the values up to bytes, but can directly write the KiBi values into the Metric event. At reading time, the reader can apply the scaling factor to get the value in its base unit, ie. in bytes.
unit	Unit of the metric. This needs to be the scale free base unit, ie. "bytes", "operations", or "seconds". In particular this unit should not have any scale prefix. References a *String* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.21 typedef OTF2_CallbackCode($*$ OTF2_DefReaderCallback_-Parameter)(void $*$userData, OTF2_ParameterRef self, OTF2_StringRef name, OTF2_ParameterType parameterType)

Function pointer definition for the callback which is triggered by a *Parameter* definition record.

The parameter definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *Parameter* definition.
name	Name of the parameter (variable name etc.) References a *String* definition.
parameter-Type	Type of the parameter, *OTF2_ParameterType* for possible types.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.22 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-Region)(void ∗userData, OTF2_RegionRef self, OTF2_StringRef name, OTF2_StringRef canonicalName, OTF2_StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)

Function pointer definition for the callback which is triggered by a *Region* definition record.

The region definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *Region* definition.
name	Name of the region (demangled name if available). References a *String* definition.
canonical-Name	Alternative name of the region (e.g. mangled name). References a *String* definition. Since version 1.1.
description	A more detailed description of this region. References a *String* definition.
regionRole	Region role. Since version 1.1.
paradigm	Paradigm. Since version 1.1.
regionFlags	Region flags. Since version 1.1.

sourceFile	The source file where this region was declared. References a *String* definition.
beginLineNumber	Starting line number of this region in the source file.
endLineNumber	Ending line number of this region in the source file.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.23 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_RmaWin)(void ∗userData, OTF2_RmaWinRef self, OTF2_StringRef name, OTF2_CommRef comm)

Function pointer definition for the callback which is triggered by a *RmaWin* definition record.

A window defines the communication context for any remote-memory access operation.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_DefReader_SetCallbacks*.
self	The unique identifier for this *RmaWin* definition.
name	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a *String* definition.
comm	Communicator object used to create the window. References a *Comm* definition.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.24 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-SourceCodeLocation)(void ∗userData, OTF2_SourceCodeLocationRef self, OTF2_StringRef file, uint32_t lineNumber)

Function pointer definition for the callback which is triggered by a *SourceCodeLocation* definition record.

The definition of a source code location as tuple of the corresponding file name and line number.

When used to attach source code annotations to events, use the *OTF2_AttributeList* with a *Attribute* definition named `"SOURCE_CODE_LOCATION"` and typed *OTF2_-TYPE_SOURCE_CODE_LOCATION*.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *SourceCodeLocation* definition.
file	The name of the file for the source code location. References a *String* definition.
lineNumber	The line number for the source code location.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.25 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-String)(void ∗userData, OTF2_StringRef self, const char ∗string)

Function pointer definition for the callback which is triggered by a *String* definition record.

The string definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *String* definition.
string	The string, null terminated.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.26 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-SystemTreeNode)(void ∗userData, OTF2_SystemTreeNodeRef self, OTF2_StringRef name, OTF2_StringRef className, OTF2_SystemTreeNodeRef parent)

Function pointer definition for the callback which is triggered by a *SystemTreeNode* definition record.

The system tree node definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
self	The unique identifier for this *SystemTreeNode* definition.
name	Free form instance name of this node. References a *String* definition.
className	Free form class name of this node References a *String* definition.
parent	Parent id of this node. May be *OTF2_UNDEFINED_SYSTEM_TREE_-NODE* to indicate that there is no parent. References a *SystemTreeNode* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.27 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-SystemTreeNodeDomain)(void ∗userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_SystemTreeDomain systemTreeDomain)

Function pointer definition for the callback which is triggered by a *SystemTreeN-odeDomain* definition record.

The system tree node domain definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
sys-temTreeN-ode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
systemTree-Domain	The domain in which the referenced *SystemTreeNode* operates in.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.28 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-SystemTreeNodeProperty)(void ∗userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_StringRef name, OTF2_StringRef value)

Function pointer definition for the callback which is triggered by a *SystemTreeN-odeProperty* definition record.

An arbitrary key/value property for a *SystemTreeNode* definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.
sys-temTreeN-ode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.2.29 typedef OTF2_CallbackCode(∗ OTF2_DefReaderCallback_-Unknown)(void ∗userData)

Function pointer definition for the callback which is triggered for an unknown definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterDefCallbacks* or *OTF2_-DefReader_SetCallbacks*.

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.9.3 Function Documentation

E.9.3.1 void OTF2_DefReaderCallbacks_Clear (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*)

Clears a struct for the definition callbacks.

Parameters

defReader-Callbacks	Handle to a struct previously allocated with *OTF2_-DefReaderCallbacks_New*.

E.9.3.2 void OTF2_DefReaderCallbacks_Delete (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*)

Deallocates a struct for the definition callbacks.

Parameters

defReader-Callbacks	Handle to a struct previously allocated with *OTF2_-DefReaderCallbacks_New*.

E.9.3.3 OTF2_DefReaderCallbacks∗ OTF2_DefReaderCallbacks_New (void)

Allocates a new struct for the definition callbacks.

Returns

A newly allocated struct of type *OTF2_DefReaderCallbacks*.

E.9.3.4 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetAttributeCallback (OTF2_DefReaderCallbacks $*$ *defReaderCallbacks*, OTF2_DefReaderCallback_Attribute *attributeCallback*)

Registers the callback for the *Attribute* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
attributeCallback	Function which should be called for all *Attribute* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.5 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallingContextCallback (OTF2_DefReaderCallbacks $*$ *defReaderCallbacks*, OTF2_DefReaderCallback_CallingContext *callingContextCallback*)

Registers the callback for the *CallingContext* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
callingContextCallback	Function which should be called for all *CallingContext* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

E.9.3.6 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallpathCallback (OTF2_DefReaderCallbacks * *defReaderCallbacks*, OTF2_DefReaderCallback_Callpath *callpathCallback*)

Registers the callback for the *Callpath* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.
callpath-Callback	Function which should be called for all *Callpath* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.7 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallsiteCallback (OTF2_DefReaderCallbacks * *defReaderCallbacks*, OTF2_DefReaderCallback_Callsite *callsiteCallback*)

Registers the callback for the *Callsite* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.
callsite-Callback	Function which should be called for all *Callsite* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.8 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCartCoordinateCallback (OTF2_DefReaderCallbacks * *defReaderCallbacks*, OTF2_DefReaderCallback_CartCoordinate *cartCoordinateCallback*)

Registers the callback for the *CartCoordinate* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
cartCoordinateCallback	Function which should be called for all *CartCoordinate* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.9 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCartDimensionCallback (OTF2_DefReaderCallbacks * *defReaderCallbacks*, OTF2_DefReaderCallback_CartDimension *cartDimensionCallback*)

Registers the callback for the *CartDimension* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
cartDimensionCallback	Function which should be called for all *CartDimension* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.10 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCartTopologyCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_CartTopology *cartTopologyCallback*)

Registers the callback for the *CartTopology* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
cartTopologyCallback	Function which should be called for all *CartTopology* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.11 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetClockOffsetCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_ClockOffset *clockOffsetCallback*)

Registers the callback for the *ClockOffset* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
clockOffsetCallback	Function which should be called for all *ClockOffset* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.12 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCommCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_Comm *commCallback*)

Registers the callback for the *Comm* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
commCallback	Function which should be called for all *Comm* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.13 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetGroupCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_Group *groupCallback*)

Registers the callback for the *Group* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
groupCallback	Function which should be called for all *Group* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.14 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetInterruptGeneratorCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_InterruptGenerator *interruptGeneratorCallback*)

Registers the callback for the *InterruptGenerator* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
interruptGeneratorCallback	Function which should be called for all *InterruptGenerator* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.15 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_Location *locationCallback*)

Registers the callback for the *Location* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.
location-Callback	Function which should be called for all *Location* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.16 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationGroupCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_LocationGroup *locationGroupCallback*)

Registers the callback for the *LocationGroup* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.
location-GroupCall-back	Function which should be called for all *LocationGroup* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.17 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationGroupPropertyCallback (OTF2_DefReaderCallbacks * *defReaderCallbacks*, OTF2_DefReaderCallback_LocationGroupProperty *locationGroupPropertyCallback*)

Registers the callback for the *LocationGroupProperty* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
locationGroupPropertyCallback	Function which should be called for all *LocationGroupProperty* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.18 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationPropertyCallback (OTF2_DefReaderCallbacks * *defReaderCallbacks*, OTF2_DefReaderCallback_LocationProperty *locationPropertyCallback*)

Registers the callback for the *LocationProperty* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
locationPropertyCallback	Function which should be called for all *LocationProperty* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.19 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMappingTableCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_MappingTable *mappingTableCallback*)

Registers the callback for the *MappingTable* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.
map-pingTable-Callback	Function which should be called for all *MappingTable* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.20 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricClassCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_MetricClass *metricClassCallback*)

Registers the callback for the *MetricClass* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.

metricClassCallback	Function which should be called for all *MetricClass* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.21 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricClassRecorderCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_MetricClassRecorder *metricClassRecorderCallback*)

Registers the callback for the *MetricClassRecorder* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
metricClassRecorderCallback	Function which should be called for all *MetricClassRecorder* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.22 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricInstanceCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_MetricInstance *metricInstanceCallback*)

Registers the callback for the *MetricInstance* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
metricInstanceCallback	Function which should be called for all *MetricInstance* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.23 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricMemberCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_MetricMember *metricMemberCallback*)

Registers the callback for the *MetricMember* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
metricMemberCallback	Function which should be called for all *MetricMember* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

E.9.3.24 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetParameterCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_Parameter *parameterCallback*)

Registers the callback for the *Parameter* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.
parameter-Callback	Function which should be called for all *Parameter* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.25 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetRegionCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_Region *regionCallback*)

Registers the callback for the *Region* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.
regionCall-back	Function which should be called for all *Region* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.26 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetRmaWinCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_RmaWin *rmaWinCallback*)

Registers the callback for the *RmaWin* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
rmaWinCallback	Function which should be called for all *RmaWin* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.27 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSourceCodeLocationCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_SourceCodeLocation *sourceCodeLocationCallback*)

Registers the callback for the *SourceCodeLocation* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
sourceCodeLocationCallback	Function which should be called for all *SourceCodeLocation* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.28 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetStringCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_String *stringCallback*)

Registers the callback for the *String* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.
stringCall-back	Function which should be called for all *String* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.29 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodeCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_SystemTreeNode *systemTreeNodeCallback*)

Registers the callback for the *SystemTreeNode* definition.

Parameters

defReader-Callbacks	Struct for all callbacks.

systemTreeNodeCallback	Function which should be called for all *SystemTreeNode* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.30 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodeDomainCallback (OTF2_DefReaderCallbacks ∗ *defReaderCallbacks*, OTF2_DefReaderCallback_SystemTreeNodeDomain *systemTreeNodeDomainCallback*)

Registers the callback for the *SystemTreeNodeDomain* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
systemTreeNodeDomainCallback	Function which should be called for all *SystemTreeNodeDomain* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.31 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodePropertyCallback (OTF2_DefReaderCallbacks * *defReaderCallbacks*, OTF2_DefReaderCallback_SystemTreeNodeProperty *systemTreeNodePropertyCallback*)

Registers the callback for the *SystemTreeNodeProperty* definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
systemTreeNodePropertyCallback	Function which should be called for all *SystemTreeNodeProperty* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9.3.32 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetUnknownCallback (OTF2_DefReaderCallbacks * *defReaderCallbacks*, OTF2_DefReaderCallback_Unknown *unknownCallback*)

Registers the callback for an unknown definition.

Parameters

defReaderCallbacks	Struct for all callbacks.
unknownCallback	Function which should be called for all unknown definitions.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

E.10 otf2/OTF2_DefWriter.h File Reference

This file provides all routines that write definition records of a single location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_IdMap.h>
```

Typedefs

- typedef struct OTF2_DefWriter_struct OTF2_DefWriter

  Handle definition for the external definition writer.

Functions

- OTF2_ErrorCode OTF2_DefWriter_GetLocationID (const OTF2_DefWriter ∗writer, OTF2_LocationRef ∗location)

  Returns the location ID of the location which is related to the writer object.
- OTF2_ErrorCode OTF2_DefWriter_WriteAttribute (OTF2_DefWriter ∗writer, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_Type type)

  Writes a Attribute definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteCallingContext (OTF2_DefWriter ∗writer, OTF2_CallingContextRef self, uint64_t ip, OTF2_RegionRef region, uint32_t offsetLineNumber, OTF2_CallingContextRef parent)

  Writes a CallingContext definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteCallpath (OTF2_DefWriter ∗writer, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region)

  Writes a Callpath definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteCallsite (OTF2_DefWriter ∗writer, OTF2_CallsiteRef self, OTF2_StringRef sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion, OTF2_RegionRef leftRegion)

  Writes a Callsite definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteCartCoordinate (OTF2_DefWriter ∗writer, OTF2_CartTopologyRef cartTopology, uint32_t rank, uint8_t numberOfDimensions, const uint32_t ∗coordinates)

Writes a CartCoordinate definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteCartDimension (OTF2_DefWriter ∗writer, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_t size, OTF2_CartPeriodicity cartPeriodicity)

  Writes a CartDimension definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteCartTopology (OTF2_DefWriter ∗writer, OTF2_CartTopologyRef self, OTF2_StringRef name, OTF2_CommRef communicator, uint8_t numberOfDimensions, const OTF2_CartDimensionRef ∗cartDimensions)

  Writes a CartTopology definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteClockOffset (OTF2_DefWriter ∗writer, OTF2_TimeStamp time, int64_t offset, double standardDeviation)

  Writes a ClockOffset definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteComm (OTF2_DefWriter ∗writer, OTF2_CommRef self, OTF2_StringRef name, OTF2_GroupRef group, OTF2_-CommRef parent)

  Writes a Comm definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteGroup (OTF2_DefWriter ∗writer, OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_t numberOfMembers, const uint64_t ∗members)

  Writes a Group definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteInterruptGenerator (OTF2_DefWriter ∗writer, OTF2_InterruptGeneratorRef self, OTF2_StringRef name, OTF2_-StringRef unit, uint64_t period)

  Writes a InterruptGenerator definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteLocation (OTF2_DefWriter ∗writer, OTF2_LocationRef self, OTF2_StringRef name, OTF2_LocationType locationType, uint64_t numberOfEvents, OTF2_LocationGroupRef locationGroup)

  Writes a Location definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteLocationGroup (OTF2_DefWriter ∗writer, OTF2_LocationGroupRef self, OTF2_StringRef name, OTF2_LocationGroupType locationGroupType, OTF2_SystemTreeNodeRef systemTreeParent)

  Writes a LocationGroup definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteLocationGroupProperty (OTF2_-DefWriter ∗writer, OTF2_LocationGroupRef locationGroup, OTF2_StringRef name, OTF2_StringRef value)

  Writes a LocationGroupProperty definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteLocationProperty (OTF2_DefWriter ∗writer, OTF2_LocationRef location, OTF2_StringRef name, OTF2_StringRef value)

Writes a LocationProperty definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteMappingTable (OTF2_DefWriter ∗writer, OTF2_MappingType mappingType, const OTF2_IdMap ∗idMap)

  Writes a MappingTable definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteMetricClass (OTF2_DefWriter ∗writer, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_MetricMemberRef ∗metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)

  Writes a MetricClass definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteMetricClassRecorder (OTF2_DefWriter ∗writer, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)

  Writes a MetricClassRecorder definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteMetricInstance (OTF2_DefWriter ∗writer, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)

  Writes a MetricInstance definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteMetricMember (OTF2_DefWriter ∗writer, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2_StringRef unit)

  Writes a MetricMember definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteParameter (OTF2_DefWriter ∗writer, OTF2_ParameterRef self, OTF2_StringRef name, OTF2_ParameterType parameterType)

  Writes a Parameter definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteRegion (OTF2_DefWriter ∗writer, OTF2_RegionRef self, OTF2_StringRef name, OTF2_StringRef canonicalName, OTF2_StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)

  Writes a Region definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteRmaWin (OTF2_DefWriter ∗writer, OTF2_RmaWinRef self, OTF2_StringRef name, OTF2_CommRef comm)

  Writes a RmaWin definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteSourceCodeLocation (OTF2_DefWriter ∗writer, OTF2_SourceCodeLocationRef self, OTF2_StringRef file, uint32_t lineNumber)

  Writes a SourceCodeLocation definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteString (OTF2_DefWriter ∗writer, OTF2_StringRef self, const char ∗string)

Writes a String definition record into the DefWriter.

- OTF2_ErrorCode OTF2_DefWriter_WriteSystemTreeNode (OTF2_DefWriter ∗writer, OTF2_SystemTreeNodeRef self, OTF2_StringRef name, OTF2_-StringRef className, OTF2_SystemTreeNodeRef parent)

  Writes a SystemTreeNode definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteSystemTreeNodeDomain (OTF2_-DefWriter ∗writer, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_-SystemTreeDomain systemTreeDomain)

  Writes a SystemTreeNodeDomain definition record into the DefWriter.
- OTF2_ErrorCode OTF2_DefWriter_WriteSystemTreeNodeProperty (OTF2_-DefWriter ∗writer, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_-StringRef name, OTF2_StringRef value)

  Writes a SystemTreeNodeProperty definition record into the DefWriter.

E.10.1 Detailed Description

This file provides all routines that write definition records of a single location.

Source Template:

templates/OTF2_DefWriter.tmpl.h

E.10.2 Function Documentation

E.10.2.1 OTF2_ErrorCode OTF2_DefWriter_GetLocationID (const OTF2_DefWriter ∗ *writer*, OTF2_LocationRef ∗ *location*)

Returns the location ID of the location which is related to the writer object.

Parameters

writer	Writer object.
location	Return location reference.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.2 OTF2_ErrorCode OTF2_DefWriter_WriteAttribute (OTF2_DefWriter * *writer*, OTF2_AttributeRef *self*, OTF2_StringRef *name*, OTF2_StringRef *description*, OTF2_Type *type*)

Writes a Attribute definition record into the DefWriter.

The attribute definition.

Parameters

writer	Writer object.
self	The unique identifier for this *Attribute* definition.
name	Name of the attribute. References a *String* definition.
description	Description of the attribute. References a *String* definition. Since version 1.4.
type	Type of the attribute value.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.3 OTF2_ErrorCode OTF2_DefWriter_WriteCallingContext (OTF2_DefWriter * *writer*, OTF2_CallingContextRef *self*, uint64_t *ip*, OTF2_RegionRef *region*, uint32_t *offsetLineNumber*, OTF2_CallingContextRef *parent*)

Writes a CallingContext definition record into the DefWriter.

Parameters

writer	Writer object.
self	The unique identifier for this *CallingContext* definition.
ip	Instruction pointer as the offset to the start of the function.
region	The region. References a *Region* definition.
offsetLineNumber	The line offset inside the region.
parent	Parent id of this context. References a *CallingContext* definition.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.4 OTF2_ErrorCode OTF2_DefWriter_WriteCallpath (OTF2_DefWriter ∗ *writer,* OTF2_CallpathRef *self,* OTF2_CallpathRef *parent,* OTF2_RegionRef *region*)

Writes a Callpath definition record into the DefWriter.

The callpath definition.

Parameters

writer	Writer object.
self	The unique identifier for this *Callpath* definition.
parent	The parent of this callpath. References a *Callpath* definition.
region	The region of this callpath. References a *Region* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.5 OTF2_ErrorCode OTF2_DefWriter_WriteCallsite (OTF2_DefWriter ∗ *writer,* OTF2_CallsiteRef *self,* OTF2_StringRef *sourceFile,* uint32_t *lineNumber,* OTF2_RegionRef *enteredRegion,* OTF2_RegionRef *leftRegion*)

Writes a Callsite definition record into the DefWriter.

The callsite definition.

Parameters

writer	Writer object.
self	The unique identifier for this *Callsite* definition.
sourceFile	The source file where this call was made. References a *String* definition.
lineNumber	Line number in the source file where this call was made.
enteredRegion	The region which was called. References a *Region* definition.
leftRegion	The region which made the call. References a *Region* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.6 OTF2_ErrorCode OTF2_DefWriter_WriteCartCoordinate (OTF2_DefWriter * *writer*, OTF2_CartTopologyRef *cartTopology*, uint32_t *rank*, uint8_t *numberOfDimensions*, const uint32_t * *coordinates*)

Writes a CartCoordinate definition record into the DefWriter.

Defines the coordinate of the location referenced by the given rank (w.r.t. the communicator associated to the topology) in the referenced topology.

Parameters

writer	Writer object.
cartTopology	Parent *CartTopology* definition to which this one is a supplementary definition. References a *CartTopology* definition.
rank	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
numberOfDimensions	Number of dimensions.
coordinates	Coordinates, indexed by dimension.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.7 OTF2_ErrorCode OTF2_DefWriter_WriteCartDimension (OTF2_DefWriter * *writer*, OTF2_CartDimensionRef *self*, OTF2_StringRef *name*, uint32_t *size*, OTF2_CartPeriodicity *cartPeriodicity*)

Writes a CartDimension definition record into the DefWriter.

Each dimension in a Cartesian topology is composed of a global id, a name, its size, and whether it is periodic or not.

Parameters

writer	Writer object.
self	The unique identifier for this *CartDimension* definition.
name	The name of the cartesian topology dimension. References a *String* definition.
size	The size of the cartesian topology dimension.
cartPeriodicity	Periodicity of the cartesian topology dimension.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.8 OTF2_ErrorCode OTF2_DefWriter_WriteCartTopology (OTF2_DefWriter ∗ *writer*, OTF2_CartTopologyRef *self*, OTF2_StringRef *name*, OTF2_CommRef *communicator*, uint8_t *numberOfDimensions*, const OTF2_CartDimensionRef ∗ *cartDimensions*)

Writes a CartTopology definition record into the DefWriter.

Each topology is described by a global id, a reference to its name, a reference to a communicator, the number of dimensions, and references to those dimensions. The topology type is defined by the paradigm of the group referenced by the associated communicator.

Parameters

writer	Writer object.
self	The unique identifier for this *CartTopology* definition.
name	The name of the topology. References a *String* definition.
communicator	Communicator object used to create the topology. References a *Comm* definition.
numberOfDimensions	Number of dimensions.
cartDimensions	The dimensions of this topology. References a *CartDimension* definition.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.9 OTF2_ErrorCode OTF2_DefWriter_WriteClockOffset (OTF2_DefWriter ∗ *writer,* OTF2_TimeStamp *time,* int64_t *offset,* double *standardDeviation*)

Writes a ClockOffset definition record into the DefWriter.

Clock offsets are used for clock corrections.

Parameters

writer	Writer object.
time	Time when this offset was determined.
offset	The offset to the global clock which was determined at `time`.
standard-Deviation	A possible standard deviation, which can be used as a metric for the quality of the offset.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.10 OTF2_ErrorCode OTF2_DefWriter_WriteComm (OTF2_DefWriter ∗ *writer,* OTF2_CommRef *self,* OTF2_StringRef *name,* OTF2_GroupRef *group,* OTF2_CommRef *parent*)

Writes a Comm definition record into the DefWriter.

The communicator definition.

Parameters

writer	Writer object.
self	The unique identifier for this *Comm* definition.
name	The name given by calling MPI_Comm_set_name on this communicator. Or the empty name to indicate that no name was given. References a *String* definition.
group	The describing MPI group of this MPI communicator The group needs to be of type *OTF2_GROUP_TYPE_COMM_GROUP* or *OTF2_GROUP_TYPE_COMM_SELF*. References a *Group* definition.

parent	The parent MPI communicator from which this communicator was created, if any. Use *OTF2_UNDEFINED_COMM* to indicate no parent. References a *Comm* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.11 OTF2_ErrorCode OTF2_DefWriter_WriteGroup (OTF2_DefWriter * *writer*, OTF2_GroupRef *self*, OTF2_StringRef *name*, OTF2_GroupType *groupType*, OTF2_Paradigm *paradigm*, OTF2_GroupFlag *groupFlags*, uint32_t *numberOfMembers*, const uint64_t * *members*)

Writes a Group definition record into the DefWriter.

The group definition.

Parameters

writer	Writer object.
self	The unique identifier for this *Group* definition.
name	Name of this group References a *String* definition.
groupType	The type of this group. Since version 1.2.
paradigm	The paradigm of this communication group. Since version 1.2.
groupFlags	Flags for this group. Since version 1.2.
numberOfMembers	The number of members in this group.
members	The identifiers of the group members.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.12 OTF2_ErrorCode OTF2_DefWriter_WriteInterruptGenerator (OTF2_DefWriter * *writer*, OTF2_InterruptGeneratorRef *self*, OTF2_StringRef *name*, OTF2_StringRef *unit*, uint64_t *period*)

Writes a InterruptGenerator definition record into the DefWriter.

Parameters

writer	Writer object.
self	The unique identifier for this *InterruptGenerator* definition.
name	The name of this interrupt generator. References a *String* definition.
unit	The unit used by this interrupt generator for the period. References a *String* definition.
period	The period this interrupt generator generates interrupts.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.13 OTF2_ErrorCode OTF2_DefWriter_WriteLocation (OTF2_DefWriter * *writer*, OTF2_LocationRef *self*, OTF2_StringRef *name*, OTF2_LocationType *locationType*, uint64_t *numberOfEvents*, OTF2_LocationGroupRef *locationGroup*)

Writes a Location definition record into the DefWriter.

The location definition.

Parameters

writer	Writer object.
self	The unique identifier for this *Location* definition.
name	Name of the location References a *String* definition.
location-Type	Location type.
numberO-fEvents	Number of events this location has recorded.
location-Group	Location group which includes this location. References a *Location-Group* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.14 OTF2_ErrorCode OTF2_DefWriter_WriteLocationGroup (OTF2_DefWriter ∗ *writer,* OTF2_LocationGroupRef *self,* OTF2_StringRef *name,* OTF2_LocationGroupType *locationGroupType,* OTF2_SystemTreeNodeRef *systemTreeParent*)

Writes a LocationGroup definition record into the DefWriter.

The location group definition.

Parameters

writer	Writer object.
self	The unique identifier for this *LocationGroup* definition.
name	Name of the group. References a *String* definition.
location-GroupType	Type of this group.
sys-temTreePar-ent	Parent of this location group in the system tree. References a *Sys-temTreeNode* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.15 OTF2_ErrorCode OTF2_DefWriter_WriteLocationGroupProperty (OTF2_DefWriter ∗ *writer,* OTF2_LocationGroupRef *locationGroup,* OTF2_StringRef *name,* OTF2_StringRef *value*)

Writes a LocationGroupProperty definition record into the DefWriter.

An arbitrary key/value property for a *LocationGroup* definition.

Parameters

writer	Writer object.
location-Group	Parent *LocationGroup* definition to which this one is a supplementary definition. References a *LocationGroup* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.16 OTF2_ErrorCode OTF2_DefWriter_WriteLocationProperty (OTF2_DefWriter ∗ *writer*, OTF2_LocationRef *location*, OTF2_StringRef *name*, OTF2_StringRef *value*)

Writes a LocationProperty definition record into the DefWriter.

An arbitrary key/value property for a *Location* definition.

Parameters

writer	Writer object.
location	Parent *Location* definition to which this one is a supplementary definition. References a *Location* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.17 OTF2_ErrorCode OTF2_DefWriter_WriteMappingTable (OTF2_DefWriter ∗ *writer*, OTF2_MappingType *mappingType*, const OTF2_IdMap ∗ *idMap*)

Writes a MappingTable definition record into the DefWriter.

Mapping tables are needed for situations where an ID is not globally known at measurement time. They are applied automatically at reading.

Parameters

writer	Writer object.
mapping-Type	Says to what type of ID the mapping table has to be applied.
idMap	Mapping table.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.18 OTF2_ErrorCode OTF2_DefWriter_WriteMetricClass (OTF2_DefWriter * *writer*, OTF2_MetricRef *self*, uint8_t *numberOfMetrics*, const OTF2_MetricMemberRef * *metricMembers*, OTF2_MetricOccurrence *metricOccurrence*, OTF2_RecorderKind *recorderKind*)

Writes a MetricClass definition record into the DefWriter.

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

Parameters

writer	Writer object.
self	The unique identifier for this *MetricClass* definition.
numberOf-Metrics	Number of metrics within the set.
metricMem-bers	List of metric members. References a *MetricMember* definition.
metricOc-currence	Defines occurrence of a metric set.
recorderKind	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.19 OTF2_ErrorCode OTF2_DefWriter_WriteMetricClassRecorder (OTF2_DefWriter ∗ *writer*, OTF2_MetricRef *metricClass*, OTF2_LocationRef *recorder*)

Writes a MetricClassRecorder definition record into the DefWriter.

The metric class recorder definition.

Parameters

writer	Writer object.
metricClass	Parent *MetricClass* definition to which this one is a supplementary definition. References a *MetricClass* definition.
recorder	The location which recorded the referenced metric class. References a *Location* definition.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.20 OTF2_ErrorCode OTF2_DefWriter_WriteMetricInstance (OTF2_DefWriter ∗ *writer*, OTF2_MetricRef *self*, OTF2_MetricRef *metricClass*, OTF2_LocationRef *recorder*, OTF2_MetricScope *metricScope*, uint64_t *scope*)

Writes a MetricInstance definition record into the DefWriter.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type *OTF2_METRIC_ASYNCHRONOUS*.

Parameters

writer	Writer object.
self	The unique identifier for this *MetricClass* definition.
metricClass	The instanced *MetricClass*. This metric class must be of kind *OTF2_-RECORDER_KIND_ABSTRACT*. References a *MetricClass* definition.

recorder	Recorder of the metric: location ID. References a *Location* definition.
metric-Scope	Defines type of scope: location, location group, system tree node, or a generic group of locations.
scope	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.21 OTF2_ErrorCode OTF2_DefWriter_WriteMetricMember (OTF2_DefWriter * *writer*, OTF2_MetricMemberRef *self*, OTF2_StringRef *name*, OTF2_StringRef *description*, OTF2_MetricType *metricType*, OTF2_MetricMode *metricMode*, OTF2_Type *valueType*, OTF2_MetricBase *metricBase*, int64_t *exponent*, OTF2_StringRef *unit*)

Writes a MetricMember definition record into the DefWriter.

A metric is defined by a metric member definition. A metric member is always a member of a metric class. Therefore, a single metric is a special case of a metric class with only one member. It is not allowed to reference a metric member id in a metric event, but only metric class IDs.

Parameters

writer	Writer object.
self	The unique identifier for this *MetricMember* definition.
name	Name of the metric. References a *String* definition.
description	Description of the metric. References a *String* definition.
metricType	Metric type: PAPI, etc.
metricMode	Metric mode: accumulative, fix, relative, etc.
valueType	Type of the value. Only *OTF2_TYPE_INT64*, *OTF2_TYPE_UINT64*, and *OTF2_TYPE_DOUBLE* are valid types. If this metric member is recorded in an *Metric* event, than this type and the type in the event must match.
metricBase	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.

exponent	The values inside the Metric events should be scaled by the factor base^exponent, to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be *OTF2_BASE_-BINARY* and the exponent 10. Than the writer does not need to scale the values up to bytes, but can directly write the KiBi values into the Metric event. At reading time, the reader can apply the scaling factor to get the value in its base unit, ie. in bytes.
unit	Unit of the metric. This needs to be the scale free base unit, ie. "bytes", "operations", or "seconds". In particular this unit should not have any scale prefix. References a *String* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.22 OTF2_ErrorCode OTF2_DefWriter_WriteParameter (OTF2_DefWriter ∗ *writer,* OTF2_ParameterRef *self,* OTF2_StringRef *name,* OTF2_ParameterType *parameterType*)

Writes a Parameter definition record into the DefWriter.

The parameter definition.

Parameters

writer	Writer object.
self	The unique identifier for this *Parameter* definition.
name	Name of the parameter (variable name etc.) References a *String* definition.
parameter-Type	Type of the parameter, *OTF2_ParameterType* for possible types.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.23 OTF2_ErrorCode OTF2_DefWriter_WriteRegion (OTF2_DefWriter * *writer*, OTF2_RegionRef *self*, OTF2_StringRef *name*, OTF2_StringRef *canonicalName*, OTF2_StringRef *description*, OTF2_RegionRole *regionRole*, OTF2_Paradigm *paradigm*, OTF2_RegionFlag *regionFlags*, OTF2_StringRef *sourceFile*, uint32_t *beginLineNumber*, uint32_t *endLineNumber*)

Writes a Region definition record into the DefWriter.

The region definition.

Parameters

writer	Writer object.
self	The unique identifier for this *Region* definition.
name	Name of the region (demangled name if available). References a *String* definition.
canonicalName	Alternative name of the region (e.g. mangled name). References a *String* definition. Since version 1.1.
description	A more detailed description of this region. References a *String* definition.
regionRole	Region role. Since version 1.1.
paradigm	Paradigm. Since version 1.1.
regionFlags	Region flags. Since version 1.1.
sourceFile	The source file where this region was declared. References a *String* definition.
beginLineNumber	Starting line number of this region in the source file.
endLineNumber	Ending line number of this region in the source file.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.24 OTF2_ErrorCode OTF2_DefWriter_WriteRmaWin (OTF2_DefWriter * *writer*, OTF2_RmaWinRef *self*, OTF2_StringRef *name*, OTF2_CommRef *comm*)

Writes a RmaWin definition record into the DefWriter.

A window defines the communication context for any remote-memory access operation.

Parameters

writer	Writer object.
self	The unique identifier for this *RmaWin* definition.
name	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a *String* definition.
comm	Communicator object used to create the window. References a *Comm* definition.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.25 OTF2_ErrorCode OTF2_DefWriter_WriteSourceCodeLocation (OTF2_DefWriter ∗ *writer*, OTF2_SourceCodeLocationRef *self*, OTF2_StringRef *file*, uint32_t *lineNumber*)

Writes a SourceCodeLocation definition record into the DefWriter.

The definition of a source code location as tuple of the corresponding file name and line number.

When used to attach source code annotations to events, use the *OTF2_AttributeList* with a *Attribute* definition named `"SOURCE_CODE_LOCATION"` and typed *OTF2_TYPE_SOURCE_CODE_LOCATION*.

Parameters

writer	Writer object.
self	The unique identifier for this *SourceCodeLocation* definition.
file	The name of the file for the source code location. References a *String* definition.
lineNumber	The line number for the source code location.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.26 OTF2_ErrorCode OTF2_DefWriter_WriteString (OTF2_DefWriter ∗ *writer*, OTF2_StringRef *self*, const char ∗ *string*)

Writes a String definition record into the DefWriter.

The string definition.

Parameters

writer	Writer object.
self	The unique identifier for this *String* definition.
string	The string, null terminated.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.27 OTF2_ErrorCode OTF2_DefWriter_WriteSystemTreeNode (OTF2_DefWriter ∗ *writer*, OTF2_SystemTreeNodeRef *self*, OTF2_StringRef *name*, OTF2_StringRef *className*, OTF2_SystemTreeNodeRef *parent*)

Writes a SystemTreeNode definition record into the DefWriter.

The system tree node definition.

Parameters

writer	Writer object.
self	The unique identifier for this *SystemTreeNode* definition.
name	Free form instance name of this node. References a *String* definition.
className	Free form class name of this node References a *String* definition.
parent	Parent id of this node. May be *OTF2_UNDEFINED_SYSTEM_TREE_-NODE* to indicate that there is no parent. References a *SystemTreeNode* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.28 OTF2_ErrorCode OTF2_DefWriter_WriteSystemTreeNodeDomain (OTF2_DefWriter * *writer*, OTF2_SystemTreeNodeRef *systemTreeNode*, OTF2_SystemTreeDomain *systemTreeDomain*)

Writes a SystemTreeNodeDomain definition record into the DefWriter.

The system tree node domain definition.

Parameters

writer	Writer object.
systemTreeNode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
systemTreeDomain	The domain in which the referenced *SystemTreeNode* operates in.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.10.2.29 OTF2_ErrorCode OTF2_DefWriter_WriteSystemTreeNodeProperty (OTF2_DefWriter * *writer*, OTF2_SystemTreeNodeRef *systemTreeNode*, OTF2_StringRef *name*, OTF2_StringRef *value*)

Writes a SystemTreeNodeProperty definition record into the DefWriter.

An arbitrary key/value property for a *SystemTreeNode* definition.

Parameters

writer	Writer object.

systemTreeNode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.11 otf2/OTF2_Events.h File Reference

Enums and types used in event records.

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
```

Data Structures

- union OTF2_MetricValue

  Metric value.

Typedefs

- typedef uint8_t OTF2_CollectiveOp

  Wrapper for enum OTF2_CollectiveOp_enum.
- typedef uint8_t OTF2_LockType

  Wrapper for enum OTF2_LockType_enum.
- typedef uint8_t OTF2_MeasurementMode

  Wrapper for enum OTF2_MeasurementMode_enum.
- typedef uint8_t OTF2_RmaAtomicType

  Wrapper for enum OTF2_RmaAtomicType_enum.
- typedef uint32_t OTF2_RmaSyncLevel

  Wrapper for enum OTF2_RmaSyncLevel_enum.
- typedef uint8_t OTF2_RmaSyncType

  Wrapper for enum OTF2_RmaSyncType_enum.

Enumerations

- enum OTF2_CollectiveOp_enum {
  OTF2_COLLECTIVE_OP_BARRIER = 0,
  OTF2_COLLECTIVE_OP_BCAST = 1,
  OTF2_COLLECTIVE_OP_GATHER = 2,
  OTF2_COLLECTIVE_OP_GATHERV = 3,
  OTF2_COLLECTIVE_OP_SCATTER = 4,
  OTF2_COLLECTIVE_OP_SCATTERV = 5,
  OTF2_COLLECTIVE_OP_ALLGATHER = 6,
  OTF2_COLLECTIVE_OP_ALLGATHERV = 7,
  OTF2_COLLECTIVE_OP_ALLTOALL = 8,
  OTF2_COLLECTIVE_OP_ALLTOALLV = 9,
  OTF2_COLLECTIVE_OP_ALLTOALLW = 10,
  OTF2_COLLECTIVE_OP_ALLREDUCE = 11,
  OTF2_COLLECTIVE_OP_REDUCE = 12,
  OTF2_COLLECTIVE_OP_REDUCE_SCATTER = 13,
  OTF2_COLLECTIVE_OP_SCAN = 14,
  OTF2_COLLECTIVE_OP_EXSCAN = 15,
  OTF2_COLLECTIVE_OP_REDUCE_SCATTER_BLOCK = 16,
  OTF2_COLLECTIVE_OP_CREATE_HANDLE = 17,
  OTF2_COLLECTIVE_OP_DESTROY_HANDLE = 18,
  OTF2_COLLECTIVE_OP_ALLOCATE = 19,
  OTF2_COLLECTIVE_OP_DEALLOCATE = 20,
  OTF2_COLLECTIVE_OP_CREATE_HANDLE_AND_ALLOCATE = 21,
  OTF2_COLLECTIVE_OP_DESTROY_HANDLE_AND_DEALLOCATE = 22 }

  Types of collective operations.
- enum OTF2_LockType_enum {
  OTF2_LOCK_EXCLUSIVE = 0,
  OTF2_LOCK_SHARED = 1 }

  General Lock Type.
- enum OTF2_MeasurementMode_enum {
  OTF2_MEASUREMENT_ON = 1,
  OTF2_MEASUREMENT_OFF = 2 }

Types for use in the MeasurementOnOff event.

- enum OTF2_RmaAtomicType_enum {
  OTF2_RMA_ATOMIC_TYPE_ACCUMULATE = 0,
  OTF2_RMA_ATOMIC_TYPE_INCREMENT = 1,
  OTF2_RMA_ATOMIC_TYPE_TEST_AND_SET = 2,
  OTF2_RMA_ATOMIC_TYPE_COMPARE_AND_SWAP = 3,
  OTF2_RMA_ATOMIC_TYPE_SWAP = 4,
  OTF2_RMA_ATOMIC_TYPE_FETCH_AND_ADD = 5,
  OTF2_RMA_ATOMIC_TYPE_FETCH_AND_INCREMENT = 6 }

  RMA Atomic Operation Type.
- enum OTF2_RmaSyncLevel_enum {
  OTF2_RMA_SYNC_LEVEL_NONE = 0,
  OTF2_RMA_SYNC_LEVEL_PROCESS = (1 << 0),
  OTF2_RMA_SYNC_LEVEL_MEMORY = (1 << 1) }

  Synchronization level used in RMA synchronization records.
- enum OTF2_RmaSyncType_enum {
  OTF2_RMA_SYNC_TYPE_MEMORY = 0,
  OTF2_RMA_SYNC_TYPE_NOTIFY_IN = 1,
  OTF2_RMA_SYNC_TYPE_NOTIFY_OUT = 2 }

  Type of direct RMA synchronization call.

E.11.1 Detailed Description

Enums and types used in event records.

Source Template:

templates/OTF2_Events.tmpl.h

E.11.2 Enumeration Type Documentation

E.11.2.1 enum OTF2_CollectiveOp_enum

Types of collective operations.

Since

Version 1.0

Enumerator:

- ***OTF2_COLLECTIVE_OP_BARRIER*** Barrier synchronization.
- ***OTF2_COLLECTIVE_OP_BCAST*** Broadcast data from one member to all group members.
- ***OTF2_COLLECTIVE_OP_GATHER*** Gather data from all group members to one member.
- ***OTF2_COLLECTIVE_OP_GATHERV*** Gather data from all group members to one member, varying count of data from each member.
- ***OTF2_COLLECTIVE_OP_SCATTER*** Scatter data from one member to all group members.
- ***OTF2_COLLECTIVE_OP_SCATTERV*** Scatter data from one member to all group members, varying count of data from each member.
- ***OTF2_COLLECTIVE_OP_ALLGATHER*** Gather data from all group members, all members of the group will receive the result.
- ***OTF2_COLLECTIVE_OP_ALLGATHERV*** Gather data from all group members, varying count of data from each member, all members of the group will receive the result.
- ***OTF2_COLLECTIVE_OP_ALLTOALL*** Collective scatter/gather operation (complete exchange)
- ***OTF2_COLLECTIVE_OP_ALLTOALLV*** Collective scatter/gather operation (complete exchange), varying count of data from each member.
- ***OTF2_COLLECTIVE_OP_ALLTOALLW*** Collective scatter/gather operation (complete exchange), varying count of data from each member, varying data type from each member.
- ***OTF2_COLLECTIVE_OP_ALLREDUCE*** Collective reduction operation, all members of the group will receive the result.
- ***OTF2_COLLECTIVE_OP_REDUCE*** Collective reduction operation.
- ***OTF2_COLLECTIVE_OP_REDUCE_SCATTER*** Collective reduce/scatter operation, varying size of scattered blocks.
- ***OTF2_COLLECTIVE_OP_SCAN*** Collective scan operation across all members of a group.
- ***OTF2_COLLECTIVE_OP_EXSCAN*** Collective exclusive scan operation across all members of a group.
- ***OTF2_COLLECTIVE_OP_REDUCE_SCATTER_BLOCK*** Collective reduce/scatter operation.
- ***OTF2_COLLECTIVE_OP_CREATE_HANDLE*** Collectively create a handle (ie. MPI_Win, MPI_Comm, MPI_File).
- ***OTF2_COLLECTIVE_OP_DESTROY_HANDLE*** Collectively destroy a handle.

OTF2_COLLECTIVE_OP_ALLOCATE Collectively allocate memory.

OTF2_COLLECTIVE_OP_DEALLOCATE Collectively deallocate memory.

OTF2_COLLECTIVE_OP_CREATE_HANDLE_AND_ALLOCATE Collectively create a handle and allocate memory.

OTF2_COLLECTIVE_OP_DESTROY_HANDLE_AND_DEALLOCATE Collectively destroy a handle and deallocate memory.

E.11.2.2 enum OTF2_LockType_enum

General Lock Type.

Since

Version 1.2

Enumerator:

OTF2_LOCK_EXCLUSIVE Exclusive lock. No other lock will be granted.

OTF2_LOCK_SHARED Shared lock. Other shared locks will be granted, but no exclusive locks.

E.11.2.3 enum OTF2_MeasurementMode_enum

Types for use in the MeasurementOnOff event.

Since

Version 1.0

Enumerator:

OTF2_MEASUREMENT_ON The measurement resumed with event recording.

OTF2_MEASUREMENT_OFF The measurement suspended with event recording.

E.11.2.4 enum OTF2_RmaAtomicType_enum

RMA Atomic Operation Type.

Since

Version 1.2

Enumerator:

OTF2_RMA_ATOMIC_TYPE_ACCUMULATE Atomic accumulate operation.

OTF2_RMA_ATOMIC_TYPE_INCREMENT Atomic increment operation.

OTF2_RMA_ATOMIC_TYPE_TEST_AND_SET Atomic test-and-set operation.

OTF2_RMA_ATOMIC_TYPE_COMPARE_AND_SWAP Atomic compare-and-swap operation.

OTF2_RMA_ATOMIC_TYPE_SWAP Atomic swap operation.

Since

Version 1.4.

OTF2_RMA_ATOMIC_TYPE_FETCH_AND_ADD Atomic fetch-and-add operation.

Since

Version 1.4.

OTF2_RMA_ATOMIC_TYPE_FETCH_AND_INCREMENT Atomic fetch-and-increment operation.

Since

Version 1.4.

E.11.2.5 enum OTF2_RmaSyncLevel_enum

Synchronization level used in RMA synchronization records.

Since

Version 1.2

Enumerator:

OTF2_RMA_SYNC_LEVEL_NONE No process synchronization or access completion (e.g., MPI_Win_post, MPI_Win_start).

OTF2_RMA_SYNC_LEVEL_PROCESS Synchronize processes (e.g., MPI_-Win_create/free).

OTF2_RMA_SYNC_LEVEL_MEMORY Complete memory accesses (e.g., MPI_Win_complete, MPI_Win_wait).

E.11.2.6 enum OTF2_RmaSyncType_enum

Type of direct RMA synchronization call.

Since

Version 1.2

Enumerator:

OTF2_RMA_SYNC_TYPE_MEMORY Synchronize memory copy.

OTF2_RMA_SYNC_TYPE_NOTIFY_IN Incoming remote notification.

OTF2_RMA_SYNC_TYPE_NOTIFY_OUT Outgoing remote notification.

E.12 otf2/OTF2_EventSizeEstimator.h File Reference

Provides a interface to estimate the size of an resulting trace file.

```
#include <stdint.h>
#include <stdlib.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_AttributeList.h>
```

Typedefs

- typedef struct OTF2_EventSizeEstimator OTF2_EventSizeEstimator

  Keeps all necessary information about the event size estimator. See OTF2_-EventSizeEstimator_struct for detailed information.

Functions

- OTF2_ErrorCode OTF2_EventSizeEstimator_Delete (OTF2_EventSizeEstimator ∗estimator)

  Deletes an OTF2_EventSizeEstimator object.

- size_t OTF2_EventSizeEstimator_GetSizeOfAttributeList (const OTF2_EventSizeEstimator ∗estimator, const OTF2_AttributeList ∗attributeList)

  Returns the size estimate for an attribute list.
- size_t OTF2_EventSizeEstimator_GetSizeOfBufferFlushEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the BufferFlush event.
- size_t OTF2_EventSizeEstimator_GetSizeOfCallingContextSampleEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the CallingContextSample event.
- size_t OTF2_EventSizeEstimator_GetSizeOfEnterEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the Enter event.
- size_t OTF2_EventSizeEstimator_GetSizeOfLeaveEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the Leave event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMeasurementOnOffEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MeasurementOnOff event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMetricEvent (OTF2_EventSizeEstimator ∗estimator, uint8_t numberOfMetrics)

  Calculates the size estimate for the Metric event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiCollectiveBeginEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiCollectiveBegin event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiCollectiveEndEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiCollectiveEnd event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiIrecvEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiIrecv event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiIrecvRequestEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiIrecvRequest event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiIsendCompleteEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiIsendComplete event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiIsendEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiIsend event.

- size_t OTF2_EventSizeEstimator_GetSizeOfMpiRecvEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiRecv event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiRequestCancelledEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiRequestCancelled event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiRequestTestEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiRequestTest event.
- size_t OTF2_EventSizeEstimator_GetSizeOfMpiSendEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the MpiSend event.
- size_t OTF2_EventSizeEstimator_GetSizeOfOmpAcquireLockEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the OmpAcquireLock event.
- size_t OTF2_EventSizeEstimator_GetSizeOfOmpForkEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the OmpFork event.
- size_t OTF2_EventSizeEstimator_GetSizeOfOmpJoinEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the OmpJoin event.
- size_t OTF2_EventSizeEstimator_GetSizeOfOmpReleaseLockEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the OmpReleaseLock event.
- size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskCompleteEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the OmpTaskComplete event.
- size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskCreateEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the OmpTaskCreate event.
- size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskSwitchEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the OmpTaskSwitch event.
- size_t OTF2_EventSizeEstimator_GetSizeOfParameterIntEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ParameterInt event.
- size_t OTF2_EventSizeEstimator_GetSizeOfParameterStringEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ParameterString event.

- size_t OTF2_EventSizeEstimator_GetSizeOfParameterUnsignedIntEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ParameterUnsignedInt event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaAcquireLockEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaAcquireLock event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaAtomicEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaAtomic event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveBeginEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaCollectiveBegin event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveEndEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaCollectiveEnd event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaGetEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaGet event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaGroupSyncEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaGroupSync event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteBlockingEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaOpCompleteBlocking event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteNonBlockingEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaOpCompleteNonBlocking event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteRemoteEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaOpCompleteRemote event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpTestEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaOpTest event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaPutEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaPut event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaReleaseLockEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaReleaseLock event.

- size_t OTF2_EventSizeEstimator_GetSizeOfRmaRequestLockEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaRequestLock event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaSyncEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaSync event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaTryLockEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaTryLock event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaWaitChangeEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaWaitChange event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinCreateEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaWinCreate event.
- size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinDestroyEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the RmaWinDestroy event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadAcquireLockEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadAcquireLock event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadBeginEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadBegin event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadCreateEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadCreate event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadEnd event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadForkEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadFork event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadJoinEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadJoin event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadReleaseLockEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadReleaseLock event.

- size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCompleteEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadTaskComplete event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCreateEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadTaskCreate event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskSwitchEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadTaskSwitch event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamBeginEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadTeamBegin event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamEndEvent (OTF2_-EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadTeamEnd event.
- size_t OTF2_EventSizeEstimator_GetSizeOfThreadWaitEvent (OTF2_EventSizeEstimator ∗estimator)

  Calculates the size estimate for the ThreadWait event.
- size_t OTF2_EventSizeEstimator_GetSizeOfTimestamp (OTF2_EventSizeEstimator ∗estimator)

  Returns the size for an timestamp record.
- OTF2_EventSizeEstimator ∗ OTF2_EventSizeEstimator_New (void)

  Creates a new OTF2_EventSizeEstimator object.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfAttributeDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfAttributeDefinitions)

  Sets the number of Attribute definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfCallingContextDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfCallingContextDefinitions)

  Sets the number of CallingContext definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfCommDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfCommDefinitions)

  Sets the number of Comm definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfGroupDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfGroupDefinitions)

  Sets the number of Group definitions used.

- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfInterruptGeneratorDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfInterruptGeneratorDefinitions)

  Sets the number of InterruptGenerator definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfLocationDefinitions (OTF2_EventSizeEstimator ∗estimator, uint64_t numberOfLocationDefinitions)

  Sets the number of Location definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfMetricDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfMetricDefinitions)

  Sets the number of Metric definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfParameterDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfParameterDefinitions)

  Sets the number of Parameter definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfRegionDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfRegionDefinitions)

  Sets the number of Region definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfRmaWinDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfRmaWinDefinitions)

  Sets the number of RmaWin definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfSourceCodeLocationDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfSourceCodeLocationDefinitions)

  Sets the number of SourceCodeLocation definitions used.
- OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfStringDefinitions (OTF2_EventSizeEstimator ∗estimator, uint32_t numberOfStringDefinitions)

  Sets the number of String definitions used.

E.12.1 Detailed Description

Provides a interface to estimate the size of an resulting trace file.

Source Template:

templates/OTF2_EventSizeEstimator.tmpl.h

E.12.2 Function Documentation

E.12.2.1 OTF2_ErrorCode OTF2_EventSizeEstimator_Delete (OTF2_EventSizeEstimator ∗ *estimator*)

Deletes an OTF2_EventSizeEstimator object.

Parameters

estimator	Estimator object.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.2 size_t OTF2_EventSizeEstimator_GetSizeOfAttributeList (const OTF2_EventSizeEstimator ∗ *estimator,* const OTF2_AttributeList ∗ *attributeList*)

Returns the size estimate for an attribute list.

The attribute list should be filled with the used types. The attribute references are taken from the number of attribute definitions and the values are the upper bounds for integral and floating point types, and the estimates for definition references.

Parameters

estimator	Estimator object.
attributeList	Attribute List.

Returns

The estimated size.

E.12.2.3 size_t OTF2_EventSizeEstimator_GetSizeOfBufferFlushEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the BufferFlush event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.4 size_t OTF2_EventSizeEstimator_GetSizeOfCallingContextSampleEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the CallingContextSample event.

Parameters

estimator	Estimator object.

Since

Version 1.5

Returns

The estimated size.

E.12.2.5 size_t OTF2_EventSizeEstimator_GetSizeOfEnterEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the Enter event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.6 size_t OTF2_EventSizeEstimator_GetSizeOfLeaveEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the Leave event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.7 size_t OTF2_EventSizeEstimator_GetSizeOfMeasurementOnOffEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MeasurementOnOff event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.8 size_t OTF2_EventSizeEstimator_GetSizeOfMetricEvent (OTF2_EventSizeEstimator ∗ *estimator,* uint8_t *numberOfMetrics*)

Calculates the size estimate for the Metric event.

Parameters

estimator	Estimator object.
numberOf-Metrics	Number of metrics with in the set.

Since

Version 1.0

Returns

The estimated size.

E.12.2.9 size_t OTF2_EventSizeEstimator_GetSizeOfMpiCollectiveBeginEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiCollectiveBegin event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.10 size_t OTF2_EventSizeEstimator_GetSizeOfMpiCollectiveEndEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiCollectiveEnd event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.11 size_t OTF2_EventSizeEstimator_GetSizeOfMpiIrecvEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiIrecv event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.12 size_t OTF2_EventSizeEstimator_GetSizeOfMpiIrecvRequestEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiIrecvRequest event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.13 size_t OTF2_EventSizeEstimator_GetSizeOfMpiIsendCompleteEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiIsendComplete event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.14 size_t OTF2_EventSizeEstimator_GetSizeOfMpiIsendEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiIsend event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.15 size_t OTF2_EventSizeEstimator_GetSizeOfMpiRecvEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiRecv event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.16 size_t OTF2_EventSizeEstimator_GetSizeOfMpiRequestCancelledEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiRequestCancelled event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.17 size_t OTF2_EventSizeEstimator_GetSizeOfMpiRequestTestEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiRequestTest event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.18 size_t OTF2_EventSizeEstimator_GetSizeOfMpiSendEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the MpiSend event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.19 size_t OTF2_EventSizeEstimator_GetSizeOfOmpAcquireLockEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the OmpAcquireLock event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.12.2.20 size_t OTF2_EventSizeEstimator_GetSizeOfOmpForkEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the OmpFork event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.12.2.21 size_t OTF2_EventSizeEstimator_GetSizeOfOmpJoinEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the OmpJoin event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.12.2.22 size_t OTF2_EventSizeEstimator_GetSizeOfOmpReleaseLockEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the OmpReleaseLock event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.12.2.23 size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskCompleteEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the OmpTaskComplete event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.12.2.24 size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskCreateEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the OmpTaskCreate event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.12.2.25 size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskSwitchEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the OmpTaskSwitch event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.12.2.26 size_t OTF2_EventSizeEstimator_GetSizeOfParameterIntEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ParameterInt event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.27 size_t OTF2_EventSizeEstimator_GetSizeOfParameterStringEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ParameterString event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.28 size_t OTF2_EventSizeEstimator_GetSizeOfParameterUnsignedIntEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ParameterUnsignedInt event.

Parameters

estimator	Estimator object.

Since

Version 1.0

Returns

The estimated size.

E.12.2.29 size_t OTF2_EventSizeEstimator_GetSizeOfRmaAcquireLockEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaAcquireLock event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.30 size_t OTF2_EventSizeEstimator_GetSizeOfRmaAtomicEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaAtomic event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.31 size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveBeginEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaCollectiveBegin event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.32 size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveEndEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaCollectiveEnd event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.33 size_t OTF2_EventSizeEstimator_GetSizeOfRmaGetEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaGet event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.34 size_t OTF2_EventSizeEstimator_GetSizeOfRmaGroupSyncEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaGroupSync event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.35 size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteBlockingEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaOpCompleteBlocking event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.36 size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteNonBlockingEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaOpCompleteNonBlocking event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.37 size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteRemoteEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaOpCompleteRemote event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.38 size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpTestEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaOpTest event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.39 size_t OTF2_EventSizeEstimator_GetSizeOfRmaPutEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaPut event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.40 size_t OTF2_EventSizeEstimator_GetSizeOfRmaReleaseLockEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaReleaseLock event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.41 size_t OTF2_EventSizeEstimator_GetSizeOfRmaRequestLockEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the RmaRequestLock event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.42 size_t OTF2_EventSizeEstimator_GetSizeOfRmaSyncEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the RmaSync event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.43 size_t OTF2_EventSizeEstimator_GetSizeOfRmaTryLockEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaTryLock event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.44 size_t OTF2_EventSizeEstimator_GetSizeOfRmaWaitChangeEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaWaitChange event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.45 size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinCreateEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaWinCreate event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.46 size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinDestroyEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the RmaWinDestroy event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.47 size_t OTF2_EventSizeEstimator_GetSizeOfThreadAcquireLockEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadAcquireLock event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.48 size_t OTF2_EventSizeEstimator_GetSizeOfThreadBeginEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadBegin event.

Parameters

estimator	Estimator object.

Since

Version 1.3

Returns

The estimated size.

E.12.2.49 size_t OTF2_EventSizeEstimator_GetSizeOfThreadCreateEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadCreate event.

Parameters

estimator	Estimator object.

Since

Version 1.3

Returns

The estimated size.

E.12.2.50 size_t OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadEnd event.

Parameters

estimator	Estimator object.

Since

Version 1.3

Returns

The estimated size.

E.12.2.51 size_t OTF2_EventSizeEstimator_GetSizeOfThreadForkEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadFork event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.52 size_t OTF2_EventSizeEstimator_GetSizeOfThreadJoinEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadJoin event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.53 size_t OTF2_EventSizeEstimator_GetSizeOfThreadReleaseLockEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadReleaseLock event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.54 size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCompleteEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadTaskComplete event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.55 size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCreateEvent (OTF2_EventSizeEstimator ∗ *estimator*)

Calculates the size estimate for the ThreadTaskCreate event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.56 size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskSwitchEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the ThreadTaskSwitch event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.57 size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamBeginEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the ThreadTeamBegin event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.58 size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamEndEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the ThreadTeamEnd event.

Parameters

estimator	Estimator object.

Since

Version 1.2

Returns

The estimated size.

E.12.2.59 size_t OTF2_EventSizeEstimator_GetSizeOfThreadWaitEvent (OTF2_EventSizeEstimator * *estimator*)

Calculates the size estimate for the ThreadWait event.

Parameters

estimator	Estimator object.

Since

Version 1.3

Returns

The estimated size.

E.12.2.60 size_t OTF2_EventSizeEstimator_GetSizeOfTimestamp (OTF2_EventSizeEstimator * *estimator*)

Returns the size for an timestamp record.

OTF2 does only store a timestamp, if it changed between two events.

Parameters

estimator	Estimator object.

Returns

The estimated size.

E.12.2.61 OTF2_EventSizeEstimator∗ OTF2_EventSizeEstimator_New (void)

Creates a new OTF2_EventSizeEstimator object.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.62 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfAttributeDefinitions (OTF2_EventSizeEstimator ∗ *estimator,* uint32_t *numberOfAttributeDefinitions*)

Sets the number of Attribute definitions used.

Definition ids are considered to be in the range [0, numberOfAttributeDefinitions).

Parameters

estimator	Estimator object.
numberOfAttributeDefinitions	The number of definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.63 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfCallingContextDefinitions (OTF2_EventSizeEstimator ∗ *estimator,* uint32_t *numberOfCallingContextDefinitions*)

Sets the number of CallingContext definitions used.

Definition ids are considered to be in the range [0, numberOfCallingContextDefinitions).

Parameters

estimator	Estimator object.
numberOf-Calling-ContextDef-initions	The number of definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.64 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfCommDefinitions (OTF2_EventSizeEstimator ∗ *estimator,* uint32_t *numberOfCommDefinitions*)

Sets the number of Comm definitions used.

Definition ids are considered to be in the range [0, numberOfCommDefinitions).

Parameters

estimator	Estimator object.
numberOf-CommDefi-nitions	The number of definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.65 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfGroupDefinitions (OTF2_EventSizeEstimator ∗ *estimator,* uint32_t *numberOfGroupDefinitions*)

Sets the number of Group definitions used.

Definition ids are considered to be in the range [0, numberOfGroupDefinitions).

Parameters

estimator	Estimator object.
numberOfGroupDefinitions	The number of definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.66 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfInterruptGeneratorDefinitions (OTF2_EventSizeEstimator * *estimator*, uint32_t *numberOfInterruptGeneratorDefinitions*)

Sets the number of InterruptGenerator definitions used.

Definition ids are considered to be in the range [0, numberOfInterruptGeneratorDefinitions).

Parameters

estimator	Estimator object.
numberOfInterruptGeneratorDefinitions	The number of definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.67 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfLocationDefinitions (OTF2_EventSizeEstimator * *estimator*, uint64_t *numberOfLocationDefinitions*)

Sets the number of Location definitions used.

Definition ids are considered to be in the range [0, numberOfLocationDefinitions).

Parameters

estimator	Estimator object.
numberOfLocationDefinitions	The number of definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.68 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfMetricDefinitions (OTF2_EventSizeEstimator ∗ *estimator*, uint32_t *numberOfMetricDefinitions*)

Sets the number of Metric definitions used.

Definition ids are considered to be in the range [0, numberOfMetricDefinitions).

Parameters

estimator	Estimator object.
numberOfMetricDefinitions	The number of definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.69 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfParameterDefinitions (OTF2_EventSizeEstimator * *estimator*, uint32_t *numberOfParameterDefinitions*)

Sets the number of Parameter definitions used.

Definition ids are considered to be in the range [0, numberOfParameterDefinitions).

Parameters

estimator	Estimator object.
numberOfParameterDefinitions	The number of definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.70 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfRegionDefinitions (OTF2_EventSizeEstimator * *estimator*, uint32_t *numberOfRegionDefinitions*)

Sets the number of Region definitions used.

Definition ids are considered to be in the range [0, numberOfRegionDefinitions).

Parameters

estimator	Estimator object.
numberOfRegionDefinitions	The number of definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.71 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfRmaWinDefinitions (OTF2_EventSizeEstimator ∗ *estimator*, uint32_t *numberOfRmaWinDefinitions*)

Sets the number of RmaWin definitions used.

Definition ids are considered to be in the range [0, numberOfRmaWinDefinitions).

Parameters

estimator	Estimator object.
numberOfRmaWinDefinitions	The number of definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.72 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfSourceCodeLocationDefinitions (OTF2_EventSizeEstimator ∗ *estimator*, uint32_t *numberOfSourceCodeLocationDefinitions*)

Sets the number of SourceCodeLocation definitions used.

Definition ids are considered to be in the range [0, numberOfSourceCodeLocationDefinitions).

Parameters

estimator	Estimator object.
numberOfSourceCodeLocationDefinitions	The number of definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.12.2.73 OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfStringDefinitions (OTF2_EventSizeEstimator ∗ *estimator,* uint32_t *numberOfStringDefinitions*)

Sets the number of String definitions used.

Definition ids are considered to be in the range [0, numberOfStringDefinitions).

Parameters

estimator	Estimator object.
numberOf-StringDefi-nitions	The number of definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13 otf2/OTF2_EvtReader.h File Reference

This is the local event reader, which reads events from one location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_EvtReaderCallbacks.h>
```

Functions

- OTF2_ErrorCode OTF2_EvtReader_ApplyClockOffsets (OTF2_EvtReader ∗reader, bool action)

Enable or disable applying of the clock offset to event timestamps read from this event reader.

- OTF2_ErrorCode OTF2_EvtReader_ApplyMappingTables (OTF2_EvtReader ∗reader, bool action)

  Enable or disable applying of the mapping tabes to events read from this event reader.

- OTF2_ErrorCode OTF2_EvtReader_GetLocationID (const OTF2_EvtReader ∗reader, OTF2_LocationRef ∗location)

  Return the location ID of the reading related location.

- OTF2_ErrorCode OTF2_EvtReader_GetPos (OTF2_EvtReader ∗reader, uint64_-t ∗position)

  The following function can be used to get the position (number of the event in the stream) of last read event.

- OTF2_ErrorCode OTF2_EvtReader_ReadEvents (OTF2_EvtReader ∗reader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  After callback registration, the local events could be read with the following function. Readn reads recordsToRead records. The reader indicates that it reached the end of the trace by just reading less records than requested.

- OTF2_ErrorCode OTF2_EvtReader_ReadEventsBackward (OTF2_EvtReader ∗reader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  This functions reads recordsRead events backwards from the current position.

- OTF2_ErrorCode OTF2_EvtReader_Seek (OTF2_EvtReader ∗reader, uint64_-t position)

  Seek jumps to an event position.

- OTF2_ErrorCode OTF2_EvtReader_SetCallbacks (OTF2_EvtReader ∗reader, const OTF2_EvtReaderCallbacks ∗callbacks, void ∗userData)

  Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

- OTF2_ErrorCode OTF2_EvtReader_TimeStampRewrite (OTF2_EvtReader ∗reader, OTF2_TimeStamp time)

  The following function rewrites the timestamp from the event on the actual reading position if the buffer is in OTF2_BUFFER_MODIFY mode. It also modifies the timestamp for all other events in the same timestamp bundle. This function also has to keep track that not only the last timestamp, but all records equal to the last timestamp has to be modified. This is done by a position list, if there has no seek appeared before. In this case a position list can be easily generated because of that the reader has seen all related timestamps before. This not the case if there has a seek appeared before. In this case the related timestamp positions are generated by a linear search.

E.13.1 Detailed Description

This is the local event reader, which reads events from one location.

E.13.2 Function Documentation

E.13.2.1 OTF2_ErrorCode OTF2_EvtReader_ApplyClockOffsets (OTF2_EvtReader ∗ *reader,* bool *action*)

Enable or disable applying of the clock offset to event timestamps read from this event reader.

This setting has no effect if the events are read by an global event reader.

Parameters

reader	Reader object.
action	Truth value whether the clock offsets should be applied or not.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13.2.2 OTF2_ErrorCode OTF2_EvtReader_ApplyMappingTables (OTF2_EvtReader ∗ *reader,* bool *action*)

Enable or disable applying of the mapping tabes to events read from this event reader.

This setting has no effect if the events are read by an global event reader.

Parameters

reader	Reader object.
action	Truth value whether the mappings should be applied or not.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13.2.3 OTF2_ErrorCode OTF2_EvtReader_GetLocationID (const OTF2_EvtReader ∗ *reader,* OTF2_LocationRef ∗ *location*)

Return the location ID of the reading related location.

Parameters

	reader	Reader object which reads the events from its buffer.
`out`	*location*	ID of the location.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13.2.4 OTF2_ErrorCode OTF2_EvtReader_GetPos (OTF2_EvtReader * *reader*, uint64_t * *position*)

The following function can be used to get the position (number of the event in the stream) of last read event.

Parameters

	reader	Reader object which reads the events from its buffer.
`out`	*position*	Number of the event in the stream.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13.2.5 OTF2_ErrorCode OTF2_EvtReader_ReadEvents (OTF2_EvtReader * *reader*, uint64_t *recordsToRead*, uint64_t * *recordsRead*)

After callback registration, the local events could be read with the following function. Readn reads *recordsToRead* records. The reader indicates that it reached the end of the trace by just reading less records than requested.

Parameters

	reader	Reader object which reads the events from its buffer.
	record-sToRead	How many records can be read next.
`out`	*record-sRead*	Return how many records where really read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13.2.6 OTF2_ErrorCode OTF2_EvtReader_ReadEventsBackward (OTF2_EvtReader ∗ *reader,* uint64_t *recordsToRead,* uint64_t ∗ *recordsRead*)

This functions reads recordsRead events backwards from the current position.

Parameters

	reader	Reader object which reads the events from its buffer.
	record-sToRead	How many records can be read next.
out	*record-sRead*	Return how many records where really read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13.2.7 OTF2_ErrorCode OTF2_EvtReader_Seek (OTF2_EvtReader ∗ *reader,* uint64_t *position*)

Seek jumps to an event position.

Parameters

reader	Reader object which reads the events from its buffer.
position	Number of the event, where the reader has to jump.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13.2.8 OTF2_ErrorCode OTF2_EvtReader_SetCallbacks (OTF2_EvtReader ∗ *reader,* const OTF2_EvtReaderCallbacks ∗ *callbacks,* void ∗ *userData*)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

These callbacks are ignored, if the events are read by an global event reader.

Parameters

reader	Reader object which reads the events from its buffer.

callbacks	Struct which holds a function pointer for each record type. *OTF2_-EvtReaderCallbacks_New*.
userData	Data passed as argument *userData* to the record callbacks.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.13.2.9 OTF2_ErrorCode OTF2_EvtReader_TimeStampRewrite (OTF2_EvtReader ∗ *reader*, OTF2_TimeStamp *time*)

The following function rewrites the timestamp from the event on the actual reading position if the buffer is in OTF2_BUFFER_MODIFY mode. It also modifies the timestamp for all other events in the same timestamp bundle. This function also has to keep track that not only the last timestamp, but all records equal to the last timestamp has to be modified. This is done by a position list, if there has no seek appeared before. In this case a position list can be easily generated because of that the reader has seen all related timestamps before. This not the case if there has a seek appeared before. In this case the related timestamp positions are generated by a linear search.

Parameters

reader	Reader object which reads the events from its buffer.
time	New timestamp

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.14 otf2/OTF2_EvtReaderCallbacks.h File Reference

This defines the callbacks for the event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_BufferFlush)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp stopTime)

  Callback for the BufferFlush event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_CallingContextSample)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CallingContextRef callingContext, uint32_t unwindDistance, OTF2_InterruptGeneratorRef interruptGenerator)

  Callback for the CallingContextSample event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_Enter)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RegionRef region)

  Callback for the Enter event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_Leave)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RegionRef region)

  Callback for the Leave event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MeasurementOnOff)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_MeasurementMode measurementMode)

  Callback for the MeasurementOnOff event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_Metric)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_MetricRef metric, uint8_-t numberOfMetrics, const OTF2_Type ∗typeIDs, const OTF2_MetricValue ∗metricValues)

  Callback for the Metric event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList)

  Callback for the MpiCollectiveBegin event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

  Callback for the MpiCollectiveEnd event record.

- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiIrecv)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

  Callback for the MpiIrecv event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

  Callback for the MpiIrecvRequest event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiIsend)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

  Callback for the MpiIsend event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

  Callback for the MpiIsendComplete event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiRecv)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Callback for the MpiRecv event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiRequestCancelled)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

  Callback for the MpiRequestCancelled event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiRequestTest)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

  Callback for the MpiRequestTest event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiSend)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Callback for the MpiSend event record.

- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t lockID, uint32_t acquisitionOrder)

  Callback for the OmpAcquireLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpFork)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t numberOfRequestedThreads)

  Callback for the OmpFork event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpJoin)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList)

  Callback for the OmpJoin event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t lockID, uint32_t acquisitionOrder)

  Callback for the OmpReleaseLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpTaskComplete)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

  Callback for the OmpTaskComplete event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

  Callback for the OmpTaskCreate event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

  Callback for the OmpTaskSwitch event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ParameterInt)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, int64_t value)

  Callback for the ParameterInt event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, OTF2_StringRef string)

  Callback for the ParameterString event record.

- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, uint64_t value)

    Callback for the ParameterUnsignedInt event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

    Callback for the RmaAcquireLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaAtomic)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_-t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)

    Callback for the RmaAtomic event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList)

    Callback for the RmaCollectiveBegin event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_-t root, uint64_t bytesSent, uint64_t bytesReceived)

    Callback for the RmaCollectiveEnd event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaGet)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_-t remote, uint64_t bytes, uint64_t matchingId)

    Callback for the RmaGet event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaGroupSync)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)

    Callback for the RmaGroupSync event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaOpCompleteBlocking)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

    Callback for the RmaOpCompleteBlocking event record.

- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

  Callback for the RmaOpCompleteNonBlocking event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaOpCompleteRemote)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

  Callback for the RmaOpCompleteRemote event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaOpTest)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_-t matchingId)

  Callback for the RmaOpTest event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaPut)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_-t remote, uint64_t bytes, uint64_t matchingId)

  Callback for the RmaPut event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)

  Callback for the RmaReleaseLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaRequestLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

  Callback for the RmaRequestLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaSync)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_-t remote, OTF2_RmaSyncType syncType)

  Callback for the RmaSync event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaTryLock)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_-t remote, uint64_t lockId, OTF2_LockType lockType)

  Callback for the RmaTryLock event record.

- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaWaitChange)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

  Callback for the RmaWaitChange event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaWinCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

  Callback for the RmaWinCreate event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaWinDestroy)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

  Callback for the RmaWinDestroy event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

  Callback for the ThreadAcquireLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadBegin)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

  Callback for the ThreadBegin event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadCreate)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

  Callback for the ThreadCreate event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadEnd)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

  Callback for the ThreadEnd event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadFork)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_-t numberOfRequestedThreads)

  Callback for the ThreadFork event record.

- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadJoin)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model)

  Callback for the ThreadJoin event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

  Callback for the ThreadReleaseLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTaskComplete)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Callback for the ThreadTaskComplete event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Callback for the ThreadTaskCreate event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Callback for the ThreadTaskSwitch event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTeamBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam)

  Callback for the ThreadTeamBegin event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTeamEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam)

  Callback for the ThreadTeamEnd event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadWait)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

  Callback for the ThreadWait event record.
- typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_Unknown)(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for an unknown event record.

- typedef struct OTF2_EvtReaderCallbacks_struct OTF2_EvtReaderCallbacks

  Opaque struct which holdes all event record callbacks.

Functions

- void OTF2_EvtReaderCallbacks_Clear (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks)

  Clears a struct for the event callbacks.
- void OTF2_EvtReaderCallbacks_Delete (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks)

  Deallocates a struct for the event callbacks.
- OTF2_EvtReaderCallbacks ∗ OTF2_EvtReaderCallbacks_New (void)

  Allocates a new struct for the event callbacks.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetBufferFlushCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_BufferFlush bufferFlushCallback)

  Registers the callback for the BufferFlush event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetCallingContextSampleCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-CallingContextSample callingContextSampleCallback)

  Registers the callback for the CallingContextSample event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetEnterCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_Enter enterCallback)

  Registers the callback for the Enter event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetLeaveCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_Leave leaveCallback)

  Registers the callback for the Leave event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMeasurementOnOffCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-MeasurementOnOff measurementOnOffCallback)

  Registers the callback for the MeasurementOnOff event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMetricCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_Metric metricCallback)

  Registers the callback for the Metric event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-MpiCollectiveBegin mpiCollectiveBeginCallback)

Registers the callback for the MpiCollectiveBegin event.

- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-MpiCollectiveEnd mpiCollectiveEndCallback)

  Registers the callback for the MpiCollectiveEnd event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIrecvCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_MpiIrecv mpiIrecvCallback)

  Registers the callback for the MpiIrecv event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-MpiIrecvRequest mpiIrecvRequestCallback)

  Registers the callback for the MpiIrecvRequest event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIsendCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_MpiIsend mpiIsendCallback)

  Registers the callback for the MpiIsend event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-MpiIsendComplete mpiIsendCompleteCallback)

  Registers the callback for the MpiIsendComplete event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRecvCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_MpiRecv mpiRecvCallback)

  Registers the callback for the MpiRecv event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRequestCancelledCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-MpiRequestCancelled mpiRequestCancelledCallback)

  Registers the callback for the MpiRequestCancelled event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRequestTestCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-MpiRequestTest mpiRequestTestCallback)

  Registers the callback for the MpiRequestTest event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiSendCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_MpiSend mpiSendCallback)

  Registers the callback for the MpiSend event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-OmpAcquireLock ompAcquireLockCallback)

  Registers the callback for the OmpAcquireLock event.

- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpForkCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_OmpFork ompForkCallback)

  Registers the callback for the OmpFork event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpJoinCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_OmpJoin ompJoinCallback)

  Registers the callback for the OmpJoin event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-OmpReleaseLock ompReleaseLockCallback)

  Registers the callback for the OmpReleaseLock event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-OmpTaskComplete ompTaskCompleteCallback)

  Registers the callback for the OmpTaskComplete event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_OmpTaskCreate ompTaskCreateCallback)

  Registers the callback for the OmpTaskCreate event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpTaskSwitchCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-OmpTaskSwitch ompTaskSwitchCallback)

  Registers the callback for the OmpTaskSwitch event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetParameterIntCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_ParameterInt parameterIntCallback)

  Registers the callback for the ParameterInt event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetParameterStringCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_ParameterString parameterStringCallback)

  Registers the callback for the ParameterString event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetParameterUnsignedIntCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-ParameterUnsignedInt parameterUnsignedIntCallback)

  Registers the callback for the ParameterUnsignedInt event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaAcquireLock rmaAcquireLockCallback)

  Registers the callback for the RmaAcquireLock event.

- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaAtomicCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_RmaAtomic rmaAtomicCallback)

  Registers the callback for the RmaAtomic event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaCollectiveBegin rmaCollectiveBeginCallback)

  Registers the callback for the RmaCollectiveBegin event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaCollectiveEnd rmaCollectiveEndCallback)

  Registers the callback for the RmaCollectiveEnd event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaGetCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_RmaGet rmaGetCallback)

  Registers the callback for the RmaGet event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_RmaGroupSync rmaGroupSyncCallback)

  Registers the callback for the RmaGroupSync event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteBlockingCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaOpCompleteBlocking rmaOpCompleteBlockingCallback)

  Registers the callback for the RmaOpCompleteBlocking event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaOpCompleteNonBlocking rmaOpCompleteNonBlockingCallback)

  Registers the callback for the RmaOpCompleteNonBlocking event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteRemoteCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaOpCompleteRemote rmaOpCompleteRemoteCallback)

  Registers the callback for the RmaOpCompleteRemote event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpTestCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_RmaOpTest rmaOpTestCallback)

  Registers the callback for the RmaOpTest event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaPutCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_RmaPut rmaPutCallback)

  Registers the callback for the RmaPut event.

- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaReleaseLockCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaReleaseLock rmaReleaseLockCallback)

  Registers the callback for the RmaReleaseLock event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaRequestLockCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaRequestLock rmaRequestLockCallback)

  Registers the callback for the RmaRequestLock event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaSyncCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_RmaSync rmaSyncCallback)

  Registers the callback for the RmaSync event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaTryLockCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_RmaTryLock rmaTryLockCallback)

  Registers the callback for the RmaTryLock event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaWaitChangeCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaWaitChange rmaWaitChangeCallback)

  Registers the callback for the RmaWaitChange event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaWinCreateCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_RmaWinCreate rmaWinCreateCallback)

  Registers the callback for the RmaWinCreate event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaWinDestroyCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-RmaWinDestroy rmaWinDestroyCallback)

  Registers the callback for the RmaWinDestroy event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadAcquireLockCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-ThreadAcquireLock threadAcquireLockCallback)

  Registers the callback for the ThreadAcquireLock event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadBeginCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_ThreadBegin threadBeginCallback)

  Registers the callback for the ThreadBegin event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadCreateCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_ThreadCreate threadCreateCallback)

  Registers the callback for the ThreadCreate event.

- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadEndCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_ThreadEnd threadEndCallback)

  Registers the callback for the ThreadEnd event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadForkCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_ThreadFork threadForkCallback)

  Registers the callback for the ThreadFork event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadJoinCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_ThreadJoin threadJoinCallback)

  Registers the callback for the ThreadJoin event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadReleaseLockCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-ThreadReleaseLock threadReleaseLockCallback)

  Registers the callback for the ThreadReleaseLock event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTaskCompleteCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-ThreadTaskComplete threadTaskCompleteCallback)

  Registers the callback for the ThreadTaskComplete event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTaskCreateCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-ThreadTaskCreate threadTaskCreateCallback)

  Registers the callback for the ThreadTaskCreate event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTaskSwitchCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-ThreadTaskSwitch threadTaskSwitchCallback)

  Registers the callback for the ThreadTaskSwitch event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTeamBeginCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-ThreadTeamBegin threadTeamBeginCallback)

  Registers the callback for the ThreadTeamBegin event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTeamEndCallback (OTF2_EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_-ThreadTeamEnd threadTeamEndCallback)

  Registers the callback for the ThreadTeamEnd event.
- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadWaitCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_ThreadWait threadWaitCallback)

  Registers the callback for the ThreadWait event.

- OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetUnknownCallback (OTF2_-EvtReaderCallbacks ∗evtReaderCallbacks, OTF2_EvtReaderCallback_Unknown unknownCallback)

  Registers the callback for the Unknown event.

E.14.1 Detailed Description

This defines the callbacks for the event reader.

Source Template:

templates/OTF2_EvtReaderCallbacks.tmpl.h

E.14.2 Typedef Documentation

E.14.2.1 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-BufferFlush)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp stopTime)

Callback for the BufferFlush event record.

This event signals that the internal buffer was flushed at the given time.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
stopTime	The time the buffer flush finished.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.2 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-CallingContextSample)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CallingContextRef callingContext, uint32_t unwindDistance, OTF2_InterruptGeneratorRef interruptGenerator)

Callback for the CallingContextSample event record.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
callingContext	References a *CallingContext* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_CALLING_CONTEXT* is available.
unwindDistance	The unwindContext specifies the first context whose ip(return adress) was still marked since the last sample this means that no progress was made in the repsective region The last region that was not returned from since the last sample Is one stack level higher, but may now be at at different line number OTF2_CallingContextRef unwindContext; However, instead of this we specify the distance (number of intermediate edges) between the calling context and the unwind context Note: unwindDistance=0 would mean no progress in the leaf region since the last sample which is unlikely If not available, UNDEFINED should be used.
interruptGenerator	References a *InterruptGenerator* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_INTERRUPT_GENERATOR* is available.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.3 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-Enter)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RegionRef region)

Callback for the Enter event record.

An enter record indicates that the program enters a code region.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.4 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-Leave)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RegionRef region)

Callback for the Leave event record.

A leave record indicates that the program leaves a code region.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.

userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.5 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MeasurementOnOff)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_MeasurementMode measurementMode)

Callback for the MeasurementOnOff event record.

This event signals where the measurement system turned measurement on or off.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosi-tion	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
measure-mentMode	Is the measurement turned on (*OTF2_MEASUREMENT_ON*) or off (*OTF2_MEASUREMENT_OFF*)?

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.6 typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_-Metric)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)

Callback for the Metric event record.

A metric event is always stored at the location that recorded the metric. A metric event can reference a metric class or metric instance. Therefore, metric classes and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
metric	Could be a metric class or a metric instance. References a *MetricClass*, or a *MetricInstance* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_METRIC* is available.
numberOfMetrics	Number of metrics with in the set.
typeIDs	List of metric types. These types must match that of the corresponding *MetricMember* definitions.
metricValues	List of metric values.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.7 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for the MpiCollectiveBegin event record.

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI_-GATHER, MPI_SCATTER etc.).

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosi-tion	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.8 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

Callback for the MpiCollectiveEnd event record.

A MpiCollectiveEnd record marks the end of an MPI collective operation (MPI_-GATHER, MPI_SCATTER etc.). It keeps the necessary information for this event: type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

Parameters

location	The location where this event happened.
time	The time when this event happened.

eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
collectiveOp	Determines which collective operation it is.
communicator	Communicator References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
root	MPI rank of root in `communicator`.
sizeSent	Size of the sent message.
sizeReceived	Size of the received message.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.9 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiIrecv)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIrecv event record.

A MpiIrecv record indicates that a MPI message was received (MPI_IRECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.

attributeList	Additional attributes for this event.
sender	MPI rank of sender in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.10 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

Callback for the MpiIrecvRequest event record.

Signals the request of an receive, which can be completed later.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosi-tion	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
requestID	ID of the requested receive

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.11 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MpiIsend)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIsend event record.

A MpiIsend record indicates that a MPI message send process was initiated (MPI_-ISEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
receiver	MPI rank of receiver in `communicator`.
communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.12 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

Callback for the MpiIsendComplete event record.

Signals the completion of non-blocking send request.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.13 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_MpiRecv)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiRecv event record.

A MpiRecv record indicates that a MPI message was received (MPI_RECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
sender	MPI rank of sender in `communicator`.
communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.

msgTag	Message tag
msgLength	Message length

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.14 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MpiRequestCancelled)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

Callback for the MpiRequestCancelled event record.

This events appears if the program canceled a request.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.15 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MpiRequestTest)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

Callback for the MpiRequestTest event record.

This events appears if the program tests if a request has already completed but the test failed.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.16 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-MpiSend)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiSend event record.

A MpiSend record indicates that a MPI message send process was initiated (MPI_-SEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

Parameters

location	The location where this event happened.

time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
receiver	MPI rank of receiver in `communicator`.
communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.17 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the OmpAcquireLock event record.

An OmpAcquireLock record marks that a thread acquires an OpenMP lock.

This event record is superseded by the *ThreadAcquireLock* event record and should not be used when the *ThreadAcquireLock* event record is in use.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
lockID	ID of the lock.

acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.18 typedef OTF2_CallbackCode($*$ OTF2_EvtReaderCallback_OmpFork)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void $*$userData, OTF2_AttributeList $*$attributeList, uint32_t numberOfRequestedThreads)

Callback for the OmpFork event record.

An OmpFork record marks that an OpenMP Thread forks a thread team.

This event record is superseded by the *ThreadFork* event record and should not be used when the *ThreadFork* event record is in use.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
numberOfRequestedThreads	Requested size of the team.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.19 typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_-OmpJoin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)

Callback for the OmpJoin event record.

An OmpJoin record marks that a team of threads is joint and only the master thread continues execution.

This event record is superseded by the *ThreadJoin* event record and should not be used when the *ThreadJoin* event record is in use.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.20 typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_-OmpReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the OmpReleaseLock event record.

An OmpReleaseLock record marks that a thread releases an OpenMP lock.

This event record is superseded by the *ThreadReleaseLock* event record and should not be used when the *ThreadReleaseLock* event record is in use.

Parameters

location	The location where this event happened.
time	The time when this event happened.

eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.21 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_OmpTaskComplete)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

Callback for the OmpTaskComplete event record.

An OmpTaskComplete record indicates that the execution of an OpenMP task has finished.

This event record is superseded by the *ThreadTaskComplete* event record and should not be used when the *ThreadTaskComplete* event record is in use.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
taskID	Identifier of the completed task instance.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.22 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

Callback for the OmpTaskCreate event record.

An OmpTaskCreate record marks that an OpenMP Task was/will be created in the current region.

This event record is superseded by the *ThreadTaskCreate* event record and should not be used when the *ThreadTaskCreate* event record is in use.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
taskID	Identifier of the newly created task instance.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.23 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

Callback for the OmpTaskSwitch event record.

An OmpTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

This event record is superseded by the *ThreadTaskSwitch* event record and should not be used when the *ThreadTaskSwitch* event record is in use.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
taskID	Identifier of the now active task instance.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.24 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, int64_t value)

Callback for the ParameterInt event record.

A ParameterInt record marks that in the current region, the specified integer parameter has the specified value.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.25 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, OTF2_StringRef string)

Callback for the ParameterString event record.

A ParameterString record marks that in the current region, the specified string parameter has the specified value.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
string	Value: Handle of a string definition References a *String* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_STRING* is available.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.26 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-ParameterUnsignedInt)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, uint64_t value)

Callback for the ParameterUnsignedInt event record.

A ParameterUnsignedInt record marks that in the current region, the specified unsigned integer parameter has the specified value.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.27 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

Callback for the RmaAcquireLock event record.

An RmaAcquireLock record denotes the time a lock was acquired by the process.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.28 typedef OTF2_CallbackCode($*$ OTF2_EvtReaderCallback_RmaAtomic)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void $*$userData, OTF2_AttributeList $*$attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)

Callback for the RmaAtomic event record.

An RmaAtomic record denotes the time a atomic operation was issued.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.
type	Type of atomic operation.
bytesSent	Bytes sent to target.
bytesReceived	Bytes received from target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.29 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for the RmaCollectiveBegin event record.

An RmaCollectiveBegin record denotes the beginnig of a collective RMA operation.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.30 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)

Callback for the RmaCollectiveEnd event record.

An RmaCollectiveEnd record denotes the end of a collective RMA operation.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
collectiveOp	Determines which collective operation it is.
syncLevel	Synchronization level of this collective operation.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
root	Root process for this operation.
bytesSent	Bytes sent in operation.
bytesReceived	Bytes receives in operation.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.31 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaGet)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)

Callback for the RmaGet event record.

An RmaGet record denotes the time a get operation was issued.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.
bytes	Bytes received from target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.32 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaGroupSync)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)

Callback for the RmaGroupSync event record.

An RmaGroupSync record denotes the synchronization with a subgroup of processes on a window.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.

attributeList	Additional attributes for this event.
syncLevel	Synchronization level of this collective operation.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
group	Group of remote processes involved in synchronization. References a *Group* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_GROUP* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.33 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaOpCompleteBlocking)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpCompleteBlocking event record.

An RmaOpCompleteBlocking record denotes the local completion of a blocking RMA operation.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.34 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaOpCompleteNonBlocking)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpCompleteNonBlocking event record.

An RmaOpCompleteNonBlocking record denotes the local completion of a non-blocking RMA operation.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.35 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaOpCompleteRemote)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpCompleteRemote event record.

An RmaOpCompleteRemote record denotes the remote completion of an RMA operation.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.36 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaOpTest)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpTest event record.

An RmaOpTest record denotes that a non-blocking RMA operation has been tested for completion unsuccessfully.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.37 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaPut)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)

Callback for the RmaPut event record.

An RmaPut record denotes the time a put operation was issued.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.
bytes	Bytes sent to target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.38 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)

Callback for the RmaReleaseLock event record.

An RmaReleaseLock record denotes the time the lock was released.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock released, if multiple locks are defined on a window.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.39 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaRequestLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

Callback for the RmaRequestLock event record.

An RmaRequestLock record denotes the time a lock was requested and with it the earliest time it could have been granted. It is used to mark (possibly) non-blocking lock request, as defined by the MPI standard.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.40 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaSync)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaSyncType syncType)

Callback for the RmaSync event record.

An RmaSync record denotes the direct synchronization with a possibly remote process.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
syncType	Type of synchronization.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.41 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_RmaTryLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

Callback for the RmaTryLock event record.

An RmaTryLock record denotes the time of an unsuccessful attempt to acquire the lock.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.

remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.42 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-RmaWaitChange)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

Callback for the RmaWaitChange event record.

An RmaWaitChange record denotes the change of a window that was waited for.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.43 typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_-RmaWinCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win)

Callback for the RmaWinCreate event record.

An RmaWinCreate record denotes the creation of an RMA window.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window created. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.44 typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_-RmaWinDestroy)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win)

Callback for the RmaWinDestroy event record.

An RmaWinDestroy record denotes the destruction of an RMA window.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.

userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window destructed. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.45 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-ThreadAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the ThreadAcquireLock event record.

An ThreadAcquireLock record marks that a thread acquires an lock.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
model	The threading paradigm this event took place.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.46 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-ThreadBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

Callback for the ThreadBegin event record.

Marks the begin of a thread created by another thread.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadCreate* event does have the same number.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.47 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-ThreadCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

Callback for the ThreadCreate event record.

The location created successfully a new thread.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadBegin* event does have the same number.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.48 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

Callback for the ThreadEnd event record.

Marks the end of a thread.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadWait* event does have the same number. *OTF2_UNDEFINED_UINT64* in case no corresponding *ThreadWait* event exists.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.49 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadFork)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)

Callback for the ThreadFork event record.

An ThreadFork record marks that an thread forks a thread team.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
model	The threading paradigm this event took place.
numberOfRequestedThreads	Requested size of the team.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.50 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-ThreadJoin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model)

Callback for the ThreadJoin event record.

An ThreadJoin record marks that a team of threads is joint and only the master thread continues execution.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
model	The threading paradigm this event took place.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.51 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_-ThreadReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the ThreadReleaseLock event record.

An ThreadReleaseLock record marks that a thread releases an lock.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.

userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
model	The threading paradigm this event took place.
lockID	ID of the lock.
acquisi-tionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.52 typedef **OTF2_CallbackCode**(∗ **OTF2_EvtReaderCallback_-ThreadTaskComplete**)(**OTF2_LocationRef** location, **OTF2_TimeStamp** time, uint64_t eventPosition, void ∗userData, **OTF2_AttributeList** ∗attributeList, **OTF2_CommRef** threadTeam, uint32_t creatingThread, uint32_t generationNumber)

Callback for the ThreadTaskComplete event record.

An ThreadTaskComplete record indicates that the execution of an OpenMP task has finished.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosi-tion	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_-EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
creatingTh-read	Creating thread of this task.
genera-tionNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.53 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

Callback for the ThreadTaskCreate event record.

An ThreadTaskCreate record marks that an task in was/will be created and will be processed by the specified thread team.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
creatingThread	Creating thread of this task.
generationNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.54 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

Callback for the ThreadTaskSwitch event record.

An ThreadTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
creatingThread	Creating thread of this task.
generationNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.55 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTeamBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam)

Callback for the ThreadTeamBegin event record.

The current location enters the specified thread team.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.56 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadTeamEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam)

Callback for the ThreadTeamEnd event record.

The current location leaves the specified thread team.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.57 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_ThreadWait)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

Callback for the ThreadWait event record.

The location waits for the completion of another thread.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadEnd* event does have the same number.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.2.58 typedef OTF2_CallbackCode(∗ OTF2_EvtReaderCallback_Unknown)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for an unknown event record.

Parameters

location	The location where this event happened.
time	The time when this event happened.
eventPosition	The event position of this event in the trace. Starting with 1.
userData	User data as set by *OTF2_Reader_RegisterEvtCallbacks* or *OTF2_EvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.14.3 Function Documentation

E.14.3.1 void OTF2_EvtReaderCallbacks_Clear (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*)

Clears a struct for the event callbacks.

Parameters

evtReaderCallbacks	Handle to a struct previously allocated with *OTF2_EvtReaderCallbacks_New*.

E.14.3.2 void OTF2_EvtReaderCallbacks_Delete (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*)

Deallocates a struct for the event callbacks.

Parameters

evtReaderCallbacks	Handle to a struct previously allocated with *OTF2_EvtReaderCallbacks_New*.

E.14.3.3 OTF2_EvtReaderCallbacks∗ OTF2_EvtReaderCallbacks_New (void)

Allocates a new struct for the event callbacks.

Returns

A newly allocated struct of type *OTF2_EvtReaderCallbacks*.

E.14.3.4 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetBufferFlushCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_BufferFlush *bufferFlushCallback*)

Registers the callback for the BufferFlush event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
buffer-FlushCall-back	Function which should be called for all *BufferFlush* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.5 OTF2_ErrorCode OTF2_EvtReaderCallbacks_-SetCallingContextSampleCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_CallingContextSample *callingContextSampleCallback*)

Registers the callback for the CallingContextSample event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
callingCon-textSample-Callback	Function which should be called for all *CallingContextSample* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.6 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetEnterCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks,* OTF2_EvtReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
enterCall-back	Function which should be called for all *Enter* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.7 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetLeaveCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks,* OTF2_EvtReaderCallback_Leave *leaveCallback*)

Registers the callback for the Leave event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
leaveCall-back	Function which should be called for all *Leave* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.8 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMeasurementOnOffCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_MeasurementOnOff *measurementOnOffCallback*)

Registers the callback for the MeasurementOnOff event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
measure-mentOnOff-Callback	Function which should be called for all *MeasurementOnOff* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.9 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMetricCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_Metric *metricCallback*)

Registers the callback for the Metric event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
metricCall-back	Function which should be called for all *Metric* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.10 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiCollectiveBegin *mpiCollectiveBeginCallback*)

Registers the callback for the MpiCollectiveBegin event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
mpiCollec-tiveBegin-Callback	Function which should be called for all *MpiCollectiveBegin* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.11 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_-EvtReaderCallback_MpiCollectiveEnd *mpiCollectiveEndCallback*)

Registers the callback for the MpiCollectiveEnd event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
mpiCollec-tiveEnd-Callback	Function which should be called for all *MpiCollectiveEnd* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.12 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIrecvCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiIrecv *mpiIrecvCallback*)

Registers the callback for the MpiIrecv event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
mpiIrecv-Callback	Function which should be called for all *MpiIrecv* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.13 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiIrecvRequest *mpiIrecvRequestCallback*)

Registers the callback for the MpiIrecvRequest event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
mpiIrecvRequestCallback	Function which should be called for all *MpiIrecvRequest* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.14 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIsendCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiIsend *mpiIsendCallback*)

Registers the callback for the MpiIsend event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
mpiIsendCallback	Function which should be called for all *MpiIsend* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.15 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiIsendComplete *mpiIsendCompleteCallback*)

Registers the callback for the MpiIsendComplete event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
mpiIsend-Complete-Callback	Function which should be called for all *MpiIsendComplete* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.16 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRecvCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiRecv *mpiRecvCallback*)

Registers the callback for the MpiRecv event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
mpiRecv-Callback	Function which should be called for all *MpiRecv* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.17 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRequestCancelledCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiRequestCancelled *mpiRequestCancelledCallback*)

Registers the callback for the MpiRequestCancelled event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
mpiRe-questCan-celledCall-back	Function which should be called for all *MpiRequestCancelled* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.18 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRequestTestCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiRequestTest *mpiRequestTestCallback*)

Registers the callback for the MpiRequestTest event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
mpiRe-questTest-Callback	Function which should be called for all *MpiRequestTest* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.19 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiSendCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_MpiSend *mpiSendCallback*)

Registers the callback for the MpiSend event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
mpiSend-Callback	Function which should be called for all *MpiSend* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

E.14.3.20 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_OmpAcquireLock *ompAcquireLockCallback*)

Registers the callback for the OmpAcquireLock event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
ompAcquireLockCallback	Function which should be called for all *OmpAcquireLock* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.21 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpForkCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
ompForkCallback	Function which should be called for all *OmpFork* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.22 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpJoinCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_OmpJoin *ompJoinCallback*)

Registers the callback for the OmpJoin event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
ompJoin-Callback	Function which should be called for all *OmpJoin* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.23 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_OmpReleaseLock *ompReleaseLockCallback*)

Registers the callback for the OmpReleaseLock event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
ompRe-leaseLock-Callback	Function which should be called for all *OmpReleaseLock* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.24 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_OmpTaskComplete *ompTaskCompleteCallback*)

Registers the callback for the OmpTaskComplete event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
ompTaskCompleteCallback	Function which should be called for all *OmpTaskComplete* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.25 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_OmpTaskCreate *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
omp-TaskCreate-Callback	Function which should be called for all *OmpTaskCreate* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.26 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetOmpTaskSwitchCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks,* OTF2_EvtReaderCallback_OmpTaskSwitch *ompTaskSwitchCallback*)

Registers the callback for the OmpTaskSwitch event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
omp-TaskSwitch-Callback	Function which should be called for all *OmpTaskSwitch* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.27 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetParameterIntCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_ParameterInt *parameterIntCallback*)

Registers the callback for the ParameterInt event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
parameterIntCallback	Function which should be called for all *ParameterInt* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.28 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetParameterStringCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_ParameterString *parameterStringCallback*)

Registers the callback for the ParameterString event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
parameterStringCallback	Function which should be called for all *ParameterString* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

E.14.3.29 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetParameterUnsignedIntCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_ParameterUnsignedInt *parameterUnsignedIntCallback*)

Registers the callback for the ParameterUnsignedInt event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
parameterUnsignedIntCallback	Function which should be called for all *ParameterUnsignedInt* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.30 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaAcquireLock *rmaAcquireLockCallback*)

Registers the callback for the RmaAcquireLock event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
rmaAcquireLockCallback	Function which should be called for all *RmaAcquireLock* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.31 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaAtomicCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaAtomic *rmaAtomicCallback*)

Registers the callback for the RmaAtomic event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaAtomic-Callback	Function which should be called for all *RmaAtomic* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.32 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaCollectiveBegin *rmaCollectiveBeginCallback*)

Registers the callback for the RmaCollectiveBegin event.

Parameters

evtReader-Callbacks	Struct for all callbacks.

rmaCollectiveBegin-Callback	Function which should be called for all *RmaCollectiveBegin* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.33 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaCollectiveEnd *rmaCollectiveEndCallback*)

Registers the callback for the RmaCollectiveEnd event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaCollectiveEnd-Callback	Function which should be called for all *RmaCollectiveEnd* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.34 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaGetCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaGet *rmaGetCallback*)

Registers the callback for the RmaGet event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
rmaGetCallback	Function which should be called for all *RmaGet* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.35 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaGroupSync *rmaGroupSyncCallback*)

Registers the callback for the RmaGroupSync event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
rmaGroupSyncCallback	Function which should be called for all *RmaGroupSync* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

E.14.3.36 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteBlockingCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaOpCompleteBlocking *rmaOpCompleteBlockingCallback*)

Registers the callback for the RmaOpCompleteBlocking event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaOp-Complete-Blocking-Callback	Function which should be called for all *RmaOpCompleteBlocking* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.37 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking *rmaOpCompleteNonBlockingCallback*)

Registers the callback for the RmaOpCompleteNonBlocking event.

Parameters

evtReader-Callbacks	Struct for all callbacks.

rmaOpCompleteNonBlockingCallback	Function which should be called for all *RmaOpCompleteNonBlocking* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.38 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteRemoteCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaOpCompleteRemote *rmaOpCompleteRemoteCallback*)

Registers the callback for the RmaOpCompleteRemote event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
rmaOpCompleteRemoteCallback	Function which should be called for all *RmaOpCompleteRemote* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.39 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpTestCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaOpTest *rmaOpTestCallback*)

Registers the callback for the RmaOpTest event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaOpTest-Callback	Function which should be called for all *RmaOpTest* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.40 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaPutCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaPut *rmaPutCallback*)

Registers the callback for the RmaPut event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaPut-Callback	Function which should be called for all *RmaPut* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.41 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaReleaseLockCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaReleaseLock *rmaReleaseLockCallback*)

Registers the callback for the RmaReleaseLock event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaRe-leaseLock-Callback	Function which should be called for all *RmaReleaseLock* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.42 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaRequestLockCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaRequestLock *rmaRequestLockCallback*)

Registers the callback for the RmaRequestLock event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaRe-questLock-Callback	Function which should be called for all *RmaRequestLock* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.43 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaSyncCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaSync *rmaSyncCallback*)

Registers the callback for the RmaSync event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaSync-Callback	Function which should be called for all *RmaSync* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.44 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaTryLockCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaTryLock *rmaTryLockCallback*)

Registers the callback for the RmaTryLock event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaTry-LockCall-back	Function which should be called for all *RmaTryLock* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.45 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaWaitChangeCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaWaitChange *rmaWaitChangeCallback*)

Registers the callback for the RmaWaitChange event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaWait-Change-Callback	Function which should be called for all *RmaWaitChange* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.46 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaWinCreateCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaWinCreate *rmaWinCreateCallback*)

Registers the callback for the RmaWinCreate event.

Parameters

evtReader-Callbacks	Struct for all callbacks.

rmaWin-CreateCall-back	Function which should be called for all *RmaWinCreate* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.47 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaWinDestroyCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_RmaWinDestroy *rmaWinDestroyCallback*)

Registers the callback for the RmaWinDestroy event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
rmaWinDe-stroyCall-back	Function which should be called for all *RmaWinDestroy* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.48 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadAcquireLockCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadAcquireLock *threadAcquireLockCallback*)

Registers the callback for the ThreadAcquireLock event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
threadAcquireLockCallback	Function which should be called for all *ThreadAcquireLock* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.49 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadBeginCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadBegin *threadBeginCallback*)

Registers the callback for the ThreadBegin event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
threadBeginCallback	Function which should be called for all *ThreadBegin* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.50 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadCreateCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadCreate *threadCreateCallback*)

Registers the callback for the ThreadCreate event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
threadCreateCallback	Function which should be called for all *ThreadCreate* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.51 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadEndCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadEnd *threadEndCallback*)

Registers the callback for the ThreadEnd event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
threadEnd-Callback	Function which should be called for all *ThreadEnd* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.52 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadForkCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks,* OTF2_EvtReaderCallback_ThreadFork *threadForkCallback*)

Registers the callback for the ThreadFork event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
threadFork-Callback	Function which should be called for all *ThreadFork* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.53 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadJoinCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks,* OTF2_EvtReaderCallback_ThreadJoin *threadJoinCallback*)

Registers the callback for the ThreadJoin event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
threadJoin-Callback	Function which should be called for all *ThreadJoin* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.54 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadReleaseLockCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadReleaseLock *threadReleaseLockCallback*)

Registers the callback for the ThreadReleaseLock event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
thread-Release-LockCall-back	Function which should be called for all *ThreadReleaseLock* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.55 OTF2_ErrorCode OTF2_EvtReaderCallbacks_-SetThreadTaskCompleteCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadTaskComplete *threadTaskCompleteCallback*)

Registers the callback for the ThreadTaskComplete event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
threadTaskCompleteCallback	Function which should be called for all *ThreadTaskComplete* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.56 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTaskCreateCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadTaskCreate *threadTaskCreateCallback*)

Registers the callback for the ThreadTaskCreate event.

Parameters

evtReaderCallbacks	Struct for all callbacks.
threadTaskCreateCallback	Function which should be called for all *ThreadTaskCreate* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.57 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTaskSwitchCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadTaskSwitch *threadTaskSwitchCallback*)

Registers the callback for the ThreadTaskSwitch event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
thread-TaskSwitch-Callback	Function which should be called for all *ThreadTaskSwitch* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.58 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTeamBeginCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadTeamBegin *threadTeamBeginCallback*)

Registers the callback for the ThreadTeamBegin event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
thread-TeamBegin-Callback	Function which should be called for all *ThreadTeamBegin* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.59 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadTeamEndCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadTeamEnd *threadTeamEndCallback*)

Registers the callback for the ThreadTeamEnd event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
threadTea-mEndCall-back	Function which should be called for all *ThreadTeamEnd* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.60 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetThreadWaitCallback (OTF2_EvtReaderCallbacks ∗ *evtReaderCallbacks*, OTF2_EvtReaderCallback_ThreadWait *threadWaitCallback*)

Registers the callback for the ThreadWait event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
threadWait-Callback	Function which should be called for all *ThreadWait* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14.3.61 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetUnknownCallback (OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_EvtReaderCallback_Unknown *unknownCallback*)

Registers the callback for the Unknown event.

Parameters

evtReader-Callbacks	Struct for all callbacks.
unknown-Callback	Function which should be called for all unknown events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.15 otf2/OTF2_EvtWriter.h File Reference

This lowest user-visible layer provides write routines to write event data of a single location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_AttributeList.h>
```

Typedefs

- typedef struct OTF2_EvtWriter_struct OTF2_EvtWriter

Keeps all necessary information about the event writer. See OTF2_EvtWriter_-struct for detailed information.

Functions

- OTF2_ErrorCode OTF2_EvtWriter_BufferFlush (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_TimeStamp stopTime)

  Records an BufferFlush event.
- OTF2_ErrorCode OTF2_EvtWriter_CallingContextSample (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_-CallingContextRef callingContext, uint32_t unwindDistance, OTF2_InterruptGeneratorRef interruptGenerator)

  Records an CallingContextSample event.
- OTF2_ErrorCode OTF2_EvtWriter_ClearRewindPoint (OTF2_EvtWriter ∗writer, uint32_t rewindId)

  Please give me a documantation.
- OTF2_ErrorCode OTF2_EvtWriter_Enter (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RegionRef region)

  Records an Enter event.
- OTF2_ErrorCode OTF2_EvtWriter_GetLocationID (const OTF2_EvtWriter ∗writer, OTF2_LocationRef ∗locationID)

  Function to get the location ID of a writer object.
- OTF2_ErrorCode OTF2_EvtWriter_GetNumberOfEvents (OTF2_EvtWriter ∗writer, uint64_t ∗numberOfEvents)

  Get the number of events.
- OTF2_ErrorCode OTF2_EvtWriter_GetUserData (const OTF2_EvtWriter ∗writer, void ∗∗userData)

  Function to get the location of a writer object.
- OTF2_ErrorCode OTF2_EvtWriter_Leave (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RegionRef region)

  Records an Leave event.
- OTF2_ErrorCode OTF2_EvtWriter_MeasurementOnOff (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_-MeasurementMode measurementMode)

  Records an MeasurementOnOff event.

- OTF2_ErrorCode OTF2_EvtWriter_Metric (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type ∗typeIDs, const OTF2_-MetricValue ∗metricValues)

  Records an Metric event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiCollectiveBegin (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time)

  Records an MpiCollectiveBegin event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiCollectiveEnd (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

  Records an MpiCollectiveEnd event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiIrecv (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, uint32_t sender, OTF2_-CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

  Records an MpiIrecv event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiIrecvRequest (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint64_t requestID)

  Records an MpiIrecvRequest event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiIsend (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, uint32_t receiver, OTF2_-CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

  Records an MpiIsend event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiIsendComplete (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint64_t requestID)

  Records an MpiIsendComplete event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiRecv (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, uint32_t sender, OTF2_-CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Records an MpiRecv event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiRequestCancelled (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint64_-t requestID)

  Records an MpiRequestCancelled event.
- OTF2_ErrorCode OTF2_EvtWriter_MpiRequestTest (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint64_t requestID)

Records an MpiRequestTest event.

- OTF2_ErrorCode OTF2_EvtWriter_MpiSend (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, uint32_t receiver, OTF2_-CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Records an MpiSend event.
- OTF2_ErrorCode OTF2_EvtWriter_OmpAcquireLock (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint32_t lockID, uint32_t acquisitionOrder)

  Records an OmpAcquireLock event.
- OTF2_ErrorCode OTF2_EvtWriter_OmpFork (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, uint32_t numberOfRe-questedThreads)

  Records an OmpFork event.
- OTF2_ErrorCode OTF2_EvtWriter_OmpJoin (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time)

  Records an OmpJoin event.
- OTF2_ErrorCode OTF2_EvtWriter_OmpReleaseLock (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint32_t lockID, uint32_t acquisitionOrder)

  Records an OmpReleaseLock event.
- OTF2_ErrorCode OTF2_EvtWriter_OmpTaskComplete (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint64_t taskID)

  Records an OmpTaskComplete event.
- OTF2_ErrorCode OTF2_EvtWriter_OmpTaskCreate (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint64_t taskID)

  Records an OmpTaskCreate event.
- OTF2_ErrorCode OTF2_EvtWriter_OmpTaskSwitch (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, uint64_t taskID)

  Records an OmpTaskSwitch event.
- OTF2_ErrorCode OTF2_EvtWriter_ParameterInt (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_ParameterRef parameter, int64_t value)

  Records an ParameterInt event.
- OTF2_ErrorCode OTF2_EvtWriter_ParameterString (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_ParameterRef parameter, OTF2_StringRef string)

  Records an ParameterString event.

- OTF2_ErrorCode OTF2_EvtWriter_ParameterUnsignedInt (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_-ParameterRef parameter, uint64_t value)

  Records an ParameterUnsignedInt event.
- OTF2_ErrorCode OTF2_EvtWriter_Rewind (OTF2_EvtWriter ∗writer, uint32_-t rewindId)

  Please give me a documantation.
- OTF2_ErrorCode OTF2_EvtWriter_RmaAcquireLock (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

  Records an RmaAcquireLock event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaAtomic (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_-t bytesReceived, uint64_t matchingId)

  Records an RmaAtomic event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveBegin (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time)

  Records an RmaCollectiveBegin event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveEnd (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_-t root, uint64_t bytesSent, uint64_t bytesReceived)

  Records an RmaCollectiveEnd event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaGet (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)

  Records an RmaGet event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaGroupSync (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)

  Records an RmaGroupSync event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteBlocking (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_-RmaWinRef win, uint64_t matchingId)

  Records an RmaOpCompleteBlocking event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteNonBlocking (OTF2_-EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint64_t matchingId)

  Records an RmaOpCompleteNonBlocking event.

- OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteRemote (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_-RmaWinRef win, uint64_t matchingId)

  Records an RmaOpCompleteRemote event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaOpTest (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint64_t matchingId)

  Records an RmaOpTest event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaPut (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)

  Records an RmaPut event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaReleaseLock (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)

  Records an RmaReleaseLock event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaRequestLock (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

  Records an RmaRequestLock event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaSync (OTF2_EvtWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaSyncType syncType)

  Records an RmaSync event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaTryLock (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

  Records an RmaTryLock event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaWaitChange (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win)

  Records an RmaWaitChange event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaWinCreate (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win)

  Records an RmaWinCreate event.
- OTF2_ErrorCode OTF2_EvtWriter_RmaWinDestroy (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win)

  Records an RmaWinDestroy event.

- OTF2_ErrorCode OTF2_EvtWriter_SetLocationID (OTF2_EvtWriter ∗writer, OTF2_LocationRef location)

  The location ID is not always known on measurment start, and only needed on the first buffer flush to generate the file name. This function enables setting of the location ID after generating the buffer object.

- OTF2_ErrorCode OTF2_EvtWriter_SetUserData (OTF2_EvtWriter ∗writer, void ∗userData)

  Function to set user defined data to a writer object.

- OTF2_ErrorCode OTF2_EvtWriter_StoreRewindPoint (OTF2_EvtWriter ∗writer, uint32_t rewindId)

  Please give me a documantation.

- OTF2_ErrorCode OTF2_EvtWriter_ThreadAcquireLock (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_-Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

  Records an ThreadAcquireLock event.

- OTF2_ErrorCode OTF2_EvtWriter_ThreadBegin (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CommRef threadContingent, uint64_t sequenceCount)

  Records an ThreadBegin event.

- OTF2_ErrorCode OTF2_EvtWriter_ThreadCreate (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CommRef threadContingent, uint64_t sequenceCount)

  Records an ThreadCreate event.

- OTF2_ErrorCode OTF2_EvtWriter_ThreadEnd (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CommRef threadContingent, uint64_t sequenceCount)

  Records an ThreadEnd event.

- OTF2_ErrorCode OTF2_EvtWriter_ThreadFork (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)

  Records an ThreadFork event.

- OTF2_ErrorCode OTF2_EvtWriter_ThreadJoin (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_Paradigm model)

  Records an ThreadJoin event.

- OTF2_ErrorCode OTF2_EvtWriter_ThreadReleaseLock (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_-Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

  Records an ThreadReleaseLock event.

- OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskComplete (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_-CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Records an ThreadTaskComplete event.
- OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskCreate (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Records an ThreadTaskCreate event.
- OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskSwitch (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Records an ThreadTaskSwitch event.
- OTF2_ErrorCode OTF2_EvtWriter_ThreadTeamBegin (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CommRef threadTeam)

  Records an ThreadTeamBegin event.
- OTF2_ErrorCode OTF2_EvtWriter_ThreadTeamEnd (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CommRef threadTeam)

  Records an ThreadTeamEnd event.
- OTF2_ErrorCode OTF2_EvtWriter_ThreadWait (OTF2_EvtWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp time, OTF2_CommRef threadContingent, uint64_t sequenceCount)

  Records an ThreadWait event.

E.15.1 Detailed Description

This lowest user-visible layer provides write routines to write event data of a single location.

Source Template:

templates/OTF2_EvtWriter.tmpl.h

E.15.2 Function Documentation

E.15.2.1 OTF2_ErrorCode OTF2_EvtWriter_BufferFlush (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_TimeStamp *stopTime*)

Records an BufferFlush event.

This event signals that the internal buffer was flushed at the given time.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
stopTime	The time the buffer flush finished.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.2 OTF2_ErrorCode OTF2_EvtWriter_CallingContextSample (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_CallingContextRef *callingContext,* uint32_t *unwindDistance,* OTF2_InterruptGeneratorRef *interruptGenerator*)

Records an CallingContextSample event.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
callingContext	References a *CallingContext* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_CALLING_CONTEXT* is available.
unwindDistance	The unwindContext specifies the first context whose ip(return adress) was still marked since the last sample this means that no progress was made in the repsective region The last region that was not returned from since the last sample Is one stack level higher, but may now be at at different line number OTF2_CallingContextRef unwindContext; However, instead of this we specify the distance (number of intermediate edges) between the calling context and the unwind context Note: unwindDistance=0 would mean no progress in the leaf region since the last sample which is unlikely If not available, UNDEFINED should be used.
interruptGenerator	References a *InterruptGenerator* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_INTERRUPT_GENERATOR* is available.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.3 OTF2_ErrorCode OTF2_EvtWriter_ClearRewindPoint (OTF2_EvtWriter ∗ *writer*, uint32_t *rewindId*)

Please give me a documantation.

Parameters

writer	Writer object.
rewindId	Generic attributes for the event.

Since

Version 1.1

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.4 OTF2_ErrorCode OTF2_EvtWriter_Enter (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RegionRef *region*)

Records an Enter event.

An enter record indicates that the program enters a code region.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.5 OTF2_ErrorCode OTF2_EvtWriter_GetLocationID (const OTF2_EvtWriter ∗ *writer*, OTF2_LocationRef ∗ *locationID*)

Function to get the location ID of a writer object.

Parameters

writer	Writer object which has to be deleted
locationID	Pointer to a variable where the ID is returned in

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.6 OTF2_ErrorCode OTF2_EvtWriter_GetNumberOfEvents (OTF2_EvtWriter ∗ *writer*, uint64_t ∗ *numberOfEvents*)

Get the number of events.

Get the number of events written with this event writer. You should call this function right before closing the event writer to get the correct number of stored event records.

Parameters

	writer	Writer object.
`out`	*numberOfEvents*	Return pointer to the number of events.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.7 OTF2_ErrorCode OTF2_EvtWriter_GetUserData (const OTF2_EvtWriter * *writer*, void ** *userData*)

Function to get the location of a writer object.

Parameters

	writer	Writer object.
out	*userData*	Pointer to a variable where the pointer to the location is returned in.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.8 OTF2_ErrorCode OTF2_EvtWriter_Leave (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_RegionRef *region*)

Records an Leave event.

A leave record indicates that the program leaves a code region.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.9 OTF2_ErrorCode OTF2_EvtWriter_MeasurementOnOff (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_MeasurementMode *measurementMode*)

Records an MeasurementOnOff event.

This event signals where the measurement system turned measurement on or off.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
measurementMode	Is the measurement turned on (*OTF2_MEASUREMENT_ON*) or off (*OTF2_MEASUREMENT_OFF*)?

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.10 OTF2_ErrorCode OTF2_EvtWriter_Metric (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_MetricRef *metric*, uint8_t *numberOfMetrics*, const OTF2_Type ∗ *typeIDs*, const OTF2_MetricValue ∗ *metricValues*)

Records an Metric event.

A metric event is always stored at the location that recorded the metric. A metric event can reference a metric class or metric instance. Therefore, metric classes and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
metric	Could be a metric class or a metric instance. References a *MetricClass*, or a *MetricInstance* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_METRIC* is available.
numberOfMetrics	Number of metrics with in the set.
typeIDs	List of metric types. These types must match that of the corresponding *MetricMember* definitions.
metricValues	List of metric values.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.11 OTF2_ErrorCode OTF2_EvtWriter_MpiCollectiveBegin (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time*)

Records an MpiCollectiveBegin event.

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI_-GATHER, MPI_SCATTER etc.).

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.12 OTF2_ErrorCode OTF2_EvtWriter_MpiCollectiveEnd (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_CollectiveOp *collectiveOp,* OTF2_CommRef *communicator,* uint32_t *root,* uint64_t *sizeSent,* uint64_t *sizeReceived*)

Records an MpiCollectiveEnd event.

A MpiCollectiveEnd record marks the end of an MPI collective operation (MPI_-GATHER, MPI_SCATTER etc.). It keeps the necessary information for this event: type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

Parameters

writer	Writer object.

attributeList	Generic attributes for the event.
time	The time for this event.
collectiveOp	Determines which collective operation it is.
communicator	Communicator References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
root	MPI rank of root in `communicator`.
sizeSent	Size of the sent message.
sizeReceived	Size of the received message.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.13 OTF2_ErrorCode OTF2_EvtWriter_MpiIrecv (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, uint32_t *sender*, OTF2_CommRef *communicator*, uint32_t *msgTag*, uint64_t *msgLength*, uint64_t *requestID*)

Records an MpiIrecv event.

A MpiIrecv record indicates that a MPI message was received (MPI_IRECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
sender	MPI rank of sender in `communicator`.
communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.14 OTF2_ErrorCode OTF2_EvtWriter_MpiIrecvRequest (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, uint64_t *requestID*)

Records an MpiIrecvRequest event.

Signals the request of an receive, which can be completed later.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
requestID	ID of the requested receive

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.15 OTF2_ErrorCode OTF2_EvtWriter_MpiIsend (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, uint32_t *receiver*, OTF2_CommRef *communicator*, uint32_t *msgTag*, uint64_t *msgLength*, uint64_t *requestID*)

Records an MpiIsend event.

A MpiIsend record indicates that a MPI message send process was initiated (MPI_-ISEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.16 OTF2_ErrorCode OTF2_EvtWriter_MpiIsendComplete (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, uint64_t *requestID*)

Records an MpiIsendComplete event.

Signals the completion of non-blocking send request.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.17 OTF2_ErrorCode OTF2_EvtWriter_MpiRecv (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* uint32_t *sender,* OTF2_CommRef *communicator,* uint32_t *msgTag,* uint64_t *msgLength*)

Records an MpiRecv event.

A MpiRecv record indicates that a MPI message was received (MPI_RECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
sender	MPI rank of sender in `communicator`.
communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.18 OTF2_ErrorCode OTF2_EvtWriter_MpiRequestCancelled (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* uint64_t *requestID*)

Records an MpiRequestCancelled event.

This events appears if the program canceled a request.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.19 OTF2_ErrorCode OTF2_EvtWriter_MpiRequestTest (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* uint64_t *requestID*)

Records an MpiRequestTest event.

This events appears if the program tests if a request has already completed but the test failed.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.20 OTF2_ErrorCode OTF2_EvtWriter_MpiSend (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* uint32_t *receiver,* OTF2_CommRef *communicator,* uint32_t *msgTag,* uint64_t *msgLength*)

Records an MpiSend event.

A MpiSend record indicates that a MPI message send process was initiated (MPI_-SEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
receiver	MPI rank of receiver in `communicator`.
communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.21 OTF2_ErrorCode OTF2_EvtWriter_OmpAcquireLock (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, uint32_t *lockID*, uint32_t *acquisitionOrder*)

Records an OmpAcquireLock event.

An OmpAcquireLock record marks that a thread acquires an OpenMP lock.

This event record is superseded by the *ThreadAcquireLock* event record and should not be used when the *ThreadAcquireLock* event record is in use.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.0

Deprecated

In version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.22 OTF2_ErrorCode OTF2_EvtWriter_OmpFork (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, uint32_t *numberOfRequestedThreads*)

Records an OmpFork event.

An OmpFork record marks that an OpenMP Thread forks a thread team.

This event record is superseded by the *ThreadFork* event record and should not be used when the *ThreadFork* event record is in use.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
numberOfRequestedThreads	Requested size of the team.

Since

Version 1.0

Deprecated

In version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.23 OTF2_ErrorCode OTF2_EvtWriter_OmpJoin (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*)

Records an OmpJoin event.

An OmpJoin record marks that a team of threads is joint and only the master thread continues execution.

This event record is superseded by the *ThreadJoin* event record and should not be used when the *ThreadJoin* event record is in use.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.

Since

Version 1.0

Deprecated

In version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.24 OTF2_ErrorCode OTF2_EvtWriter_OmpReleaseLock (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* uint32_t *lockID,* uint32_t *acquisitionOrder*)

Records an OmpReleaseLock event.

An OmpReleaseLock record marks that a thread releases an OpenMP lock.

This event record is superseded by the *ThreadReleaseLock* event record and should not be used when the *ThreadReleaseLock* event record is in use.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.0

Deprecated

In version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.25 OTF2_ErrorCode OTF2_EvtWriter_OmpTaskComplete (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* uint64_t *taskID*)

Records an OmpTaskComplete event.

An OmpTaskComplete record indicates that the execution of an OpenMP task has finished.

This event record is superseded by the *ThreadTaskComplete* event record and should not be used when the *ThreadTaskComplete* event record is in use.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
taskID	Identifier of the completed task instance.

Since

Version 1.0

Deprecated

In version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.26 OTF2_ErrorCode OTF2_EvtWriter_OmpTaskCreate (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* uint64_t *taskID*)

Records an OmpTaskCreate event.

An OmpTaskCreate record marks that an OpenMP Task was/will be created in the current region.

This event record is superseded by the *ThreadTaskCreate* event record and should not be used when the *ThreadTaskCreate* event record is in use.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
taskID	Identifier of the newly created task instance.

Since

Version 1.0

Deprecated

In version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.27 OTF2_ErrorCode OTF2_EvtWriter_OmpTaskSwitch (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* uint64_t *taskID*)

Records an OmpTaskSwitch event.

An OmpTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

This event record is superseded by the *ThreadTaskSwitch* event record and should not be used when the *ThreadTaskSwitch* event record is in use.

Parameters

writer	Writer object.

attributeList	Generic attributes for the event.
time	The time for this event.
taskID	Identifier of the now active task instance.

Since

Version 1.0

Deprecated

In version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.28 OTF2_ErrorCode OTF2_EvtWriter_ParameterInt (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_ParameterRef *parameter*, int64_t *value*)

Records an ParameterInt event.

A ParameterInt record marks that in the current region, the specified integer parameter has the specified value.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.29 OTF2_ErrorCode OTF2_EvtWriter_ParameterString (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_ParameterRef *parameter,* OTF2_StringRef *string*)

Records an ParameterString event.

A ParameterString record marks that in the current region, the specified string parameter has the specified value.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_PARAMETER* is available.
string	Value: Handle of a string definition References a *String* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_STRING* is available.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.30 OTF2_ErrorCode OTF2_EvtWriter_ParameterUnsignedInt (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_ParameterRef *parameter,* uint64_t *value*)

Records an ParameterUnsignedInt event.

A ParameterUnsignedInt record marks that in the current region, the specified unsigned integer parameter has the specified value.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.

parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.31 OTF2_ErrorCode OTF2_EvtWriter_Rewind (OTF2_EvtWriter ∗ *writer*, uint32_t *rewindId*)

Please give me a documantation.

Parameters

writer	Writer object.
rewindId	Generic attributes for the event.

Since

Version 1.1

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.32 OTF2_ErrorCode OTF2_EvtWriter_RmaAcquireLock (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint32_t *remote*, uint64_t *lockId*, OTF2_LockType *lockType*)

Records an RmaAcquireLock event.

An RmaAcquireLock record denotes the time a lock was acquired by the process.

Parameters

writer	Writer object.

attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.33 OTF2_ErrorCode OTF2_EvtWriter_RmaAtomic (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint32_t *remote*, OTF2_RmaAtomicType *type*, uint64_t *bytesSent*, uint64_t *bytesReceived*, uint64_t *matchingId*)

Records an RmaAtomic event.

An RmaAtomic record denotes the time a atomic operation was issued.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.
type	Type of atomic operation.
bytesSent	Bytes sent to target.
bytesReceived	Bytes received from target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.34 OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveBegin (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time*)

Records an RmaCollectiveBegin event.

An RmaCollectiveBegin record denotes the beginnig of a collective RMA operation.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.35 OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveEnd (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_CollectiveOp *collectiveOp,* OTF2_RmaSyncLevel *syncLevel,* OTF2_RmaWinRef *win,* uint32_t *root,* uint64_t *bytesSent,* uint64_t *bytesReceived*)

Records an RmaCollectiveEnd event.

An RmaCollectiveEnd record denotes the end of a collective RMA operation.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
collectiveOp	Determines which collective operation it is.
syncLevel	Synchronization level of this collective operation.

win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
root	Root process for this operation.
bytesSent	Bytes sent in operation.
bytesReceived	Bytes receives in operation.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.36 OTF2_ErrorCode OTF2_EvtWriter_RmaGet (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint32_t *remote*, uint64_t *bytes*, uint64_t *matchingId*)

Records an RmaGet event.

An RmaGet record denotes the time a get operation was issued.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.
bytes	Bytes received from target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.37 OTF2_ErrorCode OTF2_EvtWriter_RmaGroupSync (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaSyncLevel *syncLevel*, OTF2_RmaWinRef *win*, OTF2_GroupRef *group*)

Records an RmaGroupSync event.

An RmaGroupSync record denotes the synchronization with a subgroup of processes on a window.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
syncLevel	Synchronization level of this collective operation.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
group	Group of remote processes involved in synchronization. References a *Group* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_GROUP* is available.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.38 OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteBlocking (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint64_t *matchingId*)

Records an RmaOpCompleteBlocking event.

An RmaOpCompleteBlocking record denotes the local completion of a blocking RMA operation.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.

win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.39 OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteNonBlocking (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint64_t *matchingId*)

Records an RmaOpCompleteNonBlocking event.

An RmaOpCompleteNonBlocking record denotes the local completion of a non-blocking RMA operation.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.40 OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteRemote (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_RmaWinRef *win,* uint64_t *matchingId*)

Records an RmaOpCompleteRemote event.

An RmaOpCompleteRemote record denotes the remote completion of an RMA operation.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.41 OTF2_ErrorCode OTF2_EvtWriter_RmaOpTest (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_RmaWinRef *win,* uint64_t *matchingId*)

Records an RmaOpTest event.

An RmaOpTest record denotes that a non-blocking RMA operation has been tested for completion unsuccessfully.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.42 OTF2_ErrorCode OTF2_EvtWriter_RmaPut (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint32_t *remote*, uint64_t *bytes*, uint64_t *matchingId*)

Records an RmaPut event.

An RmaPut record denotes the time a put operation was issued.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.
bytes	Bytes sent to target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.43 OTF2_ErrorCode OTF2_EvtWriter_RmaReleaseLock (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint32_t *remote*, uint64_t *lockId*)

Records an RmaReleaseLock event.

An RmaReleaseLock record denotes the time the lock was released.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock released, if multiple locks are defined on a window.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.44 OTF2_ErrorCode OTF2_EvtWriter_RmaRequestLock (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint32_t *remote*, uint64_t *lockId*, OTF2_LockType *lockType*)

Records an RmaRequestLock event.

An RmaRequestLock record denotes the time a lock was requested and with it the earliest time it could have been granted. It is used to mark (possibly) non-blocking lock request, as defined by the MPI standard.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.45 OTF2_ErrorCode OTF2_EvtWriter_RmaSync (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint32_t *remote*, OTF2_RmaSyncType *syncType*)

Records an RmaSync event.

An RmaSync record denotes the direct synchronization with a possibly remote process.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
syncType	Type of synchronization.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.46 OTF2_ErrorCode OTF2_EvtWriter_RmaTryLock (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*, uint32_t *remote*, uint64_t *lockId*, OTF2_LockType *lockType*)

Records an RmaTryLock event.

An RmaTryLock record denotes the time of an unsuccessful attempt to acquire the lock.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.47 OTF2_ErrorCode OTF2_EvtWriter_RmaWaitChange (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*)

Records an RmaWaitChange event.

An RmaWaitChange record denotes the change of a window that was waited for.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.48 OTF2_ErrorCode OTF2_EvtWriter_RmaWinCreate (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*)

Records an RmaWinCreate event.

An RmaWinCreate record denotes the creation of an RMA window.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window created. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.49 OTF2_ErrorCode OTF2_EvtWriter_RmaWinDestroy (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_RmaWinRef *win*)

Records an RmaWinDestroy event.

An RmaWinDestroy record denotes the destruction of an RMA window.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
win	ID of the window destructed. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.50 OTF2_ErrorCode OTF2_EvtWriter_SetLocationID (OTF2_EvtWriter ∗ *writer*, OTF2_LocationRef *location*)

The location ID is not always known on measurment start, and only needed on the first buffer flush to generate the file name. This function enables setting of the location ID after generating the buffer object.

Parameters

writer	Writer object.
location	Location ID.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.51 OTF2_ErrorCode OTF2_EvtWriter_SetUserData (OTF2_EvtWriter ∗ *writer*, void ∗ *userData*)

Function to set user defined data to a writer object.

Parameters

writer	Writer object.
userData	User provided data. Can be queried with *OTF2_EvtWriter_-GetUserData*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.52 OTF2_ErrorCode OTF2_EvtWriter_StoreRewindPoint (OTF2_EvtWriter ∗ *writer*, uint32_t *rewindId*)

Please give me a documantation.

Parameters

writer	Writer object.
rewindId	Generic attributes for the event.

Since

Version 1.1

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.53 OTF2_ErrorCode OTF2_EvtWriter_ThreadAcquireLock (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_Paradigm *model*, uint32_t *lockID*, uint32_t *acquisitionOrder*)

Records an ThreadAcquireLock event.

An ThreadAcquireLock record marks that a thread acquires an lock.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
model	The threading paradigm this event took place.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.54 OTF2_ErrorCode OTF2_EvtWriter_ThreadBegin (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_CommRef *threadContingent*, uint64_t *sequenceCount*)

Records an ThreadBegin event.

Marks the begin of a thread created by another thread.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadCreate* event does have the same number.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.55 OTF2_ErrorCode OTF2_EvtWriter_ThreadCreate (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_CommRef *threadContingent*, uint64_t *sequenceCount*)

Records an ThreadCreate event.

The location created successfully a new thread.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadBegin* event does have the same number.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.56 OTF2_ErrorCode OTF2_EvtWriter_ThreadEnd (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_CommRef *threadContingent,* uint64_t *sequenceCount*)

Records an ThreadEnd event.

Marks the end of a thread.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadWait* event does have the same number. *OTF2_UNDEFINED_UINT64* in case no corresponding *ThreadWait* event exists.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.57 OTF2_ErrorCode OTF2_EvtWriter_ThreadFork (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_Paradigm *model,* uint32_t *numberOfRequestedThreads*)

Records an ThreadFork event.

An ThreadFork record marks that an thread forks a thread team.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
model	The threading paradigm this event took place.
numberOfRequestedThreads	Requested size of the team.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.58 OTF2_ErrorCode OTF2_EvtWriter_ThreadJoin (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_Paradigm *model*)

Records an ThreadJoin event.

An ThreadJoin record marks that a team of threads is joint and only the master thread continues execution.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
model	The threading paradigm this event took place.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.59 OTF2_ErrorCode OTF2_EvtWriter_ThreadReleaseLock (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_Paradigm *model,* uint32_t *lockID,* uint32_t *acquisitionOrder*)

Records an ThreadReleaseLock event.

An ThreadReleaseLock record marks that a thread releases an lock.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.

model	The threading paradigm this event took place.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.60 OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskComplete (OTF2_EvtWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *time,* OTF2_CommRef *threadTeam,* uint32_t *creatingThread,* uint32_t *generationNumber*)

Records an ThreadTaskComplete event.

An ThreadTaskComplete record indicates that the execution of an OpenMP task has finished.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
creatingThread	Creating thread of this task.
generationNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.61 OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskCreate (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_CommRef *threadTeam*, uint32_t *creatingThread*, uint32_t *generationNumber*)

Records an ThreadTaskCreate event.

An ThreadTaskCreate record marks that an task in was/will be created and will be processed by the specified thread team.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
creatingThread	Creating thread of this task.
generationNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.62 OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskSwitch (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_CommRef *threadTeam*, uint32_t *creatingThread*, uint32_t *generationNumber*)

Records an ThreadTaskSwitch event.

An ThreadTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

Parameters

writer	Writer object.

attributeList	Generic attributes for the event.
time	The time for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
creatingThread	Creating thread of this task.
generationNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.63 OTF2_ErrorCode OTF2_EvtWriter_ThreadTeamBegin (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_CommRef *threadTeam*)

Records an ThreadTeamBegin event.

The current location enters the specified thread team.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.64 OTF2_ErrorCode OTF2_EvtWriter_ThreadTeamEnd (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_CommRef *threadTeam*)

Records an ThreadTeamEnd event.

The current location leaves the specified thread team.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.15.2.65 OTF2_ErrorCode OTF2_EvtWriter_ThreadWait (OTF2_EvtWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *time*, OTF2_CommRef *threadContingent*, uint64_t *sequenceCount*)

Records an ThreadWait event.

The location waits for the completion of another thread.

Parameters

writer	Writer object.
attributeList	Generic attributes for the event.
time	The time for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadEnd* event does have the same number.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.16 otf2/OTF2_GeneralDefinitions.h File Reference

This header file provides general definitions which should be accessible in all internal and external modules.

`#include <stdint.h>`

`#include <otf2/OTF2_ErrorCodes.h>`

Defines

- #define OTF2_CHUNK_SIZE_MAX (uint64_t)(1024 $*$ 1024 $*$ 16)

  Defines the maximum size of a chunk.
- #define OTF2_CHUNK_SIZE_MIN (uint64_t)(256 $*$ 1024)

  Defines the minimum size of a chunk.
- #define OTF2_UNDEFINED_ATTRIBUTE ((OTF2_AttributeRef)OTF2_-UNDEFINED_UINT32)

  The invalid value for a reference to a Attribute definition.
- #define OTF2_UNDEFINED_CALLING_CONTEXT ((OTF2_CallingContextRef)OTF2_UNDEFINED_UINT32)

  The invalid value for a reference to a CallingContext definition.
- #define OTF2_UNDEFINED_CALLPATH ((OTF2_CallpathRef)OTF2_-UNDEFINED_UINT32)

  The invalid value for a reference to a Callpath definition.
- #define OTF2_UNDEFINED_CALLSITE ((OTF2_CallsiteRef)OTF2_-UNDEFINED_UINT32)

  The invalid value for a reference to a Callsite definition.
- #define OTF2_UNDEFINED_CART_DIMENSION ((OTF2_CartDimensionRef)OTF2_UNDEFINED_UINT32)

  The invalid value for a reference to a CartDimension definition.
- #define OTF2_UNDEFINED_CART_TOPOLOGY ((OTF2_CartTopologyRef)OTF2_UNDEFINED_UINT32)

  The invalid value for a reference to a CartTopology definition.

- #define OTF2_UNDEFINED_COMM ((OTF2_CommRef)OTF2_UNDEFINED_-UINT32)

  The invalid value for a reference to a Comm definition.
- #define OTF2_UNDEFINED_GROUP ((OTF2_GroupRef)OTF2_UNDEFINED_-UINT32)

  The invalid value for a reference to a Group definition.
- #define OTF2_UNDEFINED_INTERRUPT_GENERATOR ((OTF2_InterruptGeneratorRef)OTF2_UNDEFINED_UINT32)

  The invalid value for a reference to a InterruptGenerator definition.
- #define OTF2_UNDEFINED_LOCATION ((OTF2_LocationRef)OTF2_-UNDEFINED_UINT64)

  The invalid value for a reference to a Location definition.
- #define OTF2_UNDEFINED_LOCATION_GROUP ((OTF2_LocationGroupRef)OTF2_UNDEFINED_UINT32)

  The invalid value for a reference to a LocationGroup definition.
- #define OTF2_UNDEFINED_METRIC ((OTF2_MetricRef)OTF2_UNDEFINED_-UINT32)

  The invalid value for a reference to a MetricClass, or a MetricInstance definition.
- #define OTF2_UNDEFINED_METRIC_MEMBER ((OTF2_MetricMemberRef)OTF2_UNDEFINED_UINT32)

  The invalid value for a reference to a MetricMember definition.
- #define OTF2_UNDEFINED_PARAMETER ((OTF2_ParameterRef)OTF2_-UNDEFINED_UINT32)

  The invalid value for a reference to a Parameter definition.
- #define OTF2_UNDEFINED_REGION ((OTF2_RegionRef)OTF2_UNDEFINED_-UINT32)

  The invalid value for a reference to a Region definition.
- #define OTF2_UNDEFINED_RMA_WIN ((OTF2_RmaWinRef)OTF2_-UNDEFINED_UINT32)

  The invalid value for a reference to a RmaWin definition.
- #define OTF2_UNDEFINED_SOURCE_CODE_LOCATION ((OTF2_SourceCodeLocationRef)OTF2_UNDEFINED_UINT32)

  The invalid value for a reference to a SourceCodeLocation definition.
- #define OTF2_UNDEFINED_STRING ((OTF2_StringRef)OTF2_UNDEFINED_-UINT32)

  The invalid value for a reference to a String definition.
- #define OTF2_UNDEFINED_SYSTEM_TREE_NODE ((OTF2_SystemTreeNodeRef)OTF2_UNDEFINED_UINT32)

  The invalid value for a reference to a SystemTreeNode definition.

- #define OTF2_UNDEFINED_TIMESTAMP OTF2_UNDEFINED_UINT64

  Undefined value for OTF2_TimeStamp.
- #define OTF2_UNDEFINED_TYPE OTF2_UNDEFINED_UINT8

  Undefined value for enums.

OTF2 library version.

- #define OTF2_VERSION_MAJOR 1

  Major version number of this OTF2 version.
- #define OTF2_VERSION_MINOR 5

  Minor version number of this OTF2 version.
- #define OTF2_VERSION_BUGFIX 1

  Bugfix version number of this OTF2 version.
- #define OTF2_VERSION_SUFFIX ""

  Any string suffix of this OTF2 version.
- #define OTF2_VERSION "1.5.1"

  The OTF2 version as string.

Standard undefined values for basic data types.

- #define OTF2_UNDEFINED_UINT8 ((uint8_t)(~((uint8_t)0u)))

  Undefined value for type uint8_t.
- #define OTF2_UNDEFINED_UINT16 ((uint16_t)(~((uint16_t)0u)))

  Undefined value for type uint16_t.
- #define OTF2_UNDEFINED_UINT32 ((uint32_t)(~((uint32_t)0u)))

  Undefined value for type uint32_t.
- #define OTF2_UNDEFINED_UINT64 ((uint64_t)(~((uint64_t)0u)))

  Undefined value for type uint64_t.

Typedefs

- typedef uint32_t OTF2_AttributeRef

  Type used to indicate a reference to a Attribute definition.
- typedef uint8_t OTF2_Boolean

  Wrapper for enum OTF2_Boolean_enum.
- typedef uint32_t OTF2_CallingContextRef

  Type used to indicate a reference to a CallingContext definition.

- typedef uint32_t OTF2_CallpathRef

  Type used to indicate a reference to a Callpath definition.
- typedef uint32_t OTF2_CallsiteRef

  Type used to indicate a reference to a Callsite definition.
- typedef uint32_t OTF2_CartDimensionRef

  Type used to indicate a reference to a CartDimension definition.
- typedef uint32_t OTF2_CartTopologyRef

  Type used to indicate a reference to a CartTopology definition.
- typedef uint32_t OTF2_CommRef

  Type used to indicate a reference to a Comm definition.
- typedef uint8_t OTF2_Compression

  Defines which compression is used. Please see OTF2_Compression_enum for a detailed description.
- typedef struct OTF2_DefReader_struct OTF2_DefReader

  OTF2 local definition reader handle.
- typedef struct OTF2_EvtReader_struct OTF2_EvtReader

  OTF2 local event reader handle.
- typedef uint8_t OTF2_FileMode

  Defines how to interact with files. Please see OTF2_FileMode_enum for a detailed description.
- typedef uint8_t OTF2_FileSubstrate

  Wrapper for enum OTF2_FileSubstrate_enum.
- typedef uint8_t OTF2_FileType

  Wrapper for enum OTF2_FileType_enum.
- typedef uint8_t OTF2_FlushType

  Defines whether the recorded data is flushed to a file or not. Please see OTF2_-Flushtype_enum for a detailed description.
- typedef struct OTF2_GlobalDefReader_struct OTF2_GlobalDefReader

  OTF2 global definition reader handle.
- typedef struct OTF2_GlobalEvtReader_struct OTF2_GlobalEvtReader

  OTF2 global event reader handle.
- typedef struct OTF2_GlobalSnapReader_struct OTF2_GlobalSnapReader

  OTF2 global snap reader handle.
- typedef uint32_t OTF2_GroupRef

  Type used to indicate a reference to a Group definition.
- typedef uint8_t OTF2_Hint

  Wrapper for enum OTF2_Hint_enum.
- typedef uint32_t OTF2_InterruptGeneratorRef

  Type used to indicate a reference to a InterruptGenerator definition.

- typedef uint32_t OTF2_LocationGroupRef

  Type used to indicate a reference to a LocationGroup definition.
- typedef uint64_t OTF2_LocationRef

  Type used to indicate a reference to a Location definition.
- typedef uint8_t OTF2_MappingType

  Wrapper for enum OTF2_MappingType_enum.
- typedef struct OTF2_MarkerReader_struct OTF2_MarkerReader

  OTF2 marker reader handle.
- typedef uint32_t OTF2_MetricMemberRef

  Type used to indicate a reference to a MetricMember definition.
- typedef uint32_t OTF2_MetricRef

  Type used to indicate a reference to a MetricClass, or a MetricInstance definition.
- typedef uint8_t OTF2_Paradigm

  Wrapper for enum OTF2_Paradigm_enum.
- typedef uint8_t OTF2_ParadigmClass

  Wrapper for enum OTF2_ParadigmClass_enum.
- typedef uint8_t OTF2_ParadigmProperty

  Wrapper for enum OTF2_ParadigmProperty_enum.
- typedef uint32_t OTF2_ParameterRef

  Type used to indicate a reference to a Parameter definition.
- typedef uint32_t OTF2_RegionRef

  Type used to indicate a reference to a Region definition.
- typedef uint32_t OTF2_RmaWinRef

  Type used to indicate a reference to a RmaWin definition.
- typedef struct OTF2_SnapReader_struct OTF2_SnapReader

  OTF2 local snap reader handle.
- typedef uint32_t OTF2_SourceCodeLocationRef

  Type used to indicate a reference to a SourceCodeLocation definition.
- typedef uint32_t OTF2_StringRef

  Type used to indicate a reference to a String definition.
- typedef uint32_t OTF2_SystemTreeNodeRef

  Type used to indicate a reference to a SystemTreeNode definition.
- typedef uint8_t OTF2_ThumbnailType

  Wrapper for enum OTF2_ThumbnailType_enum.
- typedef uint64_t OTF2_TimeStamp

  OTF2 time stamp.
- typedef uint8_t OTF2_Type

  Wrapper for enum OTF2_Type_enum.

Enumerations

- enum OTF2_Boolean_enum {
  OTF2_FALSE = 0,
  OTF2_TRUE = !OTF2_FALSE }

  A boolean.
- enum OTF2_CallbackCode {
  OTF2_CALLBACK_SUCCESS = 0,
  OTF2_CALLBACK_INTERRUPT = !OTF2_CALLBACK_SUCCESS,
  OTF2_CALLBACK_ERROR = !OTF2_CALLBACK_SUCCESS }

  Return value to indicate that the record reading should be interrupted.
- enum OTF2_Compression_enum {
  OTF2_COMPRESSION_UNDEFINED = 0,
  OTF2_COMPRESSION_NONE = 1,
  OTF2_COMPRESSION_ZLIB = 2 }

  Defines which compression is used.
- enum OTF2_FileMode_enum {
  OTF2_FILEMODE_WRITE = 0,
  OTF2_FILEMODE_READ = 1,
  OTF2_FILEMODE_MODIFY = 2 }

  Defines how to interact with files.
- enum OTF2_FileSubstrate_enum {
  OTF2_SUBSTRATE_UNDEFINED = 0,
  OTF2_SUBSTRATE_POSIX = 1,
  OTF2_SUBSTRATE_SION = 2,
  OTF2_SUBSTRATE_NONE = 3 }

  Defines which file substrate is used.
- enum OTF2_FileType_enum {
  OTF2_FILETYPE_ANCHOR = 0,
  OTF2_FILETYPE_GLOBAL_DEFS = 1,
  OTF2_FILETYPE_LOCAL_DEFS = 2,
  OTF2_FILETYPE_EVENTS = 3,
  OTF2_FILETYPE_SNAPSHOTS = 4,
  OTF2_FILETYPE_THUMBNAIL = 5,
  OTF2_FILETYPE_MARKER = 6,
  OTF2_FILETYPE_SIONRANKMAP = 7 }

Defines which file type is used.

- enum OTF2_FlushType_enum {

  OTF2_NO_FLUSH = 0,

  OTF2_FLUSH = 1 }

  Defines whether the recorded data is flushed to a file or not.
- enum OTF2_Hint_enum { OTF2_HINT_GLOBAL_READER = 0 }

  List of possible hints.
- enum OTF2_MappingType_enum {

  OTF2_MAPPING_STRING = 0,

  OTF2_MAPPING_ATTRIBUTE = 1,

  OTF2_MAPPING_LOCATION = 2,

  OTF2_MAPPING_REGION = 3,

  OTF2_MAPPING_GROUP = 4,

  OTF2_MAPPING_METRIC = 5,

  OTF2_MAPPING_COMM = 6,

  OTF2_MAPPING_PARAMETER = 7,

  OTF2_MAPPING_RMA_WIN = 8,

  OTF2_MAPPING_SOURCE_CODE_LOCATION = 9,

  OTF2_MAPPING_CALLING_CONTEXT = 10,

  OTF2_MAPPING_INTERRUPT_GENERATOR = 11,

  OTF2_MAPPING_MAX = 12 }

  Possible mappings from local to global identifiers.
- enum OTF2_Paradigm_enum {

  OTF2_PARADIGM_UNKNOWN = 0,

  OTF2_PARADIGM_USER = 1,

  OTF2_PARADIGM_COMPILER = 2,

  OTF2_PARADIGM_OPENMP = 3,

  OTF2_PARADIGM_MPI = 4,

  OTF2_PARADIGM_CUDA = 5,

  OTF2_PARADIGM_MEASUREMENT_SYSTEM = 6,

  OTF2_PARADIGM_PTHREAD = 7,

  OTF2_PARADIGM_HMPP = 8,

  OTF2_PARADIGM_OMPSS = 9,

  OTF2_PARADIGM_HARDWARE = 10,

OTF2_PARADIGM_GASPI = 11,
OTF2_PARADIGM_UPC = 12,
OTF2_PARADIGM_SHMEM = 13,
OTF2_PARADIGM_WINTHREAD = 14,
OTF2_PARADIGM_QTTHREAD = 15,
OTF2_PARADIGM_ACETHREAD = 16,
OTF2_PARADIGM_TBBTHREAD = 17,
OTF2_PARADIGM_OPENACC = 18,
OTF2_PARADIGM_OPENCL = 19,
OTF2_PARADIGM_MTAPI = 20,
OTF2_PARADIGM_SAMPLING = 21 }

List of known paradigms. Parallel paradigms have their expected paradigm class and known paradigm properties attached.

- enum OTF2_ParadigmClass_enum {
  OTF2_PARADIGM_CLASS_PROCESS = 0,
  OTF2_PARADIGM_CLASS_THREAD_FORK_JOIN = 1,
  OTF2_PARADIGM_CLASS_THREAD_CREATE_WAIT = 2,
  OTF2_PARADIGM_CLASS_ACCELERATOR = 3 }

  List of paradigm classes.
- enum OTF2_ParadigmProperty_enum {
  OTF2_PARADIGM_PROPERTY_COMM_NAME_TEMPLATE = 0,
  OTF2_PARADIGM_PROPERTY_RMA_WIN_NAME_TEMPLATE = 1,
  OTF2_PARADIGM_PROPERTY_RMA_ONLY = 2 }

  List of paradigm properties.
- enum OTF2_ThumbnailType_enum {
  OTF2_THUMBNAIL_TYPE_REGION = 0,
  OTF2_THUMBNAIL_TYPE_METRIC = 1,
  OTF2_THUMBNAIL_TYPE_ATTRIBUTES = 2 }

  Type of definitions used as metric in an thumbnail.
- enum OTF2_Type_enum {
  OTF2_TYPE_NONE = 0,
  OTF2_TYPE_UINT8 = 1,
  OTF2_TYPE_UINT16 = 2,
  OTF2_TYPE_UINT32 = 3,
  OTF2_TYPE_UINT64 = 4,

OTF2_TYPE_INT8 = 5,
OTF2_TYPE_INT16 = 6,
OTF2_TYPE_INT32 = 7,
OTF2_TYPE_INT64 = 8,
OTF2_TYPE_FLOAT = 9,
OTF2_TYPE_DOUBLE = 10,
OTF2_TYPE_STRING = 11,
OTF2_TYPE_ATTRIBUTE = 12,
OTF2_TYPE_LOCATION = 13,
OTF2_TYPE_REGION = 14,
OTF2_TYPE_GROUP = 15,
OTF2_TYPE_METRIC = 16,
OTF2_TYPE_COMM = 17,
OTF2_TYPE_PARAMETER = 18,
OTF2_TYPE_RMA_WIN = 19,
OTF2_TYPE_SOURCE_CODE_LOCATION = 20,
OTF2_TYPE_CALLING_CONTEXT = 21,
OTF2_TYPE_INTERRUPT_GENERATOR = 22 }

OTF2 basic data types.

E.16.1 Detailed Description

This header file provides general definitions which should be accessible in all internal and external modules.

Source Template:

templates/OTF2_GeneralDefinitions.tmpl.h

E.16.2 Enumeration Type Documentation

E.16.2.1 enum OTF2_Boolean_enum

A boolean.

Since

Version 1.5

Enumerator:

OTF2_FALSE False.

OTF2_TRUE True.

E.16.2.2 enum OTF2_CallbackCode

Return value to indicate that the record reading should be interrupted.

Returning *OTF2_CALLBACK_INTERRUPT* will stop reading more events, if functions like:

- *OTF2_Reader_ReadLocalEvents*
- *OTF2_Reader_ReadAllLocalEvents*
- *OTF2_Reader_ReadLocalEventsBackward*
- *OTF2_Reader_ReadGlobalEvents*
- *OTF2_Reader_ReadAllGlobalEvents*
- *OTF2_Reader_ReadLocalDefinitions*
- *OTF2_Reader_ReadAllLocalDefinitions*
- *OTF2_Reader_ReadGlobalDefinitions*
- *OTF2_Reader_ReadAllGlobalDefinitions*

where called. The return value for these functions is *OTF2_ERROR_INTERRUPTED_-BY_CALLBACK* in this case. It is valid to call any reader functions in such a condition again.

This type is also used as return type in the collective and locking callbacks (see Operating OTF2 in an collective context and Operating OTF2 in a multi-threads context). Any value different than *OTF2_CALLBACK_SUCCESS* is treated as an error and the calling function will return *OTF2_ERROR_COLLECTIVE_CALLBACK* or *OTF2_ERROR_LOCKING_CALLBACK* to its caller, respectively. As the name *OTF2_CALLBACK_INTERRUPT* does not really fit in this context, the alias *OTF2_-CALLBACK_ERROR* is provided for these callbacks.

Enumerator:

OTF2_CALLBACK_SUCCESS Record reading can continue.

OTF2_CALLBACK_INTERRUPT Interrupt record reading. Control returns to the caller of the read function with error OTF2_ERROR_INTERRUPTED_-BY_CALLBACK to signal this. The actual value can be any except OTF2_CALLBACK_SUCCESS.

OTF2_CALLBACK_ERROR Signaling an error in the callback.

E.16.2.3 enum OTF2_Compression_enum

Defines which compression is used.

Enumerator:

OTF2_COMPRESSION_UNDEFINED Undefined.

OTF2_COMPRESSION_NONE No compression is used.

OTF2_COMPRESSION_ZLIB Use zlib compression.

E.16.2.4 enum OTF2_FileMode_enum

Defines how to interact with files.

Enumerator:

OTF2_FILEMODE_WRITE Open a file in write-only mode.

OTF2_FILEMODE_READ Open a file in read-only mode.

OTF2_FILEMODE_MODIFY Open a file in write-read mode.

E.16.2.5 enum OTF2_FileSubstrate_enum

Defines which file substrate is used.

Since

Version 1.0

Enumerator:

OTF2_SUBSTRATE_UNDEFINED Undefined.

OTF2_SUBSTRATE_POSIX Use standard posix file interface.

OTF2_SUBSTRATE_SION Use the interface of the SIONlib to write many logical files into few physical files.

OTF2_SUBSTRATE_NONE Do not use any file interface. No data is written to a file.

E.16.2.6 enum OTF2_FileType_enum

Defines which file type is used.

Since

Version 1.0

Enumerator:

OTF2_FILETYPE_ANCHOR Represents the type for the anchor file (.otf2).

OTF2_FILETYPE_GLOBAL_DEFS Represents the type for the global definition file (.def).

OTF2_FILETYPE_LOCAL_DEFS Represents the type for a local definition file (.def).

OTF2_FILETYPE_EVENTS Represents the type for a event file (.evt).

OTF2_FILETYPE_SNAPSHOTS Represents the type for a snapshot file (.snap).

OTF2_FILETYPE_THUMBNAIL Represents the type for a thumb file (.thumb).

OTF2_FILETYPE_MARKER Represents the type for a marker file (.marker).

OTF2_FILETYPE_SIONRANKMAP Internal file which holds the SION rank map (.srm).

E.16.2.7 enum OTF2_FlushType_enum

Defines whether the recorded data is flushed to a file or not.

Enumerator:

OTF2_NO_FLUSH Flushing will be supressed when running out of memory.

OTF2_FLUSH Recorded data is flushed when running out of memory.

E.16.2.8 enum OTF2_Hint_enum

List of possible hints.

Since

Version 1.5

Enumerator:

OTF2_HINT_GLOBAL_READER Hint the reader that the user will use the global reader to read per-location data (e.g., event and snapshot data).

In case of the SIONlib substrate that means the SION handles of the per-location local reader are not duplicated and thus not thread safe.

Datatype *OTF2_Boolean* with default value *OTF2_FALSE*.

This is for an *OTF2_Archive* only valid if the file mode equals to *OTF2_-FILEMODE_READ*.

The hint will be locked when opening any of the per-location data files.

E.16.2.9 enum OTF2_MappingType_enum

Possible mappings from local to global identifiers.

Since

Version 1.0

Enumerator:

OTF2_MAPPING_STRING Mapping of *String* identifiers.

OTF2_MAPPING_ATTRIBUTE Mapping of *Attribute* identifiers.

OTF2_MAPPING_LOCATION Mapping of *Location* identifiers.

OTF2_MAPPING_REGION Mapping of *Region* identifiers.

OTF2_MAPPING_GROUP Mapping of *Group* identifiers.

OTF2_MAPPING_METRIC Mapping of *Metric* identifiers.

OTF2_MAPPING_COMM Mapping of *Comm* identifiers.

OTF2_MAPPING_PARAMETER Mapping of *Parameter* identifiers.

OTF2_MAPPING_RMA_WIN Mapping of *RmaWin* identifiers.

Since

Version 1.2.

OTF2_MAPPING_SOURCE_CODE_LOCATION Mapping of *SourceCode-Location* identifiers.

Since

Version 1.5.

OTF2_MAPPING_CALLING_CONTEXT Mapping of *CallingContext* identifiers.
Since

Version 1.5.

OTF2_MAPPING_INTERRUPT_GENERATOR Mapping of *InterruptGenerator* identifiers.
Since

Version 1.5.

OTF2_MAPPING_MAX Max entry.

E.16.2.10 enum OTF2_Paradigm_enum

List of known paradigms. Parallel paradigms have their expected paradigm class and known paradigm properties attached.

Since

Version 1.1

Enumerator:

OTF2_PARADIGM_UNKNOWN An unknown paradigm.

OTF2_PARADIGM_USER User instrumentation.

OTF2_PARADIGM_COMPILER Compiler instrumentation.

OTF2_PARADIGM_OPENMP OpenMP.
Paradigm Class:

OTF2_PARADIGM_CLASS_THREAD_FORK_JOIN

OTF2_PARADIGM_MPI MPI.
Paradigm Class:

OTF2_PARADIGM_CLASS_PROCESS

OTF2_PARADIGM_CUDA CUDA.
Paradigm Class:

OTF2_PARADIGM_CLASS_ACCELERATOR

OTF2_PARADIGM_MEASUREMENT_SYSTEM The measurement software.
Since

Version 1.2.

OTF2_PARADIGM_PTHREAD POSIX threads.

Paradigm Class:

OTF2_PARADIGM_CLASS_THREAD_CREATE_WAIT

Since

Version 1.3.

OTF2_PARADIGM_HMPP HMPP.

Paradigm Class:

OTF2_PARADIGM_CLASS_ACCELERATOR

Since

Version 1.3.

OTF2_PARADIGM_OMPSS OmpSs.

Paradigm Class:

OTF2_PARADIGM_CLASS_THREAD_FORK_JOIN

Since

Version 1.3.

OTF2_PARADIGM_HARDWARE Hardware.

Since

Version 1.3.

OTF2_PARADIGM_GASPI GASPI.

Paradigm Class:

OTF2_PARADIGM_CLASS_PROCESS

Since

Version 1.4.

OTF2_PARADIGM_UPC Unified Parallel C (UPC).

Paradigm Class:

OTF2_PARADIGM_CLASS_PROCESS

Since

Version 1.4.

OTF2_PARADIGM_SHMEM SGI SHMEM, Cray SHMEM, OpenSHMEM.

Paradigm Class:

OTF2_PARADIGM_CLASS_PROCESS

Paradigm Property:

OTF2_PARADIGM_PROPERTY_RMA_ONLY *OTF2_TRUE*

Since

Version 1.4.

OTF2_PARADIGM_WINTHREAD Windows threads.

Paradigm Class:

OTF2_PARADIGM_CLASS_THREAD_CREATE_WAIT

Since

Version 1.5.

OTF2_PARADIGM_QTTHREAD Qt threads.

Paradigm Class:

OTF2_PARADIGM_CLASS_THREAD_CREATE_WAIT

Since

Version 1.5.

OTF2_PARADIGM_ACETHREAD ACE threads.

Paradigm Class:

OTF2_PARADIGM_CLASS_THREAD_CREATE_WAIT

Since

Version 1.5.

OTF2_PARADIGM_TBBTHREAD TBB threads.

Paradigm Class:

OTF2_PARADIGM_CLASS_THREAD_FORK_JOIN

Since

Version 1.5.

OTF2_PARADIGM_OPENACC OpenACC directives.

Paradigm Class:

OTF2_PARADIGM_CLASS_ACCELERATOR

Since

Version 1.5.

OTF2_PARADIGM_OPENCL OpenCL API functions and kernels.

Paradigm Class:

OTF2_PARADIGM_CLASS_ACCELERATOR

Since

Version 1.5.

OTF2_PARADIGM_MTAPI Multicore Task API functions.
Paradigm Class:

OTF2_PARADIGM_CLASS_THREAD_FORK_JOIN

Since

Version 1.5.

OTF2_PARADIGM_SAMPLING Functions recorded by sampling.
Since

Version 1.5.

E.16.2.11 enum OTF2_ParadigmClass_enum

List of paradigm classes.

Since

Version 1.5

Enumerator:

OTF2_PARADIGM_CLASS_PROCESS A communication paradigm across multiple processes.

OTF2_PARADIGM_CLASS_THREAD_FORK_JOIN A threading paradigm which uses the fork/join model.

OTF2_PARADIGM_CLASS_THREAD_CREATE_WAIT A threading paradigm which uses the create/wait model.

OTF2_PARADIGM_CLASS_ACCELERATOR A paradigm which uses external accelerators to offload computation.

E.16.2.12 enum OTF2_ParadigmProperty_enum

List of paradigm properties.

Since

Version 1.5

Enumerator:

OTF2_PARADIGM_PROPERTY_COMM_NAME_TEMPLATE Template for unnamed *Comm* definitions. A unique name can be derived by replacing '${id}' with a unique id. Type: *String*

OTF2_PARADIGM_PROPERTY_RMA_WIN_NAME_TEMPLATE Template for unnamed *RmaWin* definitions. A unique name can be derived by replacing '${id}' with a unique id. Type: *String*

OTF2_PARADIGM_PROPERTY_RMA_ONLY Attests that this parallel paradigm only uses *RmaWin* definitions. The *Comm* definitions exists only for compliance and won't be referenced in event records. Type: *OTF2_-Boolean*

E.16.2.13 enum OTF2_ThumbnailType_enum

Type of definitions used as metric in an thumbnail.

Since

Version 1.2

Enumerator:

OTF2_THUMBNAIL_TYPE_REGION The referenced definitions are of type *Region*.

OTF2_THUMBNAIL_TYPE_METRIC The referenced definitions are of type *MetricMember*.

OTF2_THUMBNAIL_TYPE_ATTRIBUTES The referenced definitions are of type *Attribute*.

E.16.2.14 enum OTF2_Type_enum

OTF2 basic data types.

Since

Version 1.0

Enumerator:

OTF2_TYPE_NONE Undefined type. Type category: None

OTF2_TYPE_UINT8 Unsigned 8-bit integer. Type category: Integer

OTF2_TYPE_UINT16 Unsigned 16-bit integer. Type category: Integer

OTF2_TYPE_UINT32 Unsigned 32-bit integer. Type category: Integer

OTF2_TYPE_UINT64 Unsigned 64-bit integer. Type category: Integer

OTF2_TYPE_INT8 Signed 8-bit integer. Type category: Integer

OTF2_TYPE_INT16 Signed 16-bit integer. Type category: Integer

OTF2_TYPE_INT32 Signed 32-bit integer. Type category: Integer

OTF2_TYPE_INT64 Signed 64-bit integer. Type category: Integer

OTF2_TYPE_FLOAT 32-bit floating point value Type category: Floating point

OTF2_TYPE_DOUBLE 64-bit floating point value Type category: Floating point

OTF2_TYPE_STRING Mapping of *String* identifiers. Type category: Definition reference

OTF2_TYPE_ATTRIBUTE Mapping of *Attribute* identifiers. Type category: Definition reference

OTF2_TYPE_LOCATION Mapping of *Location* identifiers. Type category: Definition reference

OTF2_TYPE_REGION Mapping of *Region* identifiers. Type category: Definition reference

OTF2_TYPE_GROUP Mapping of *Group* identifiers. Type category: Definition reference

OTF2_TYPE_METRIC Mapping of *Metric* identifiers. Type category: Definition reference

OTF2_TYPE_COMM Mapping of *Comm* identifiers. Type category: Definition reference

OTF2_TYPE_PARAMETER Mapping of *Parameter* identifiers. Type category: Definition reference

OTF2_TYPE_RMA_WIN Mapping of *RmaWin* identifiers.

Since

Version 1.2.

Type category: Definition reference

OTF2_TYPE_SOURCE_CODE_LOCATION Mapping of *SourceCodeLocation* identifiers.

Since

Version 1.5.

Type category: Definition reference

OTF2_TYPE_CALLING_CONTEXT Mapping of *CallingContext* identifiers.

Since

Version 1.5.

Type category: Definition reference

OTF2_TYPE_INTERRUPT_GENERATOR Mapping of *InterruptGenerator* identifiers.

Since

Version 1.5.

Type category: Definition reference

E.17 otf2/OTF2_GlobalDefReader.h File Reference

This is the definition reader.

```
#include <stddef.h>
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_GlobalDefReaderCallbacks.h>
```

Functions

- OTF2_ErrorCode OTF2_GlobalDefReader_ReadDefinitions (OTF2_GlobalDefReader ∗reader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  Reads the given number of records from the global definition reader.

- OTF2_ErrorCode OTF2_GlobalDefReader_SetCallbacks (OTF2_GlobalDefReader ∗reader, const OTF2_GlobalDefReaderCallbacks ∗callbacks, void ∗userData)

  Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

E.17.1 Detailed Description

This is the definition reader.

E.17.2 Function Documentation

E.17.2.1 OTF2_ErrorCode OTF2_GlobalDefReader_ReadDefinitions (OTF2_GlobalDefReader ∗ *reader*, uint64_t *recordsToRead*, uint64_t ∗ *recordsRead*)

Reads the given number of records from the global definition reader.

Parameters

	reader	The records of this reader will be read when the function is issued.
	record-sToRead	This variable tells the reader how much records it has to read.
`out`	*record-sRead*	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given record-sToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finnished his job by checking recordsRead $<$ recordsToRead.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.17.2.2 OTF2_ErrorCode OTF2_GlobalDefReader_SetCallbacks (OTF2_-GlobalDefReader ∗ *reader*, const OTF2_GlobalDefReaderCallbacks ∗ *callbacks*, void ∗ *userData*)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

reader	This given reader object will be setted up with new callback functions.
callbacks	Struct which holds a function pointer for each record type. *OTF2_-GlobalDefReaderCallbacks_New*.
userData	Data passed as argument *userData* to the record callbacks.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.18 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

This defines the callbacks for the global definition reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeValue.h>
#include <otf2/OTF2_Definitions.h>
```

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Attribute)(void ∗userData, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_-StringRef description, OTF2_Type type)

  Function pointer definition for the callback which is triggered by a Attribute definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_CallingContext)(void ∗userData, OTF2_CallingContextRef self, uint64_t ip, OTF2_RegionRef region, uint32_t offsetLineNumber, OTF2_CallingContextRef parent)

  Function pointer definition for the callback which is triggered by a CallingContext definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Callpath)(void ∗userData, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_-RegionRef region)

  Function pointer definition for the callback which is triggered by a Callpath definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Callsite)(void ∗userData, OTF2_CallsiteRef self, OTF2_StringRef sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion, OTF2_RegionRef leftRegion)

  Function pointer definition for the callback which is triggered by a Callsite definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_CartCoordinate)(void ∗userData, OTF2_CartTopologyRef cartTopology, uint32_t rank, uint8_-t numberOfDimensions, const uint32_t ∗coordinates)

  Function pointer definition for the callback which is triggered by a CartCoordinate definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_CartDimension)(void ∗userData, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_-t size, OTF2_CartPeriodicity cartPeriodicity)

Function pointer definition for the callback which is triggered by a CartDimension definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_CartTopology)(void ∗userData, OTF2_CartTopologyRef self, OTF2_StringRef name, OTF2_-CommRef communicator, uint8_t numberOfDimensions, const OTF2_CartDimensionRef ∗cartDimensions)

  Function pointer definition for the callback which is triggered by a CartTopology definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_ClockProperties)(void ∗userData, uint64_t timerResolution, uint64_t globalOffset, uint64_t traceLength)

  Function pointer definition for the callback which is triggered by a ClockProperties definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Comm)(void ∗userData, OTF2_CommRef self, OTF2_StringRef name, OTF2_GroupRef group, OTF2_CommRef parent)

  Function pointer definition for the callback which is triggered by a Comm definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Group)(void ∗userData, OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_-t numberOfMembers, const uint64_t ∗members)

  Function pointer definition for the callback which is triggered by a Group definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_InterruptGenerator)(void ∗userData, OTF2_InterruptGeneratorRef self, OTF2_StringRef name, OTF2_StringRef unit, uint64_t period)

  Function pointer definition for the callback which is triggered by a InterruptGenerator definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Location)(void ∗userData, OTF2_LocationRef self, OTF2_StringRef name, OTF2_-LocationType locationType, uint64_t numberOfEvents, OTF2_LocationGroupRef locationGroup)

  Function pointer definition for the callback which is triggered by a Location definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_LocationGroup)(void ∗userData, OTF2_LocationGroupRef self, OTF2_StringRef name, OTF2_-LocationGroupType locationGroupType, OTF2_SystemTreeNodeRef systemTreeParent)

  Function pointer definition for the callback which is triggered by a LocationGroup definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_LocationGroupProperty)(void ∗userData, OTF2_LocationGroupRef locationGroup, OTF2_StringRef name, OTF2_StringRef value)

  Function pointer definition for the callback which is triggered by a LocationGroupProperty definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_LocationProperty)(void ∗userData, OTF2_LocationRef location, OTF2_StringRef name, OTF2_-StringRef value)

  Function pointer definition for the callback which is triggered by a LocationProperty definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_MetricClass)(void ∗userData, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_MetricMemberRef ∗metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)

  Function pointer definition for the callback which is triggered by a MetricClass definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_MetricClassRecorder)(void ∗userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)

  Function pointer definition for the callback which is triggered by a MetricClass-Recorder definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_MetricInstance)(void ∗userData, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_-LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)

  Function pointer definition for the callback which is triggered by a MetricInstance definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_MetricMember)(void ∗userData, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_-StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_-t exponent, OTF2_StringRef unit)

  Function pointer definition for the callback which is triggered by a MetricMember definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Paradigm)(void ∗userData, OTF2_Paradigm paradigm, OTF2_StringRef name, OTF2_-ParadigmClass paradigmClass)

  Function pointer definition for the callback which is triggered by a Paradigm definition record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_ParadigmProperty)(void ∗userData, OTF2_Paradigm paradigm, OTF2_ParadigmProperty property, OTF2_Type type, OTF2_AttributeValue attributeValue)

  Function pointer definition for the callback which is triggered by a ParadigmProperty definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Parameter)(void ∗userData, OTF2_ParameterRef self, OTF2_StringRef name, OTF2_-ParameterType parameterType)

  Function pointer definition for the callback which is triggered by a Parameter definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Region)(void ∗userData, OTF2_RegionRef self, OTF2_StringRef name, OTF2_StringRef canonicalName, OTF2_StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)

  Function pointer definition for the callback which is triggered by a Region definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_RmaWin)(void ∗userData, OTF2_RmaWinRef self, OTF2_StringRef name, OTF2_-CommRef comm)

  Function pointer definition for the callback which is triggered by a RmaWin definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_SourceCodeLocation)(void ∗userData, OTF2_SourceCodeLocationRef self, OTF2_StringRef file, uint32_t lineNumber)

  Function pointer definition for the callback which is triggered by a SourceCode-Location definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_String)(void ∗userData, OTF2_StringRef self, const char ∗string)

  Function pointer definition for the callback which is triggered by a String definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_SystemTreeNode)(void ∗userData, OTF2_SystemTreeNodeRef self, OTF2_StringRef name, OTF2_StringRef className, OTF2_SystemTreeNodeRef parent)

  Function pointer definition for the callback which is triggered by a SystemTreeN-ode definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_SystemTreeNodeDomain)(void ∗userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_SystemTreeDomain systemTreeDomain)

  Function pointer definition for the callback which is triggered by a SystemTreeN-odeDomain definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_SystemTreeNodeProperty)(void ∗userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_StringRef name, OTF2_StringRef value)

  Function pointer definition for the callback which is triggered by a SystemTreeN-odeProperty definition record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Unknown)(void ∗userData)

Function pointer definition for the callback which is triggered by an unknown definition record.

- typedef struct OTF2_GlobalDefReaderCallbacks_struct OTF2_GlobalDefReaderCallbacks

  Opaque struct which holdes all global definition record callbacks.

Functions

- void OTF2_GlobalDefReaderCallbacks_Clear (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks)

  Clears a struct for the global definition callbacks.
- void OTF2_GlobalDefReaderCallbacks_Delete (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks)

  Deallocates a struct for the global definition callbacks.
- OTF2_GlobalDefReaderCallbacks ∗ OTF2_GlobalDefReaderCallbacks_New (void)

  Allocates a new struct for the global definition callbacks.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetAttributeCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Attribute attributeCallback)

  Registers the callback for the Attribute definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCallingContextCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-CallingContext callingContextCallback)

  Registers the callback for the CallingContext definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCallpathCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Callpath callpathCallback)

  Registers the callback for the Callpath definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCallsiteCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Callsite callsiteCallback)

  Registers the callback for the Callsite definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCartCoordinateCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-CartCoordinate cartCoordinateCallback)

  Registers the callback for the CartCoordinate definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCartDimensionCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-CartDimension cartDimensionCallback)

Registers the callback for the CartDimension definition.

- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCartTopologyCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-CartTopology cartTopologyCallback)

  Registers the callback for the CartTopology definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetClockPropertiesCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-ClockProperties clockPropertiesCallback)

  Registers the callback for the ClockProperties definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCommCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Comm commCallback)

  Registers the callback for the Comm definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetGroupCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Group groupCallback)

  Registers the callback for the Group definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetInterruptGeneratorCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-InterruptGenerator interruptGeneratorCallback)

  Registers the callback for the InterruptGenerator definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetLocationCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Location locationCallback)

  Registers the callback for the Location definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetLocationGroupCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-LocationGroup locationGroupCallback)

  Registers the callback for the LocationGroup definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetLocationGroupPropertyCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-LocationGroupProperty locationGroupPropertyCallback)

  Registers the callback for the LocationGroupProperty definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetLocationPropertyCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-LocationProperty locationPropertyCallback)

  Registers the callback for the LocationProperty definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetMetricClassCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-MetricClass metricClassCallback)

  Registers the callback for the MetricClass definition.

- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetMetricClassRecorderCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-MetricClassRecorder metricClassRecorderCallback)

  Registers the callback for the MetricClassRecorder definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetMetricInstanceCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-MetricInstance metricInstanceCallback)

  Registers the callback for the MetricInstance definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetMetricMemberCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-MetricMember metricMemberCallback)

  Registers the callback for the MetricMember definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetParadigmCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Paradigm paradigmCallback)

  Registers the callback for the Paradigm definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetParadigmPropertyCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-ParadigmProperty paradigmPropertyCallback)

  Registers the callback for the ParadigmProperty definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetParameterCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Parameter parameterCallback)

  Registers the callback for the Parameter definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetRegionCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Region regionCallback)

  Registers the callback for the Region definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetRmaWinCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-RmaWin rmaWinCallback)

  Registers the callback for the RmaWin definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSourceCodeLocationCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-SourceCodeLocation sourceCodeLocationCallback)

  Registers the callback for the SourceCodeLocation definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetStringCallback (OTF2_-GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-String stringCallback)

  Registers the callback for the String definition.

- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-SystemTreeNode systemTreeNodeCallback)

  Registers the callback for the SystemTreeNode definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeDomainCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-SystemTreeNodeDomain systemTreeNodeDomainCallback)

  Registers the callback for the SystemTreeNodeDomain definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodePropertyCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-SystemTreeNodeProperty systemTreeNodePropertyCallback)

  Registers the callback for the SystemTreeNodeProperty definition.
- OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetUnknownCallback (OTF2_GlobalDefReaderCallbacks ∗globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-Unknown unknownCallback)

  Registers the callback for an unknown definition.

E.18.1 Detailed Description

This defines the callbacks for the global definition reader.

Source Template:

templates/OTF2_GlobalDefReaderCallbacks.tmpl.h

E.18.2 Typedef Documentation

E.18.2.1 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-Attribute)(void ∗userData, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_Type type)

Function pointer definition for the callback which is triggered by a *Attribute* definition record.

The attribute definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *Attribute* definition.
name	Name of the attribute. References a *String* definition.

description	Description of the attribute. References a *String* definition. Since version 1.4.
type	Type of the attribute value.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.2 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-CallingContext)(void ∗userData, OTF2_CallingContextRef self, uint64_t ip, OTF2_RegionRef region, uint32_t offsetLineNumber, OTF2_CallingContextRef parent)

Function pointer definition for the callback which is triggered by a *CallingContext* definition record.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *CallingContext* definition.
ip	Instruction pointer as the offset to the start of the function.
region	The region. References a *Region* definition.
offsetLineNumber	The line offset inside the region.
parent	Parent id of this context. References a *CallingContext* definition.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.3 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-Callpath)(void ∗userData, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region)

Function pointer definition for the callback which is triggered by a *Callpath* definition record.

The callpath definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *Callpath* definition.
parent	The parent of this callpath. References a *Callpath* definition.
region	The region of this callpath. References a *Region* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.4 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-Callsite)(void ∗userData, OTF2_CallsiteRef self, OTF2_StringRef sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion, OTF2_RegionRef leftRegion)

Function pointer definition for the callback which is triggered by a *Callsite* definition record.

The callsite definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *Callsite* definition.
sourceFile	The source file where this call was made. References a *String* definition.
lineNumber	Line number in the source file where this call was made.
enteredRegion	The region which was called. References a *Region* definition.
leftRegion	The region which made the call. References a *Region* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.5 typedef OTF2_CallbackCode(* OTF2_GlobalDefReaderCallback_CartCoordinate)(void *userData, OTF2_CartTopologyRef cartTopology, uint32_t rank, uint8_t numberOfDimensions, const uint32_t *coordinates)

Function pointer definition for the callback which is triggered by a *CartCoordinate* definition record.

Defines the coordinate of the location referenced by the given rank (w.r.t. the communicator associated to the topology) in the referenced topology.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
cartTopology	Parent *CartTopology* definition to which this one is a supplementary definition. References a *CartTopology* definition.
rank	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
numberOfDimensions	Number of dimensions.
coordinates	Coordinates, indexed by dimension.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.6 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-CartDimension)(void ∗userData, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_t size, OTF2_CartPeriodicity cartPeriodicity)

Function pointer definition for the callback which is triggered by a *CartDimension* definition record.

Each dimension in a Cartesian topology is composed of a global id, a name, its size, and whether it is periodic or not.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *CartDimension* definition.
name	The name of the cartesian topology dimension. References a *String* definition.
size	The size of the cartesian topology dimension.
cartPeriodicity	Periodicity of the cartesian topology dimension.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.7 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-CartTopology)(void ∗userData, OTF2_CartTopologyRef self, OTF2_StringRef name, OTF2_CommRef communicator, uint8_t numberOfDimensions, const OTF2_CartDimensionRef ∗cartDimensions)

Function pointer definition for the callback which is triggered by a *CartTopology* definition record.

Each topology is described by a global id, a reference to its name, a reference to a communicator, the number of dimensions, and references to those dimensions. The topology type is defined by the paradigm of the group referenced by the associated communicator.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *CartTopology* definition.
name	The name of the topology. References a *String* definition.
communicator	Communicator object used to create the topology. References a *Comm* definition.
numberOfDimensions	Number of dimensions.
cartDimensions	The dimensions of this topology. References a *CartDimension* definition.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.8 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_ClockProperties)(void ∗userData, uint64_t timerResolution, uint64_t globalOffset, uint64_t traceLength)

Function pointer definition for the callback which is triggered by a *ClockProperties* definition record.

Defines the timer resolution and time range of this trace. There will be no event with a timestamp less than `globalOffset`, and no event with timestamp greater than (`globalOffset` + `traceLength`).

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
timerResolution	Ticks per seconds.
globalOffset	A timestamp smaller than all event timestamps.
traceLength	A timespan which includes the timespan between the smallest and greatest timestamp of all event timestamps.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.9 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Comm)(void ∗userData, OTF2_CommRef self, OTF2_StringRef name, OTF2_GroupRef group, OTF2_CommRef parent)

Function pointer definition for the callback which is triggered by a *Comm* definition record.

The communicator definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *Comm* definition.
name	The name given by calling MPI_Comm_set_name on this communicator. Or the empty name to indicate that no name was given. References a *String* definition.
group	The describing MPI group of this MPI communicator The group needs to be of type *OTF2_GROUP_TYPE_COMM_GROUP* or *OTF2_GROUP_TYPE_COMM_SELF*. References a *Group* definition.
parent	The parent MPI communicator from which this communicator was created, if any. Use *OTF2_UNDEFINED_COMM* to indicate no parent. References a *Comm* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.10 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Group)(void ∗userData, OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_t numberOfMembers, const uint64_t ∗members)

Function pointer definition for the callback which is triggered by a *Group* definition record.

The group definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *Group* definition.
name	Name of this group References a *String* definition.
groupType	The type of this group. Since version 1.2.
paradigm	The paradigm of this communication group. Since version 1.2.
groupFlags	Flags for this group. Since version 1.2.
numberOfMembers	The number of members in this group.
members	The identifiers of the group members.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.11 typedef OTF2_CallbackCode($*$ OTF2_GlobalDefReaderCallback_InterruptGenerator)(void $*$userData, OTF2_InterruptGeneratorRef self, OTF2_StringRef name, OTF2_StringRef unit, uint64_t period)

Function pointer definition for the callback which is triggered by a *InterruptGenerator* definition record.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *InterruptGenerator* definition.
name	The name of this interrupt generator. References a *String* definition.
unit	The unit used by this interrupt generator for the period. References a *String* definition.
period	The period this interrupt generator generates interrupts.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.12 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Location)(void ∗userData, OTF2_LocationRef self, OTF2_StringRef name, OTF2_LocationType locationType, uint64_t numberOfEvents, OTF2_LocationGroupRef locationGroup)

Function pointer definition for the callback which is triggered by a *Location* definition record.

The location definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *Location* definition.
name	Name of the location References a *String* definition.
location-Type	Location type.
numberO-fEvents	Number of events this location has recorded.
location-Group	Location group which includes this location. References a *Location-Group* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.13 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_LocationGroup)(void ∗userData, OTF2_LocationGroupRef self, OTF2_StringRef name, OTF2_LocationGroupType locationGroupType, OTF2_SystemTreeNodeRef systemTreeParent)

Function pointer definition for the callback which is triggered by a *LocationGroup* definition record.

The location group definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *LocationGroup* definition.
name	Name of the group. References a *String* definition.
location-GroupType	Type of this group.
sys-temTreePar-ent	Parent of this location group in the system tree. References a *SystemTreeNode* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.14 typedef OTF2_CallbackCode($*$ OTF2_GlobalDefReaderCallback_-LocationGroupProperty)(void $*$userData, OTF2_LocationGroupRef locationGroup, OTF2_StringRef name, OTF2_StringRef value)

Function pointer definition for the callback which is triggered by a *LocationGroup-Property* definition record.

An arbitrary key/value property for a *LocationGroup* definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
location-Group	Parent *LocationGroup* definition to which this one is a supplementary definition. References a *LocationGroup* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.15 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-LocationProperty)(void ∗userData, OTF2_LocationRef location, OTF2_StringRef name, OTF2_StringRef value)

Function pointer definition for the callback which is triggered by a *LocationProperty* definition record.

An arbitrary key/value property for a *Location* definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
location	Parent *Location* definition to which this one is a supplementary definition. References a *Location* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.16 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-MetricClass)(void ∗userData, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_MetricMemberRef ∗metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)

Function pointer definition for the callback which is triggered by a *MetricClass* definition record.

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *MetricClass* definition.
numberOf-Metrics	Number of metrics within the set.

metricMembers	List of metric members. References a *MetricMember* definition.
metricOccurrence	Defines occurrence of a metric set.
recorderKind	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.17 typedef OTF2_CallbackCode($*$ OTF2_GlobalDefReaderCallback_MetricClassRecorder)(void $*$userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)

Function pointer definition for the callback which is triggered by a *MetricClassRecorder* definition record.

The metric class recorder definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
metricClass	Parent *MetricClass* definition to which this one is a supplementary definition. References a *MetricClass* definition.
recorder	The location which recorded the referenced metric class. References a *Location* definition.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.18 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-MetricInstance)(void ∗userData, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)

Function pointer definition for the callback which is triggered by a *MetricInstance* definition record.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type *OTF2_METRIC_ASYNCHRONOUS*.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *MetricClass* definition.
metricClass	The instanced *MetricClass*. This metric class must be of kind *OTF2_-RECORDER_KIND_ABSTRACT*. References a *MetricClass* definition.
recorder	Recorder of the metric: location ID. References a *Location* definition.
metric-Scope	Defines type of scope: location, location group, system tree node, or a generic group of locations.
scope	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.19 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-MetricMember)(void ∗userData, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2_StringRef unit)

Function pointer definition for the callback which is triggered by a *MetricMember* definition record.

A metric is defined by a metric member definition. A metric member is always a member of a metric class. Therefore, a single metric is a special case of a metric class with only one member. It is not allowed to reference a metric member id in a metric event, but only metric class IDs.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *MetricMember* definition.
name	Name of the metric. References a *String* definition.
description	Description of the metric. References a *String* definition.
metricType	Metric type: PAPI, etc.
metricMode	Metric mode: accumulative, fix, relative, etc.
valueType	Type of the value. Only *OTF2_TYPE_INT64*, *OTF2_TYPE_UINT64*, and *OTF2_TYPE_DOUBLE* are valid types. If this metric member is recorded in an *Metric* event, than this type and the type in the event must match.
metricBase	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
exponent	The values inside the Metric events should be scaled by the factor base^exponent, to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be *OTF2_BASE_-BINARY* and the exponent 10. Than the writer does not need to scale the values up to bytes, but can directly write the KiBi values into the Metric event. At reading time, the reader can apply the scaling factor to get the value in its base unit, ie. in bytes.
unit	Unit of the metric. This needs to be the scale free base unit, ie. "bytes", "operations", or "seconds". In particular this unit should not have any scale prefix. References a *String* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.20 typedef OTF2_CallbackCode(* OTF2_GlobalDefReaderCallback_-Paradigm)(void *userData, OTF2_Paradigm paradigm, OTF2_StringRef name, OTF2_ParadigmClass paradigmClass)

Function pointer definition for the callback which is triggered by a *Paradigm* definition record.

Attests that the following parallel paradigm was available at the time when the trace was recorded, and vice versa. Note that this does not attest that the paradigm was used. For convenience, this also includes a proper name for the paradigm and a classification. This definition is only allowed to appear at most once in the definitions per *Paradigm*.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
paradigm	The paradigm to attest.
name	The name of the paradigm. References a *String* definition.
paradigm-Class	The class of this paradigm.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.21 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-ParadigmProperty)(void ∗userData, OTF2_Paradigm paradigm, OTF2_ParadigmProperty property, OTF2_Type type, OTF2_AttributeValue attributeValue)

Function pointer definition for the callback which is triggered by a *ParadigmProperty* definition record.

Extensible annotation for the *Paradigm* definition.

The tuple (*paradigm*, *property*) must be unique.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
paradigm	The paradigm to annotate.
property	The property.
type	The type of this property. Must match with the defined type of the *property*.
attribute-Value	The value of this property.value

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.22 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-Parameter)(void ∗userData, OTF2_ParameterRef self, OTF2_StringRef name, OTF2_ParameterType parameterType)

Function pointer definition for the callback which is triggered by a *Parameter* definition record.

The parameter definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *Parameter* definition.
name	Name of the parameter (variable name etc.) References a *String* definition.
parameter-Type	Type of the parameter, *OTF2_ParameterType* for possible types.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.23 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-Region)(void ∗userData, OTF2_RegionRef self, OTF2_StringRef name, OTF2_StringRef canonicalName, OTF2_StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)

Function pointer definition for the callback which is triggered by a *Region* definition record.

The region definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *Region* definition.
name	Name of the region (demangled name if available). References a *String* definition.
canonical-Name	Alternative name of the region (e.g. mangled name). References a *String* definition. Since version 1.1.
description	A more detailed description of this region. References a *String* definition.
regionRole	Region role. Since version 1.1.
paradigm	Paradigm. Since version 1.1.
regionFlags	Region flags. Since version 1.1.
sourceFile	The source file where this region was declared. References a *String* definition.
beginLi-neNumber	Starting line number of this region in the source file.
endLi-neNumber	Ending line number of this region in the source file.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.24 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-RmaWin)(void ∗userData, OTF2_RmaWinRef self, OTF2_StringRef name, OTF2_CommRef comm)

Function pointer definition for the callback which is triggered by a *RmaWin* definition record.

A window defines the communication context for any remote-memory access operation.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *RmaWin* definition.
name	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a *String* definition.

comm	Communicator object used to create the window. References a *Comm* definition.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.25 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-SourceCodeLocation)(void ∗userData, OTF2_SourceCodeLocationRef self, OTF2_StringRef file, uint32_t lineNumber)

Function pointer definition for the callback which is triggered by a *SourceCodeLocation* definition record.

The definition of a source code location as tuple of the corresponding file name and line number.

When used to attach source code annotations to events, use the *OTF2_AttributeList* with a *Attribute* definition named `"SOURCE_CODE_LOCATION"` and typed *OTF2_-TYPE_SOURCE_CODE_LOCATION*.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *SourceCodeLocation* definition.
file	The name of the file for the source code location. References a *String* definition.
lineNumber	The line number for the source code location.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.26 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-String)(void ∗userData, OTF2_StringRef self, const char ∗string)

Function pointer definition for the callback which is triggered by a *String* definition record.

The string definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *String* definition.
string	The string, null terminated.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.27 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-SystemTreeNode)(void ∗userData, OTF2_SystemTreeNodeRef self, OTF2_StringRef name, OTF2_StringRef className, OTF2_SystemTreeNodeRef parent)

Function pointer definition for the callback which is triggered by a *SystemTreeNode* definition record.

The system tree node definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
self	The unique identifier for this *SystemTreeNode* definition.
name	Free form instance name of this node. References a *String* definition.
className	Free form class name of this node References a *String* definition.
parent	Parent id of this node. May be *OTF2_UNDEFINED_SYSTEM_TREE_-NODE* to indicate that there is no parent. References a *SystemTreeNode* definition.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.28 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-SystemTreeNodeDomain)(void ∗userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_SystemTreeDomain systemTreeDomain)

Function pointer definition for the callback which is triggered by a *SystemTreeNodeDomain* definition record.

The system tree node domain definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
sys-temTreeN-ode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
systemTree-Domain	The domain in which the referenced *SystemTreeNode* operates in.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.29 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_-SystemTreeNodeProperty)(void ∗userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_StringRef name, OTF2_StringRef value)

Function pointer definition for the callback which is triggered by a *SystemTreeNodeProperty* definition record.

An arbitrary key/value property for a *SystemTreeNode* definition.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.
systemTreeNode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.2.30 typedef OTF2_CallbackCode(∗ OTF2_GlobalDefReaderCallback_Unknown)(void ∗userData)

Function pointer definition for the callback which is triggered by an unknown definition record.

Parameters

userData	User data as set by *OTF2_Reader_RegisterGlobalDefCallbacks* or *OTF2_GlobalDefReader_SetCallbacks*.

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.18.3 Function Documentation

E.18.3.1 void OTF2_GlobalDefReaderCallbacks_Clear (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*)

Clears a struct for the global definition callbacks.

Parameters

globalDefReaderCallbacks	Handle to a struct previously allocated with *OTF2_GlobalDefReaderCallbacks_New*.

E.18.3.2 void OTF2_GlobalDefReaderCallbacks_Delete (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*)

Deallocates a struct for the global definition callbacks.

Parameters

globalDefReaderCallbacks	Handle to a struct previously allocated with *OTF2_GlobalDefReaderCallbacks_New*.

E.18.3.3 OTF2_GlobalDefReaderCallbacks∗ OTF2_GlobalDefReaderCallbacks_New (void)

Allocates a new struct for the global definition callbacks.

Returns

A newly allocated struct of type *OTF2_GlobalDefReaderCallbacks*.

E.18.3.4 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetAttributeCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks,* OTF2_GlobalDefReaderCallback_Attribute *attributeCallback*)

Registers the callback for the *Attribute* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
attributeCallback	Function which should be called for all *Attribute* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

E.18.3.5 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCallingContextCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_CallingContext *callingContextCallback*)

Registers the callback for the *CallingContext* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
callingContextCallback	Function which should be called for all *CallingContext* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.6 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCallpathCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Callpath *callpathCallback*)

Registers the callback for the *Callpath* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
callpathCallback	Function which should be called for all *Callpath* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.7 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCallsiteCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Callsite *callsiteCallback*)

Registers the callback for the *Callsite* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
callsite-Callback	Function which should be called for all *Callsite* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.8 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_-SetCartCoordinateCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_-CartCoordinate *cartCoordinateCallback*)

Registers the callback for the *CartCoordinate* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
cartCoordi-nateCall-back	Function which should be called for all *CartCoordinate* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.9 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_-SetCartDimensionCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_-CartDimension *cartDimensionCallback*)

Registers the callback for the *CartDimension* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
cartDimen-sionCall-back	Function which should be called for all *CartDimension* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.10 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCartTopologyCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_CartTopology *cartTopologyCallback*)

Registers the callback for the *CartTopology* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
cartTopolo-gyCallback	Function which should be called for all *CartTopology* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.11 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetClockPropertiesCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_ClockProperties *clockPropertiesCallback*)

Registers the callback for the *ClockProperties* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
clockProp-ertiesCall-back	Function which should be called for all *ClockProperties* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.12 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCommCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Comm *commCallback*)

Registers the callback for the *Comm* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
commCallback	Function which should be called for all *Comm* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.13 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetGroupCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Group *groupCallback*)

Registers the callback for the *Group* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
groupCallback	Function which should be called for all *Group* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.14 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetInterruptGeneratorCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_InterruptGenerator *interruptGeneratorCallback*)

Registers the callback for the *InterruptGenerator* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
interrupt-Generator-Callback	Function which should be called for all *InterruptGenerator* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.15 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetLocationCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Location *locationCallback*)

Registers the callback for the *Location* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
location-Callback	Function which should be called for all *Location* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.16 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_-SetLocationGroupCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_-LocationGroup *locationGroupCallback*)

Registers the callback for the *LocationGroup* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
location-GroupCall-back	Function which should be called for all *LocationGroup* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.17 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetLocationGroupPropertyCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_LocationGroupProperty *locationGroupPropertyCallback*)

Registers the callback for the *LocationGroupProperty* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
locationGroupPropertyCallback	Function which should be called for all *LocationGroupProperty* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.18 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetLocationPropertyCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_LocationProperty *locationPropertyCallback*)

Registers the callback for the *LocationProperty* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
locationPropertyCallback	Function which should be called for all *LocationProperty* definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.19 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetMetricClassCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_MetricClass *metricClassCallback*)

Registers the callback for the *MetricClass* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
metricClassCallback	Function which should be called for all *MetricClass* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.20 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetMetricClassRecorderCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_MetricClassRecorder *metricClassRecorderCallback*)

Registers the callback for the *MetricClassRecorder* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
metricClassRecorderCallback	Function which should be called for all *MetricClassRecorder* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.21 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetMetricInstanceCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_MetricInstance *metricInstanceCallback*)

Registers the callback for the *MetricInstance* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
metricInstanceCallback	Function which should be called for all *MetricInstance* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.22 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetMetricMemberCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_MetricMember *metricMemberCallback*)

Registers the callback for the *MetricMember* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
metricMem-berCallback	Function which should be called for all *MetricMember* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.23 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetParadigmCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Paradigm *paradigmCallback*)

Registers the callback for the *Paradigm* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
paradigm-Callback	Function which should be called for all *Paradigm* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.24 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetParadigmPropertyCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_ParadigmProperty *paradigmPropertyCallback*)

Registers the callback for the *ParadigmProperty* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
paradigmPropertyCallback	Function which should be called for all *ParadigmProperty* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.25 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetParameterCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Parameter *parameterCallback*)

Registers the callback for the *Parameter* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
parameterCallback	Function which should be called for all *Parameter* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.26 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetRegionCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Region *regionCallback*)

Registers the callback for the *Region* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
regionCallback	Function which should be called for all *Region* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.27 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetRmaWinCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_RmaWin *rmaWinCallback*)

Registers the callback for the *RmaWin* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.

rmaWin-Callback	Function which should be called for all *RmaWin* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.28 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSourceCodeLocationCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks,* OTF2_GlobalDefReaderCallback_SourceCodeLocation *sourceCodeLocationCallback*)

Registers the callback for the *SourceCodeLocation* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
source-CodeLoca-tionCall-back	Function which should be called for all *SourceCodeLocation* definitions.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.29 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetStringCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_String *stringCallback*)

Registers the callback for the *String* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
stringCall-back	Function which should be called for all *String* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.30 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_-SetSystemTreeNodeCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_-SystemTreeNode *systemTreeNodeCallback*)

Registers the callback for the *SystemTreeNode* definition.

Parameters

globalDef-Reader-Callbacks	Struct for all callbacks.
sys-temTreeN-odeCall-back	Function which should be called for all *SystemTreeNode* definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.31 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeDomainCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_SystemTreeNodeDomain *systemTreeNodeDomainCallback*)

Registers the callback for the *SystemTreeNodeDomain* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
systemTreeNodeDomainCallback	Function which should be called for all *SystemTreeNodeDomain* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.32 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodePropertyCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_SystemTreeNodeProperty *systemTreeNodePropertyCallback*)

Registers the callback for the *SystemTreeNodeProperty* definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
systemTreeNodePropertyCallback	Function which should be called for all *SystemTreeNodeProperty* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.18.3.33 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetUnknownCallback (OTF2_GlobalDefReaderCallbacks ∗ *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Unknown *unknownCallback*)

Registers the callback for an unknown definition.

Parameters

globalDefReaderCallbacks	Struct for all callbacks.
unknownCallback	Function which should be called for all Unknown definitions.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.19 otf2/OTF2_GlobalDefWriter.h File Reference

This layer always writes globally defined OTF2 definition records and is used to write either the global definitions in addition to local definitions or write all definitions as globally valid in combination with OTF2_GlobalEventWriter. Global definitions are stored in one global definition file, which makes it nearly impossible to write them in a distributed manner. It is therefore only allowed to get such a writer from an OTF2_Archive which is the master in the collective context.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeValue.h>
#include <otf2/OTF2_Definitions.h>
```

Typedefs

- typedef struct OTF2_GlobalDefWriter_struct OTF2_GlobalDefWriter

  Typedef of the struct which keeps all necessary information of a global definition writer. Can be used to reference these structs from external.

Functions

- OTF2_ErrorCode OTF2_GlobalDefWriter_GetNumberOfDefinitions (OTF2_-GlobalDefWriter ∗writerHandle, uint64_t ∗numberOfDefinitions)

  Returns the current number of written definitions of a global definition writer.
- OTF2_ErrorCode OTF2_GlobalDefWriter_GetNumberOfLocations (OTF2_-GlobalDefWriter ∗writerHandle, uint64_t ∗numberOfLocations)

  Returns the current number of written location definitions of a global definition writer.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteAttribute (OTF2_GlobalDefWriter ∗writerHandle, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_Type type)

  Writes a Attribute definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallingContext (OTF2_GlobalDefWriter ∗writerHandle, OTF2_CallingContextRef self, uint64_t ip, OTF2_RegionRef region, uint32_t offsetLineNumber, OTF2_CallingContextRef parent)

  Writes a CallingContext definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallpath (OTF2_GlobalDefWriter ∗writerHandle, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_-RegionRef region)

Writes a Callpath definition record into the GlobalDefWriter.

- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallsite (OTF2_GlobalDefWriter ∗writerHandle, OTF2_CallsiteRef self, OTF2_StringRef sourceFile, uint32_-t lineNumber, OTF2_RegionRef enteredRegion, OTF2_RegionRef leftRegion)

  Writes a Callsite definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCartCoordinate (OTF2_GlobalDefWriter ∗writerHandle, OTF2_CartTopologyRef cartTopology, uint32_t rank, uint8_-t numberOfDimensions, const uint32_t ∗coordinates)

  Writes a CartCoordinate definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCartDimension (OTF2_GlobalDefWriter ∗writerHandle, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_-t size, OTF2_CartPeriodicity cartPeriodicity)

  Writes a CartDimension definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCartTopology (OTF2_GlobalDefWriter ∗writerHandle, OTF2_CartTopologyRef self, OTF2_StringRef name, OTF2_-CommRef communicator, uint8_t numberOfDimensions, const OTF2_CartDimensionRef ∗cartDimensions)

  Writes a CartTopology definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteClockProperties (OTF2_GlobalDefWriter ∗writerHandle, uint64_t timerResolution, uint64_t globalOffset, uint64_t traceLength)

  Writes a ClockProperties definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteComm (OTF2_GlobalDefWriter ∗writerHandle, OTF2_CommRef self, OTF2_StringRef name, OTF2_GroupRef group, OTF2_CommRef parent)

  Writes a Comm definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteGroup (OTF2_GlobalDefWriter ∗writerHandle, OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_-t numberOfMembers, const uint64_t ∗members)

  Writes a Group definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteInterruptGenerator (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_InterruptGeneratorRef self, OTF2_-StringRef name, OTF2_StringRef unit, uint64_t period)

  Writes a InterruptGenerator definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocation (OTF2_GlobalDefWriter ∗writerHandle, OTF2_LocationRef self, OTF2_StringRef name, OTF2_LocationType locationType, uint64_t numberOfEvents, OTF2_LocationGroupRef location-Group)

  Writes a Location definition record into the GlobalDefWriter.

- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocationGroup (OTF2_GlobalDefWriter ∗writerHandle, OTF2_LocationGroupRef self, OTF2_StringRef name, OTF2_-LocationGroupType locationGroupType, OTF2_SystemTreeNodeRef systemTreeParent)

  Writes a LocationGroup definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocationGroupProperty (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_LocationGroupRef locationGroup, OTF2_StringRef name, OTF2_StringRef value)

  Writes a LocationGroupProperty definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocationProperty (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_LocationRef location, OTF2_StringRef name, OTF2_StringRef value)

  Writes a LocationProperty definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricClass (OTF2_GlobalDefWriter ∗writerHandle, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_-MetricMemberRef ∗metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)

  Writes a MetricClass definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricClassRecorder (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)

  Writes a MetricClassRecorder definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricInstance (OTF2_GlobalDefWriter ∗writerHandle, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_-LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)

  Writes a MetricInstance definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricMember (OTF2_GlobalDefWriter ∗writerHandle, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_-StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_-t exponent, OTF2_StringRef unit)

  Writes a MetricMember definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteParadigm (OTF2_GlobalDefWriter ∗writerHandle, OTF2_Paradigm paradigm, OTF2_StringRef name, OTF2_-ParadigmClass paradigmClass)

  Writes a Paradigm definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteParadigmProperty (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_Paradigm paradigm, OTF2_ParadigmProperty property, OTF2_Type type, OTF2_AttributeValue attributeValue)

  Writes a ParadigmProperty definition record into the GlobalDefWriter.

- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteParameter (OTF2_GlobalDefWriter ∗writerHandle, OTF2_ParameterRef self, OTF2_StringRef name, OTF2_-ParameterType parameterType)

  Writes a Parameter definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteRegion (OTF2_GlobalDefWriter ∗writerHandle, OTF2_RegionRef self, OTF2_StringRef name, OTF2_StringRef canonicalName, OTF2_StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)

  Writes a Region definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteRmaWin (OTF2_GlobalDefWriter ∗writerHandle, OTF2_RmaWinRef self, OTF2_StringRef name, OTF2_CommRef comm)

  Writes a RmaWin definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSourceCodeLocation (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_SourceCodeLocationRef self, OTF2_-StringRef file, uint32_t lineNumber)

  Writes a SourceCodeLocation definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteString (OTF2_GlobalDefWriter ∗writerHandle, OTF2_StringRef self, const char ∗string)

  Writes a String definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNode (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_SystemTreeNodeRef self, OTF2_-StringRef name, OTF2_StringRef className, OTF2_SystemTreeNodeRef parent)

  Writes a SystemTreeNode definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNodeDomain (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_SystemTreeNodeRef systemTreeN-ode, OTF2_SystemTreeDomain systemTreeDomain)

  Writes a SystemTreeNodeDomain definition record into the GlobalDefWriter.
- OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty (OTF2_-GlobalDefWriter ∗writerHandle, OTF2_SystemTreeNodeRef systemTreeN-ode, OTF2_StringRef name, OTF2_StringRef value)

  Writes a SystemTreeNodeProperty definition record into the GlobalDefWriter.

E.19.1 Detailed Description

This layer always writes globally defined OTF2 definition records and is used to write either the global definitions in addition to local definitions or write all definitions as globally valid in combination with OTF2_GlobalEventWriter. Global definitions are stored in one global definition file, which makes it nearly impossible to write them in a distributed manner. It is therefore only allowed to get such a writer from an OTF2_Archive which is the master in the collective context.

Source Template:

templates/OTF2_GlobalDefWriter.tmpl.h

E.19.2 Function Documentation

E.19.2.1 OTF2_ErrorCode OTF2_GlobalDefWriter_GetNumberOfDefinitions (OTF2_GlobalDefWriter ∗ *writerHandle,* uint64_t ∗ *numberOfDefinitions*)

Returns the current number of written definitions of a global definition writer.

Parameters

	writerHandle	Handle to the global definition writer.
out	*numberOfDefinitions*	Storage for the number of definitions.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.2 OTF2_ErrorCode OTF2_GlobalDefWriter_GetNumberOfLocations (OTF2_GlobalDefWriter ∗ *writerHandle,* uint64_t ∗ *numberOfLocations*)

Returns the current number of written location definitions of a global definition writer.

Parameters

	writerHandle	Handle to the global definition writer.
out	*numberOfLocations*	Storage for the number of locations.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.3 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteAttribute (OTF2_GlobalDefWriter * *writerHandle*, OTF2_AttributeRef *self*, OTF2_StringRef *name*, OTF2_StringRef *description*, OTF2_Type *type*)

Writes a *Attribute* definition record into the GlobalDefWriter.

The attribute definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *Attribute* definition.
name	Name of the attribute. References a *String* definition.
description	Description of the attribute. References a *String* definition. Since version 1.4.
type	Type of the attribute value.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.4 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallingContext (OTF2_GlobalDefWriter * *writerHandle*, OTF2_CallingContextRef *self*, uint64_t *ip*, OTF2_RegionRef *region*, uint32_t *offsetLineNumber*, OTF2_CallingContextRef *parent*)

Writes a *CallingContext* definition record into the GlobalDefWriter.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *CallingContext* definition.
ip	Instruction pointer as the offset to the start of the function.
region	The region. References a *Region* definition.
offsetLineNumber	The line offset inside the region.
parent	Parent id of this context. References a *CallingContext* definition.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.5 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallpath (OTF2_GlobalDefWriter ∗ *writerHandle,* OTF2_CallpathRef *self,* OTF2_CallpathRef *parent,* OTF2_RegionRef *region*)

Writes a *Callpath* definition record into the GlobalDefWriter.

The callpath definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *Callpath* definition.
parent	The parent of this callpath. References a *Callpath* definition.
region	The region of this callpath. References a *Region* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.6 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallsite (OTF2_GlobalDefWriter ∗ *writerHandle,* OTF2_CallsiteRef *self,* OTF2_StringRef *sourceFile,* uint32_t *lineNumber,* OTF2_RegionRef *enteredRegion,* OTF2_RegionRef *leftRegion*)

Writes a *Callsite* definition record into the GlobalDefWriter.

The callsite definition.

Parameters

writerHandle	The writer handle.

self	The unique identifier for this *Callsite* definition.
sourceFile	The source file where this call was made. References a *String* definition.
lineNumber	Line number in the source file where this call was made.
enteredRegion	The region which was called. References a *Region* definition.
leftRegion	The region which made the call. References a *Region* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.7 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCartCoordinate (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_CartTopologyRef *cartTopology*, uint32_t *rank*, uint8_t *numberOfDimensions*, const uint32_t ∗ *coordinates*)

Writes a *CartCoordinate* definition record into the GlobalDefWriter.

Defines the coordinate of the location referenced by the given rank (w.r.t. the communicator associated to the topology) in the referenced topology.

Parameters

writerHandle	The writer handle.
cartTopology	Parent *CartTopology* definition to which this one is a supplementary definition. References a *CartTopology* definition.
rank	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
numberOfDimensions	Number of dimensions.
coordinates	Coordinates, indexed by dimension.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.8 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCartDimension (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_CartDimensionRef *self*, OTF2_StringRef *name*, uint32_t *size*, OTF2_CartPeriodicity *cartPeriodicity*)

Writes a *CartDimension* definition record into the GlobalDefWriter.

Each dimension in a Cartesian topology is composed of a global id, a name, its size, and whether it is periodic or not.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *CartDimension* definition.
name	The name of the cartesian topology dimension. References a *String* definition.
size	The size of the cartesian topology dimension.
cartPeriodicity	Periodicity of the cartesian topology dimension.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.9 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCartTopology (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_CartTopologyRef *self*, OTF2_StringRef *name*, OTF2_CommRef *communicator*, uint8_t *numberOfDimensions*, const OTF2_CartDimensionRef ∗ *cartDimensions*)

Writes a *CartTopology* definition record into the GlobalDefWriter.

Each topology is described by a global id, a reference to its name, a reference to a communicator, the number of dimensions, and references to those dimensions. The topology type is defined by the paradigm of the group referenced by the associated communicator.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *CartTopology* definition.

name	The name of the topology. References a *String* definition.
communicator	Communicator object used to create the topology. References a *Comm* definition.
numberOfDimensions	Number of dimensions.
cartDimensions	The dimensions of this topology. References a *CartDimension* definition.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.10 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteClockProperties (OTF2_GlobalDefWriter * *writerHandle*, uint64_t *timerResolution*, uint64_t *globalOffset*, uint64_t *traceLength*)

Writes a *ClockProperties* definition record into the GlobalDefWriter.

Defines the timer resolution and time range of this trace. There will be no event with a timestamp less than `globalOffset`, and no event with timestamp greater than (`globalOffset` + `traceLength`).

Parameters

writerHandle	The writer handle.
timerResolution	Ticks per seconds.
globalOffset	A timestamp smaller than all event timestamps.
traceLength	A timespan which includes the timespan between the smallest and greatest timestamp of all event timestamps.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.11 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteComm (OTF2_GlobalDefWriter * *writerHandle*, OTF2_CommRef *self*, OTF2_StringRef *name*, OTF2_GroupRef *group*, OTF2_CommRef *parent*)

Writes a *Comm* definition record into the GlobalDefWriter.

The communicator definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *Comm* definition.
name	The name given by calling MPI_Comm_set_name on this communicator. Or the empty name to indicate that no name was given. References a *String* definition.
group	The describing MPI group of this MPI communicator The group needs to be of type *OTF2_GROUP_TYPE_COMM_GROUP* or *OTF2_GROUP_TYPE_COMM_SELF*. References a *Group* definition.
parent	The parent MPI communicator from which this communicator was created, if any. Use *OTF2_UNDEFINED_COMM* to indicate no parent. References a *Comm* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.12 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteGroup (OTF2_GlobalDefWriter * *writerHandle*, OTF2_GroupRef *self*, OTF2_StringRef *name*, OTF2_GroupType *groupType*, OTF2_Paradigm *paradigm*, OTF2_GroupFlag *groupFlags*, uint32_t *numberOfMembers*, const uint64_t * *members*)

Writes a *Group* definition record into the GlobalDefWriter.

The group definition.

Parameters

writerHandle	The writer handle.

self	The unique identifier for this *Group* definition.
name	Name of this group References a *String* definition.
groupType	The type of this group. Since version 1.2.
paradigm	The paradigm of this communication group. Since version 1.2.
groupFlags	Flags for this group. Since version 1.2.
numberOfMembers	The number of members in this group.
members	The identifiers of the group members.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.13 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteInterruptGenerator (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_InterruptGeneratorRef *self*, OTF2_StringRef *name*, OTF2_StringRef *unit*, uint64_t *period*)

Writes a *InterruptGenerator* definition record into the GlobalDefWriter.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *InterruptGenerator* definition.
name	The name of this interrupt generator. References a *String* definition.
unit	The unit used by this interrupt generator for the period. References a *String* definition.
period	The period this interrupt generator generates interrupts.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.14 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocation (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_LocationRef *self*, OTF2_StringRef *name*, OTF2_LocationType *locationType*, uint64_t *numberOfEvents*, OTF2_LocationGroupRef *locationGroup*)

Writes a *Location* definition record into the GlobalDefWriter.

The location definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *Location* definition.
name	Name of the location References a *String* definition.
locationType	Location type.
numberOfEvents	Number of events this location has recorded.
locationGroup	Location group which includes this location. References a *LocationGroup* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.15 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocationGroup (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_LocationGroupRef *self*, OTF2_StringRef *name*, OTF2_LocationGroupType *locationGroupType*, OTF2_SystemTreeNodeRef *systemTreeParent*)

Writes a *LocationGroup* definition record into the GlobalDefWriter.

The location group definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *LocationGroup* definition.
name	Name of the group. References a *String* definition.

locationGroupType	Type of this group.
systemTreeParent	Parent of this location group in the system tree. References a *SystemTreeNode* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.16 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocationGroupProperty (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_LocationGroupRef *locationGroup*, OTF2_StringRef *name*, OTF2_StringRef *value*)

Writes a *LocationGroupProperty* definition record into the GlobalDefWriter.

An arbitrary key/value property for a *LocationGroup* definition.

Parameters

writerHandle	The writer handle.
locationGroup	Parent *LocationGroup* definition to which this one is a supplementary definition. References a *LocationGroup* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.17 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocationProperty (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_LocationRef *location*, OTF2_StringRef *name*, OTF2_StringRef *value*)

Writes a *LocationProperty* definition record into the GlobalDefWriter.

An arbitrary key/value property for a *Location* definition.

Parameters

writerHandle	The writer handle.
location	Parent *Location* definition to which this one is a supplementary definition. References a *Location* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.18 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricClass (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_MetricRef *self*, uint8_t *numberOfMetrics*, const OTF2_MetricMemberRef ∗ *metricMembers*, OTF2_MetricOccurrence *metricOccurrence*, OTF2_RecorderKind *recorderKind*)

Writes a *MetricClass* definition record into the GlobalDefWriter.

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *MetricClass* definition.
numberOfMetrics	Number of metrics within the set.
metricMembers	List of metric members. References a *MetricMember* definition.
metricOccurrence	Defines occurrence of a metric set.
recorderKind	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.19 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricClassRecorder (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_MetricRef *metricClass*, OTF2_LocationRef *recorder*)

Writes a *MetricClassRecorder* definition record into the GlobalDefWriter.

The metric class recorder definition.

Parameters

writerHandle	The writer handle.
metricClass	Parent *MetricClass* definition to which this one is a supplementary definition. References a *MetricClass* definition.
recorder	The location which recorded the referenced metric class. References a *Location* definition.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.20 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricInstance (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_MetricRef *self*, OTF2_MetricRef *metricClass*, OTF2_LocationRef *recorder*, OTF2_MetricScope *metricScope*, uint64_t *scope*)

Writes a *MetricInstance* definition record into the GlobalDefWriter.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type *OTF2_METRIC_ASYNCHRONOUS*.

Parameters

writerHan-dle	The writer handle.
self	The unique identifier for this *MetricClass* definition.
metricClass	The instanced *MetricClass*. This metric class must be of kind *OTF2_-RECORDER_KIND_ABSTRACT*. References a *MetricClass* definition.
recorder	Recorder of the metric: location ID. References a *Location* definition.
metric-Scope	Defines type of scope: location, location group, system tree node, or a generic group of locations.
scope	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.21 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricMember (OTF2_GlobalDefWriter * *writerHandle*, OTF2_MetricMemberRef *self*, OTF2_StringRef *name*, OTF2_StringRef *description*, OTF2_MetricType *metricType*, OTF2_MetricMode *metricMode*, OTF2_Type *valueType*, OTF2_MetricBase *metricBase*, int64_t *exponent*, OTF2_StringRef *unit*)

Writes a *MetricMember* definition record into the GlobalDefWriter.

A metric is defined by a metric member definition. A metric member is always a member of a metric class. Therefore, a single metric is a special case of a metric class with only one member. It is not allowed to reference a metric member id in a metric event, but only metric class IDs.

Parameters

writerHan-dle	The writer handle.
self	The unique identifier for this *MetricMember* definition.
name	Name of the metric. References a *String* definition.
description	Description of the metric. References a *String* definition.
metricType	Metric type: PAPI, etc.
metricMode	Metric mode: accumulative, fix, relative, etc.

valueType	Type of the value. Only *OTF2_TYPE_INT64*, *OTF2_TYPE_UINT64*, and *OTF2_TYPE_DOUBLE* are valid types. If this metric member is recorded in an *Metric* event, than this type and the type in the event must match.
metricBase	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
exponent	The values inside the Metric events should be scaled by the factor base^exponent, to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be *OTF2_BASE_-BINARY* and the exponent 10. Than the writer does not need to scale the values up to bytes, but can directly write the KiBi values into the Metric event. At reading time, the reader can apply the scaling factor to get the value in its base unit, ie. in bytes.
unit	Unit of the metric. This needs to be the scale free base unit, ie. "bytes", "operations", or "seconds". In particular this unit should not have any scale prefix. References a *String* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.22 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteParadigm (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_Paradigm *paradigm*, OTF2_StringRef *name*, OTF2_ParadigmClass *paradigmClass*)

Writes a *Paradigm* definition record into the GlobalDefWriter.

Attests that the following parallel paradigm was available at the time when the trace was recorded, and vice versa. Note that this does not attest that the paradigm was used. For convenience, this also includes a proper name for the paradigm and a classification. This definition is only allowed to appear at most once in the definitions per *Paradigm*.

Parameters

writerHandle	The writer handle.
paradigm	The paradigm to attest.
name	The name of the paradigm. References a *String* definition.
paradigmClass	The class of this paradigm.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.23 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteParadigmProperty (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_Paradigm *paradigm*, OTF2_ParadigmProperty *property*, OTF2_Type *type*, OTF2_AttributeValue *attributeValue*)

Writes a *ParadigmProperty* definition record into the GlobalDefWriter.

Extensible annotation for the *Paradigm* definition.

The tuple (*paradigm*, *property*) must be unique.

Parameters

writerHandle	The writer handle.
paradigm	The paradigm to annotate.
property	The property.
type	The type of this property. Must match with the defined type of the *property*.
attributeValue	The value of this property.value

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.24 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteParameter (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_ParameterRef *self*, OTF2_StringRef *name*, OTF2_ParameterType *parameterType*)

Writes a *Parameter* definition record into the GlobalDefWriter.

The parameter definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *Parameter* definition.
name	Name of the parameter (variable name etc.) References a *String* definition.
parameterType	Type of the parameter, *OTF2_ParameterType* for possible types.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.25 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteRegion (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_RegionRef *self*, OTF2_StringRef *name*, OTF2_StringRef *canonicalName*, OTF2_StringRef *description*, OTF2_RegionRole *regionRole*, OTF2_Paradigm *paradigm*, OTF2_RegionFlag *regionFlags*, OTF2_StringRef *sourceFile*, uint32_t *beginLineNumber*, uint32_t *endLineNumber*)

Writes a *Region* definition record into the GlobalDefWriter.

The region definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *Region* definition.
name	Name of the region (demangled name if available). References a *String* definition.
canonicalName	Alternative name of the region (e.g. mangled name). References a *String* definition. Since version 1.1.
description	A more detailed description of this region. References a *String* definition.
regionRole	Region role. Since version 1.1.
paradigm	Paradigm. Since version 1.1.
regionFlags	Region flags. Since version 1.1.

sourceFile	The source file where this region was declared. References a *String* definition.
beginLineNumber	Starting line number of this region in the source file.
endLineNumber	Ending line number of this region in the source file.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.26 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteRmaWin (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_RmaWinRef *self*, OTF2_StringRef *name*, OTF2_CommRef *comm*)

Writes a *RmaWin* definition record into the GlobalDefWriter.

A window defines the communication context for any remote-memory access operation.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *RmaWin* definition.
name	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a *String* definition.
comm	Communicator object used to create the window. References a *Comm* definition.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.27 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSourceCodeLocation (OTF2_GlobalDefWriter * *writerHandle*, OTF2_SourceCodeLocationRef *self*, OTF2_StringRef *file*, uint32_t *lineNumber*)

Writes a *SourceCodeLocation* definition record into the GlobalDefWriter.

The definition of a source code location as tuple of the corresponding file name and line number.

When used to attach source code annotations to events, use the *OTF2_AttributeList* with a *Attribute* definition named `"SOURCE_CODE_LOCATION"` and typed *OTF2_TYPE_SOURCE_CODE_LOCATION*.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *SourceCodeLocation* definition.
file	The name of the file for the source code location. References a *String* definition.
lineNumber	The line number for the source code location.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.28 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteString (OTF2_GlobalDefWriter * *writerHandle*, OTF2_StringRef *self*, const char * *string*)

Writes a *String* definition record into the GlobalDefWriter.

The string definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *String* definition.
string	The string, null terminated.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.29 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNode (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_SystemTreeNodeRef *self*, OTF2_StringRef *name*, OTF2_StringRef *className*, OTF2_SystemTreeNodeRef *parent*)

Writes a *SystemTreeNode* definition record into the GlobalDefWriter.

The system tree node definition.

Parameters

writerHandle	The writer handle.
self	The unique identifier for this *SystemTreeNode* definition.
name	Free form instance name of this node. References a *String* definition.
className	Free form class name of this node References a *String* definition.
parent	Parent id of this node. May be *OTF2_UNDEFINED_SYSTEM_TREE_NODE* to indicate that there is no parent. References a *SystemTreeNode* definition.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.30 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNodeDomain (OTF2_GlobalDefWriter ∗ *writerHandle*, OTF2_SystemTreeNodeRef *systemTreeNode*, OTF2_SystemTreeDomain *systemTreeDomain*)

Writes a *SystemTreeNodeDomain* definition record into the GlobalDefWriter.

The system tree node domain definition.

Parameters

writerHandle	The writer handle.
systemTreeNode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
systemTreeDomain	The domain in which the referenced *SystemTreeNode* operates in.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.19.2.31 OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty (OTF2_GlobalDefWriter * *writerHandle*, OTF2_SystemTreeNodeRef *systemTreeNode*, OTF2_StringRef *name*, OTF2_StringRef *value*)

Writes a *SystemTreeNodeProperty* definition record into the GlobalDefWriter.

An arbitrary key/value property for a *SystemTreeNode* definition.

Parameters

writerHandle	The writer handle.
systemTreeNode	Parent *SystemTreeNode* definition to which this one is a supplementary definition. References a *SystemTreeNode* definition.
name	Name of the property. References a *String* definition.
value	Property value. References a *String* definition.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.20 otf2/OTF2_GlobalEvtReader.h File Reference

This is the global event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_EvtReader.h>
#include <otf2/OTF2_GlobalEvtReaderCallbacks.h>
```

Functions

- OTF2_ErrorCode OTF2_GlobalEvtReader_HasEvent (OTF2_GlobalEvtReader ∗reader, int ∗flag)

  Has more events.
- OTF2_ErrorCode OTF2_GlobalEvtReader_ReadEvent (OTF2_GlobalEvtReader ∗reader)

  Triggers the callback for the next event record.
- OTF2_ErrorCode OTF2_GlobalEvtReader_ReadEvents (OTF2_GlobalEvtReader ∗reader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  Reads the given number of records from the global event reader.
- OTF2_ErrorCode OTF2_GlobalEvtReader_SetCallbacks (OTF2_GlobalEvtReader ∗reader, const OTF2_GlobalEvtReaderCallbacks ∗callbacks, void ∗userData)

  Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

E.20.1 Detailed Description

This is the global event reader. Used to read from multiple local event readers, and provide them in a timely ordered sequence.

E.20.2 Function Documentation

E.20.2.1 OTF2_ErrorCode OTF2_GlobalEvtReader_HasEvent (OTF2_GlobalEvtReader ∗ *reader,* int ∗ *flag*)

Has more events.

Parameters

	reader	Global event reader handle.
out	*flag*	In case of success, the flag will be set to 1 when there is at least more more event to read. To 0 if not. Otherwise the value is undefined.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.20.2.2 OTF2_ErrorCode OTF2_GlobalEvtReader_ReadEvent (OTF2_GlobalEvtReader * *reader*)

Triggers the callback for the next event record.

Parameters

reader	Reader object which reads the events from its buffer.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.20.2.3 OTF2_ErrorCode OTF2_GlobalEvtReader_ReadEvents (OTF2_GlobalEvtReader * *reader,* uint64_t *recordsToRead,* uint64_t * *recordsRead*)

Reads the given number of records from the global event reader.

Parameters

	reader	The records of this reader will be read when the function is issued.
	record-sToRead	This variable tells the reader how much records it has to read.
out	*record-sRead*	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given record-sToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finnished his job by checking recordsRead < recordsToRead.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.20.2.4 OTF2_ErrorCode OTF2_GlobalEvtReader_SetCallbacks (OTF2_-GlobalEvtReader ∗ *reader,* const OTF2_GlobalEvtReaderCallbacks ∗ *callbacks,* void ∗ *userData*)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

reader	Reader object which reads the events from its buffer.
callbacks	Struct which holds a function pointer for each record type. *OTF2_-GlobalEvtReaderCallbacks_New*.
userData	Data passed as argument *userData* to the record callbacks.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.21 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

This defines the callbacks for the global event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_BufferFlush)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp stopTime)

  Callback for the BufferFlush event record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_CallingContextSample)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CallingContextRef callingContext, uint32_t unwindDistance, OTF2_InterruptGeneratorRef interruptGenerator)

  Callback for the CallingContextSample event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_Enter)(OTF2_-LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_RegionRef region)

  Callback for the Enter event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_Leave)(OTF2_-LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_RegionRef region)

  Callback for the Leave event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MeasurementOnOff)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_MeasurementMode measurement-Mode)

  Callback for the MeasurementOnOff event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_Metric)(OTF2_-LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_MetricRef metric, uint8_t numberOfMet-rics, const OTF2_Type ∗typeIDs, const OTF2_MetricValue ∗metricValues)

  Callback for the Metric event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList)

  Callback for the MpiCollectiveBegin event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CollectiveOp collectiveOp, OTF2_-CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeRe-ceived)

  Callback for the MpiCollectiveEnd event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiIrecv)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t sender, OTF2_CommRef com-municator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

  Callback for the MpiIrecv event record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiIrecvRequest)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

  Callback for the MpiIrecvRequest event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiIsend)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

  Callback for the MpiIsend event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiIsendComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

  Callback for the MpiIsendComplete event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiRecv)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Callback for the MpiRecv event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiRequestCancelled)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

  Callback for the MpiRequestCancelled event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiRequestTest)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

  Callback for the MpiRequestTest event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiSend)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Callback for the MpiSend event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_OmpAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t lockID, uint32_t acquisitionOrder)

  Callback for the OmpAcquireLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_OmpFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t numberOfRequestedThreads)

  Callback for the OmpFork event record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_OmpJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList)

  Callback for the OmpJoin event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_OmpReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t lockID, uint32_t acquisitionOrder)

  Callback for the OmpReleaseLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_OmpTaskComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

  Callback for the OmpTaskComplete event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

  Callback for the OmpTaskCreate event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_OmpTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

  Callback for the OmpTaskSwitch event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, int64_t value)

  Callback for the ParameterInt event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ParameterString)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, OTF2_-StringRef string)

  Callback for the ParameterString event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, uint64_t value)

  Callback for the ParameterUnsignedInt event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

  Callback for the RmaAcquireLock event record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaAtomic)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)

  Callback for the RmaAtomic event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList)

  Callback for the RmaCollectiveBegin event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CollectiveOp collectiveOp, OTF2_-RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)

  Callback for the RmaCollectiveEnd event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaGet)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)

  Callback for the RmaGet event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaGroupSync)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_-RmaWinRef win, OTF2_GroupRef group)

  Callback for the RmaGroupSync event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaOpCompleteBlocking)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

  Callback for the RmaOpCompleteBlocking event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaOpCompleteNonBlocking)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

  Callback for the RmaOpCompleteNonBlocking event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaOpCompleteRemote)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

  Callback for the RmaOpCompleteRemote event record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaOpTest)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

  Callback for the RmaOpTest event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaPut)(OTF2_-LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_-t bytes, uint64_t matchingId)

  Callback for the RmaPut event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)

  Callback for the RmaReleaseLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaRequestLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

  Callback for the RmaRequestLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaSync)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaSyncType syncType)

  Callback for the RmaSync event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaTryLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

  Callback for the RmaTryLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaWaitChange)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

  Callback for the RmaWaitChange event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaWinCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

  Callback for the RmaWinCreate event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_RmaWinDestroy)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

Callback for the RmaWinDestroy event record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

  Callback for the ThreadAcquireLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_-t sequenceCount)

  Callback for the ThreadBegin event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_-t sequenceCount)

  Callback for the ThreadCreate event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_-t sequenceCount)

  Callback for the ThreadEnd event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)

  Callback for the ThreadFork event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model)

  Callback for the ThreadJoin event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

  Callback for the ThreadReleaseLock event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadTaskComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Callback for the ThreadTaskComplete event record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Callback for the ThreadTaskCreate event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

  Callback for the ThreadTaskSwitch event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadTeamBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam)

  Callback for the ThreadTeamBegin event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadTeamEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam)

  Callback for the ThreadTeamEnd event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadWait)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_-t sequenceCount)

  Callback for the ThreadWait event record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList)

  Callback for an unknown event record.
- typedef struct OTF2_GlobalEvtReaderCallbacks_struct OTF2_GlobalEvtReaderCallbacks

  Opaque struct which holdes all event record callbacks.

Functions

- void OTF2_GlobalEvtReaderCallbacks_Clear (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks)

  Clears a struct for the global event callbacks.
- void OTF2_GlobalEvtReaderCallbacks_Delete (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks)

  Deallocates a struct for the global event callbacks.
- OTF2_GlobalEvtReaderCallbacks ∗ OTF2_GlobalEvtReaderCallbacks_New (void)

Allocates a new struct for the event callbacks.

- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetBufferFlushCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-BufferFlush bufferFlushCallback)

  Registers the callback for the BufferFlush event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetCallingContextSampleCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-CallingContextSample callingContextSampleCallback)

  Registers the callback for the CallingContextSample event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetEnterCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-Enter enterCallback)

  Registers the callback for the Enter event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-Leave leaveCallback)

  Registers the callback for the Leave event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMeasurementOnOffCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MeasurementOnOff measurementOnOffCallback)

  Registers the callback for the MeasurementOnOff event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMetricCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-Metric metricCallback)

  Registers the callback for the Metric event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiCollectiveBegin mpiCollectiveBeginCallback)

  Registers the callback for the MpiCollectiveBegin event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiCollectiveEnd mpiCollectiveEndCallback)

  Registers the callback for the MpiCollectiveEnd event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiIrecv mpiIrecvCallback)

  Registers the callback for the MpiIrecv event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiIrecvRequest mpiIrecvRequestCallback)

  Registers the callback for the MpiIrecvRequest event.

- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiIsend mpiIsendCallback)

  Registers the callback for the MpiIsend event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiIsendComplete mpiIsendCompleteCallback)

  Registers the callback for the MpiIsendComplete event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiRecvCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiRecv mpiRecvCallback)

  Registers the callback for the MpiRecv event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiRequestCancelledCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiRequestCancelled mpiRequestCancelledCallback)

  Registers the callback for the MpiRequestCancelled event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiRequestTestCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiRequestTest mpiRequestTestCallback)

  Registers the callback for the MpiRequestTest event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiSendCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-MpiSend mpiSendCallback)

  Registers the callback for the MpiSend event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpAcquireLockCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-OmpAcquireLock ompAcquireLockCallback)

  Registers the callback for the OmpAcquireLock event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpForkCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-OmpFork ompForkCallback)

  Registers the callback for the OmpFork event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpJoinCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-OmpJoin ompJoinCallback)

  Registers the callback for the OmpJoin event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpReleaseLockCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-OmpReleaseLock ompReleaseLockCallback)

  Registers the callback for the OmpReleaseLock event.

- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCompleteCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-OmpTaskComplete ompTaskCompleteCallback)

  Registers the callback for the OmpTaskComplete event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCreateCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-OmpTaskCreate ompTaskCreateCallback)

  Registers the callback for the OmpTaskCreate event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskSwitchCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-OmpTaskSwitch ompTaskSwitchCallback)

  Registers the callback for the OmpTaskSwitch event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterIntCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ParameterInt parameterIntCallback)

  Registers the callback for the ParameterInt event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterStringCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ParameterString parameterStringCallback)

  Registers the callback for the ParameterString event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterUnsignedIntCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ParameterUnsignedInt parameterUnsignedIntCallback)

  Registers the callback for the ParameterUnsignedInt event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaAcquireLockCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaAcquireLock rmaAcquireLockCallback)

  Registers the callback for the RmaAcquireLock event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaAtomicCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaAtomic rmaAtomicCallback)

  Registers the callback for the RmaAtomic event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveBeginCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaCollectiveBegin rmaCollectiveBeginCallback)

  Registers the callback for the RmaCollectiveBegin event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveEndCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaCollectiveEnd rmaCollectiveEndCallback)

  Registers the callback for the RmaCollectiveEnd event.

- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaGetCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaGet rmaGetCallback)

  Registers the callback for the RmaGet event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaGroupSyncCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaGroupSync rmaGroupSyncCallback)

  Registers the callback for the RmaGroupSync event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteBlockingCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaOpCompleteBlocking rmaOpCompleteBlockingCallback)

  Registers the callback for the RmaOpCompleteBlocking event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaOpCompleteNonBlocking rmaOpCompleteNonBlockingCallback)

  Registers the callback for the RmaOpCompleteNonBlocking event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteRemoteCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaOpCompleteRemote rmaOpCompleteRemoteCallback)

  Registers the callback for the RmaOpCompleteRemote event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpTestCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaOpTest rmaOpTestCallback)

  Registers the callback for the RmaOpTest event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaPutCallback (OTF2_-GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaPut rmaPutCallback)

  Registers the callback for the RmaPut event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaReleaseLockCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaReleaseLock rmaReleaseLockCallback)

  Registers the callback for the RmaReleaseLock event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaRequestLockCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaRequestLock rmaRequestLockCallback)

  Registers the callback for the RmaRequestLock event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaSyncCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaSync rmaSyncCallback)

  Registers the callback for the RmaSync event.

- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaTryLockCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaTryLock rmaTryLockCallback)

  Registers the callback for the RmaTryLock event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaWaitChangeCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaWaitChange rmaWaitChangeCallback)

  Registers the callback for the RmaWaitChange event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaWinCreateCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaWinCreate rmaWinCreateCallback)

  Registers the callback for the RmaWinCreate event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaWinDestroyCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaWinDestroy rmaWinDestroyCallback)

  Registers the callback for the RmaWinDestroy event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadAcquireLockCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadAcquireLock threadAcquireLockCallback)

  Registers the callback for the ThreadAcquireLock event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadBeginCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadBegin threadBeginCallback)

  Registers the callback for the ThreadBegin event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadCreateCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadCreate threadCreateCallback)

  Registers the callback for the ThreadCreate event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadEndCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadEnd threadEndCallback)

  Registers the callback for the ThreadEnd event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadForkCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadFork threadForkCallback)

  Registers the callback for the ThreadFork event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadJoinCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadJoin threadJoinCallback)

  Registers the callback for the ThreadJoin event.

- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadReleaseLockCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadReleaseLock threadReleaseLockCallback)

  Registers the callback for the ThreadReleaseLock event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTaskCompleteCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadTaskComplete threadTaskCompleteCallback)

  Registers the callback for the ThreadTaskComplete event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTaskCreateCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadTaskCreate threadTaskCreateCallback)

  Registers the callback for the ThreadTaskCreate event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTaskSwitchCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadTaskSwitch threadTaskSwitchCallback)

  Registers the callback for the ThreadTaskSwitch event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTeamBeginCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadTeamBegin threadTeamBeginCallback)

  Registers the callback for the ThreadTeamBegin event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTeamEndCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadTeamEnd threadTeamEndCallback)

  Registers the callback for the ThreadTeamEnd event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadWaitCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-ThreadWait threadWaitCallback)

  Registers the callback for the ThreadWait event.
- OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetUnknownCallback (OTF2_GlobalEvtReaderCallbacks ∗globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-Unknown unknownCallback)

  Registers the callback for unknown events.

E.21.1 Detailed Description

This defines the callbacks for the global event reader.

Source Template:

templates/OTF2_GlobalEvtReaderCallbacks.tmpl.h

E.21.2 Typedef Documentation

E.21.2.1 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-BufferFlush)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp stopTime)

Callback for the BufferFlush event record.

This event signals that the internal buffer was flushed at the given time.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
stopTime	The time the buffer flush finished.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.2 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-CallingContextSample)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CallingContextRef callingContext, uint32_t unwindDistance, OTF2_InterruptGeneratorRef interruptGenerator)

Callback for the CallingContextSample event record.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
callingCon-text	References a *CallingContext* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_CALLING_-CONTEXT* is available.

unwindDistance	The unwindContext specifies the first context whose ip(return adress) was still marked since the last sample this means that no progress was made in the repsective region The last region that was not returned from since the last sample Is one stack level higher, but may now be at at different line number OTF2_CallingContextRef unwindContext; However, instead of this we specify the distance (number of intermediate edges) between the calling context and the unwind context Note: unwindDistance=0 would mean no progress in the leaf region since the last sample which is unlikely If not available, UNDEFINED should be used.
interruptGenerator	References a *InterruptGenerator* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_INTERRUPT_GENERATOR* is available.

Since

Version 1.5

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.3 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_Enter)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RegionRef region)

Callback for the Enter event record.

An enter record indicates that the program enters a code region.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.4 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-Leave)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RegionRef region)

Callback for the Leave event record.

A leave record indicates that the program leaves a code region.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.5 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MeasurementOnOff)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_MeasurementMode measurementMode)

Callback for the MeasurementOnOff event record.

This event signals where the measurement system turned measurement on or off.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.

userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
measure-mentMode	Is the measurement turned on (*OTF2_MEASUREMENT_ON*) or off (*OTF2_MEASUREMENT_OFF*)?

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.6 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-Metric)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type ∗typeIDs, const OTF2_MetricValue ∗metricValues)

Callback for the Metric event record.

A metric event is always stored at the location that recorded the metric. A metric event can reference a metric class or metric instance. Therefore, metric classes and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
metric	Could be a metric class or a metric instance. References a *MetricClass*, or a *MetricInstance* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_METRIC* is available.
numberOf-Metrics	Number of metrics with in the set.
typeIDs	List of metric types. These types must match that of the corresponding *MetricMember* definitions.
metricVal-ues	List of metric values.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.7 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for the MpiCollectiveBegin event record.

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI_-GATHER, MPI_SCATTER etc.).

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.8 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

Callback for the MpiCollectiveEnd event record.

A MpiCollectiveEnd record marks the end of an MPI collective operation (MPI_-GATHER, MPI_SCATTER etc.). It keeps the necessary information for this event: type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
collec-tiveOp	Determines which collective operation it is.
communi-cator	Communicator References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
root	MPI rank of root in `communicator`.
sizeSent	Size of the sent message.
sizeRe-ceived	Size of the received message.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.9 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiIrecv)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIrecv event record.

A MpiIrecv record indicates that a MPI message was received (MPI_IRECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

sender	MPI rank of sender in `communicator`.
communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.10 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_MpiIrecvRequest)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

Callback for the MpiIrecvRequest event record.

Signals the request of an receive, which can be completed later.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
requestID	ID of the requested receive

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.11 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiIsend)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIsend event record.

A MpiIsend record indicates that a MPI message send process was initiated (MPI_-ISEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.12 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiIsendComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

Callback for the MpiIsendComplete event record.

Signals the completion of non-blocking send request.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.13 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiRecv)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiRecv event record.

A MpiRecv record indicates that a MPI message was received (MPI_RECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
sender	MPI rank of sender in `communicator`.
communicator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.14 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiRequestCancelled)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

Callback for the MpiRequestCancelled event record.

This events appears if the program canceled a request.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.15 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiRequestTest)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t requestID)

Callback for the MpiRequestTest event record.

This events appears if the program tests if a request has already completed but the test failed.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
requestID	ID of the related request

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.16 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-MpiSend)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiSend event record.

A MpiSend record indicates that a MPI message send process was initiated (MPI_-SEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.17 typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_-OmpAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the OmpAcquireLock event record.

An OmpAcquireLock record marks that a thread acquires an OpenMP lock.

This event record is superseded by the *ThreadAcquireLock* event record and should not be used when the *ThreadAcquireLock* event record is in use.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.18 typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_-OmpFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t numberOfRequestedThreads)

Callback for the OmpFork event record.

An OmpFork record marks that an OpenMP Thread forks a thread team.

This event record is superseded by the *ThreadFork* event record and should not be used when the *ThreadFork* event record is in use.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
numberOfRequestedThreads	Requested size of the team.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.19 typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_OmpJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList)

Callback for the OmpJoin event record.

An OmpJoin record marks that a team of threads is joint and only the master thread continues execution.

This event record is superseded by the *ThreadJoin* event record and should not be used when the *ThreadJoin* event record is in use.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.20 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-OmpReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the OmpReleaseLock event record.

An OmpReleaseLock record marks that a thread releases an OpenMP lock.

This event record is superseded by the *ThreadReleaseLock* event record and should not be used when the *ThreadReleaseLock* event record is in use.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.21 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-OmpTaskComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

Callback for the OmpTaskComplete event record.

An OmpTaskComplete record indicates that the execution of an OpenMP task has finished.

This event record is superseded by the *ThreadTaskComplete* event record and should not be used when the *ThreadTaskComplete* event record is in use.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
taskID	Identifier of the completed task instance.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.22 typedef OTF2_CallbackCode($*$ OTF2_GlobalEvtReaderCallback_-OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void $*$userData, OTF2_AttributeList $*$attributeList, uint64_t taskID)

Callback for the OmpTaskCreate event record.

An OmpTaskCreate record marks that an OpenMP Task was/will be created in the current region.

This event record is superseded by the *ThreadTaskCreate* event record and should not be used when the *ThreadTaskCreate* event record is in use.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
taskID	Identifier of the newly created task instance.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.23 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-OmpTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t taskID)

Callback for the OmpTaskSwitch event record.

An OmpTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

This event record is superseded by the *ThreadTaskSwitch* event record and should not be used when the *ThreadTaskSwitch* event record is in use.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
taskID	Identifier of the now active task instance.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.24 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, int64_t value)

Callback for the ParameterInt event record.

A ParameterInt record marks that in the current region, the specified integer parameter has the specified value.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.25 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ParameterString)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, OTF2_StringRef string)

Callback for the ParameterString event record.

A ParameterString record marks that in the current region, the specified string parameter has the specified value.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
string	Value: Handle of a string definition References a *String* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_STRING* is available.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.26 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ParameterUnsignedInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_ParameterRef parameter, uint64_t value)

Callback for the ParameterUnsignedInt event record.

A ParameterUnsignedInt record marks that in the current region, the specified unsigned integer parameter has the specified value.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.0

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.27 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

Callback for the RmaAcquireLock event record.

An RmaAcquireLock record denotes the time a lock was acquired by the process.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.28 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaAtomic)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)

Callback for the RmaAtomic event record.

An RmaAtomic record denotes the time a atomic operation was issued.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.

type	Type of atomic operation.
bytesSent	Bytes sent to target.
bytesRe-ceived	Bytes received from target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.29 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for the RmaCollectiveBegin event record.

An RmaCollectiveBegin record denotes the beginnig of a collective RMA operation.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.30 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)

Callback for the RmaCollectiveEnd event record.

An RmaCollectiveEnd record denotes the end of a collective RMA operation.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
collectiveOp	Determines which collective operation it is.
syncLevel	Synchronization level of this collective operation.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
root	Root process for this operation.
bytesSent	Bytes sent in operation.
bytesReceived	Bytes receives in operation.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.31 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaGet)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)

Callback for the RmaGet event record.

An RmaGet record denotes the time a get operation was issued.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.
bytes	Bytes received from target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.32 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaGroupSync)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)

Callback for the RmaGroupSync event record.

An RmaGroupSync record denotes the synchronization with a subgroup of processes on a window.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
syncLevel	Synchronization level of this collective operation.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
group	Group of remote processes involved in synchronization. References a *Group* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_GROUP* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.33 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaOpCompleteBlocking)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpCompleteBlocking event record.

An RmaOpCompleteBlocking record denotes the local completion of a blocking RMA operation.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.34 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaOpCompleteNonBlocking)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpCompleteNonBlocking event record.

An RmaOpCompleteNonBlocking record denotes the local completion of a non-blocking RMA operation.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.35 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaOpCompleteRemote)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpCompleteRemote event record.

An RmaOpCompleteRemote record denotes the remote completion of an RMA operation.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.36 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaOpTest)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpTest event record.

An RmaOpTest record denotes that a non-blocking RMA operation has been tested for completion unsuccessfully.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
matchingId	ID used for matching the corresponding RMA operation record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.37 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaPut)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)

Callback for the RmaPut event record.

An RmaPut record denotes the time a put operation was issued.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the target process.
bytes	Bytes sent to target.
matchingId	ID used for matching the corresponding completion record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.38 typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_-RmaReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)

Callback for the RmaReleaseLock event record.

An RmaReleaseLock record denotes the time the lock was released.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock released, if multiple locks are defined on a window.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.39 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaRequestLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

Callback for the RmaRequestLock event record.

An RmaRequestLock record denotes the time a lock was requested and with it the earliest time it could have been granted. It is used to mark (possibly) non-blocking lock request, as defined by the MPI standard.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.40 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaSync)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaSyncType syncType)

Callback for the RmaSync event record.

An RmaSync record denotes the direct synchronization with a possibly remote process.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
syncType	Type of synchronization.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.41 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaTryLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

Callback for the RmaTryLock event record.

An RmaTryLock record denotes the time of an unsuccessful attempt to acquire the lock.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.

userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.
remote	Rank of the locked remote process.
lockId	ID of the lock acquired, if multiple locks are defined on a window.
lockType	Type of lock acquired.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.42 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaWaitChange)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

Callback for the RmaWaitChange event record.

An RmaWaitChange record denotes the change of a window that was waited for.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window used for this operation. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.43 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaWinCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

Callback for the RmaWinCreate event record.

An RmaWinCreate record denotes the creation of an RMA window.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window created. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.44 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-RmaWinDestroy)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_RmaWinRef win)

Callback for the RmaWinDestroy event record.

An RmaWinDestroy record denotes the destruction of an RMA window.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
win	ID of the window destructed. References a *RmaWin* definition and will be mapped to the global definition if a mapping table of type *OTF2_-MAPPING_RMA_WIN* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.45 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ThreadAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the ThreadAcquireLock event record.

An ThreadAcquireLock record marks that a thread acquires an lock.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
model	The threading paradigm this event took place.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.46 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ThreadBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

Callback for the ThreadBegin event record.

Marks the begin of a thread created by another thread.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadCreate* event does have the same number.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.47 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

Callback for the ThreadCreate event record.

The location created successfully a new thread.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadBegin* event does have the same number.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.48 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

Callback for the ThreadEnd event record.

Marks the end of a thread.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadWait* event does have the same number. *OTF2_UNDEFINED_UINT64* in case no corresponding *ThreadWait* event exists.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.49 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)

Callback for the ThreadFork event record.

An ThreadFork record marks that an thread forks a thread team.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
model	The threading paradigm this event took place.
numberOfRequestedThreads	Requested size of the team.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.50 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model)

Callback for the ThreadJoin event record.

An ThreadJoin record marks that a team of threads is joint and only the master thread continues execution.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
model	The threading paradigm this event took place.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.51 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ThreadReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the ThreadReleaseLock event record.

An ThreadReleaseLock record marks that a thread releases an lock.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
model	The threading paradigm this event took place.
lockID	ID of the lock.
acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.52 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ThreadTaskComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

Callback for the ThreadTaskComplete event record.

An ThreadTaskComplete record indicates that the execution of an OpenMP task has finished.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
creatingThread	Creating thread of this task.
generationNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.53 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

Callback for the ThreadTaskCreate event record.

An ThreadTaskCreate record marks that an task in was/will be created and will be processed by the specified thread team.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.

creatingThread	Creating thread of this task.
generationNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.54 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)

Callback for the ThreadTaskSwitch event record.

An ThreadTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
creatingThread	Creating thread of this task.
generationNumber	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.55 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ThreadTeamBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam)

Callback for the ThreadTeamBegin event record.

The current location enters the specified thread team.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.56 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-ThreadTeamEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadTeam)

Callback for the ThreadTeamEnd event record.

The current location leaves the specified thread team.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.

userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadTeam	Thread team References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.57 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_ThreadWait)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)

Callback for the ThreadWait event record.

The location waits for the completion of another thread.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.
threadContingent	The thread contingent. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
sequenceCount	A `threadContingent` unique number. The corresponding *ThreadEnd* event does have the same number.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.2.58 typedef OTF2_CallbackCode(∗ OTF2_GlobalEvtReaderCallback_-Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for an unknown event record.

Parameters

locationID	The location where this event happened.
time	The time when this event happened.
userData	User data as set by *OTF2_Reader_RegisterGlobalEvtCallbacks* or *OTF2_GlobalEvtReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.21.3 Function Documentation

E.21.3.1 void OTF2_GlobalEvtReaderCallbacks_Clear (OTF2_-GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*)

Clears a struct for the global event callbacks.

Parameters

glob-alEvtRead-erCallbacks	Handle to a struct previously allocated with *OTF2_-GlobalEvtReaderCallbacks_New*.

E.21.3.2 void OTF2_GlobalEvtReaderCallbacks_Delete (OTF2_-GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*)

Deallocates a struct for the global event callbacks.

Parameters

glob-alEvtRead-erCallbacks	Handle to a struct previously allocated with *OTF2_-GlobalEvtReaderCallbacks_New*.

E.21.3.3 OTF2_GlobalEvtReaderCallbacks∗ OTF2_GlobalEvtReaderCallbacks_New (void)

Allocates a new struct for the event callbacks.

Returns

A newly allocated struct of type *OTF2_GlobalEvtReaderCallbacks*.

E.21.3.4 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetBufferFlushCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_BufferFlush *bufferFlushCallback*)

Registers the callback for the BufferFlush event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
bufferFlushCallback	Function which should be called for all *BufferFlush* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.5 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetCallingContextSampleCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_CallingContextSample *callingContextSampleCallback*)

Registers the callback for the CallingContextSample event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
callingContextSampleCallback	Function which should be called for all *CallingContextSample* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.6 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetEnterCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
enterCallback	Function which should be called for all *Enter* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.7 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_Leave *leaveCallback*)

Registers the callback for the Leave event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
leaveCallback	Function which should be called for all *Leave* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.8 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMeasurementOnOffCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MeasurementOnOff *measurementOnOffCallback*)

Registers the callback for the MeasurementOnOff event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.

measurementOnOffCallback	Function which should be called for all *MeasurementOnOff* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.9 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMetricCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_Metric *metricCallback*)

Registers the callback for the Metric event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
metricCallback	Function which should be called for all *Metric* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.10 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetMpiCollectiveBeginCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks,* OTF2_GlobalEvtReaderCallback_-MpiCollectiveBegin *mpiCollectiveBeginCallback*)

Registers the callback for the MpiCollectiveBegin event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
mpiCollectiveBeginCallback	Function which should be called for all *MpiCollectiveBegin* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.11 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetMpiCollectiveEndCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks,* OTF2_GlobalEvtReaderCallback_-MpiCollectiveEnd *mpiCollectiveEndCallback*)

Registers the callback for the MpiCollectiveEnd event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
mpiCollectiveEndCallback	Function which should be called for all *MpiCollectiveEnd* definitions.

globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.12 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MpiIrecv *mpiIrecvCallback*)

Registers the callback for the MpiIrecv event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
mpiIrecvCallback	Function which should be called for all *MpiIrecv* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.13 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MpiIrecvRequest *mpiIrecvRequestCallback*)

Registers the callback for the MpiIrecvRequest event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
mpiIrecvRequestCallback	Function which should be called for all *MpiIrecvRequest* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.14 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MpiIsend *mpiIsendCallback*)

Registers the callback for the MpiIsend event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
mpiIsendCallback	Function which should be called for all *MpiIsend* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.15 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MpiIsendComplete *mpiIsendCompleteCallback*)

Registers the callback for the MpiIsendComplete event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
mpiIsendCompleteCallback	Function which should be called for all *MpiIsendComplete* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.16 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiRecvCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MpiRecv *mpiRecvCallback*)

Registers the callback for the MpiRecv event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
mpiRecv-Callback	Function which should be called for all *MpiRecv* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.17 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetMpiRequestCancelledCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-MpiRequestCancelled *mpiRequestCancelledCallback*)

Registers the callback for the MpiRequestCancelled event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
mpiRe-questCan-celledCall-back	Function which should be called for all *MpiRequestCancelled* definitions.

glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.18 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiRequestTestCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MpiRequestTest *mpiRequestTestCallback*)

Registers the callback for the MpiRequestTest event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
mpiRe-questTest-Callback	Function which should be called for all *MpiRequestTest* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.19 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiSendCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MpiSend *mpiSendCallback*)

Registers the callback for the MpiSend event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
mpiSendCallback	Function which should be called for all *MpiSend* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.20 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpAcquireLockCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_OmpAcquireLock *ompAcquireLockCallback*)

Registers the callback for the OmpAcquireLock event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
ompAcquireLockCallback	Function which should be called for all *OmpAcquireLock* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.21 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpForkCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
ompForkCallback	Function which should be called for all *OmpFork* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.22 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpJoinCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_OmpJoin *ompJoinCallback*)

Registers the callback for the OmpJoin event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
ompJoinCallback	Function which should be called for all *OmpJoin* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.23 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpReleaseLockCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_OmpReleaseLock *ompReleaseLockCallback*)

Registers the callback for the OmpReleaseLock event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
ompReleaseLockCallback	Function which should be called for all *OmpReleaseLock* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.24 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCompleteCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks,* OTF2_GlobalEvtReaderCallback_OmpTaskComplete *ompTaskCompleteCallback*)

Registers the callback for the OmpTaskComplete event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
ompTaskCompleteCallback	Function which should be called for all *OmpTaskComplete* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.25 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCreateCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks,* OTF2_GlobalEvtReaderCallback_OmpTaskCreate *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
ompTaskCreateCallback	Function which should be called for all *OmpTaskCreate* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.26 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskSwitchCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_OmpTaskSwitch *ompTaskSwitchCallback*)

Registers the callback for the OmpTaskSwitch event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
ompTaskSwitchCallback	Function which should be called for all *OmpTaskSwitch* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.27 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterIntCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks,* OTF2_GlobalEvtReaderCallback_ParameterInt *parameterIntCallback*)

Registers the callback for the ParameterInt event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
parameterIntCallback	Function which should be called for all *ParameterInt* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.28 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterStringCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks,* OTF2_GlobalEvtReaderCallback_ParameterString *parameterStringCallback*)

Registers the callback for the ParameterString event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
parameterStringCallback	Function which should be called for all *ParameterString* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.29 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterUnsignedIntCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ParameterUnsignedInt *parameterUnsignedIntCallback*)

Registers the callback for the ParameterUnsignedInt event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
parameterUnsignedIntCallback	Function which should be called for all *ParameterUnsignedInt* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.30 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaAcquireLockCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaAcquireLock *rmaAcquireLockCallback*)

Registers the callback for the RmaAcquireLock event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaAcquireLockCallback	Function which should be called for all *RmaAcquireLock* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.31 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaAtomicCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaAtomic *rmaAtomicCallback*)

Registers the callback for the RmaAtomic event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaAtomicCallback	Function which should be called for all *RmaAtomic* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.32 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveBeginCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaCollectiveBegin *rmaCollectiveBeginCallback*)

Registers the callback for the RmaCollectiveBegin event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaCollectiveBeginCallback	Function which should be called for all *RmaCollectiveBegin* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.33 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveEndCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaCollectiveEnd *rmaCollectiveEndCallback*)

Registers the callback for the RmaCollectiveEnd event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaCollectiveEndCallback	Function which should be called for all *RmaCollectiveEnd* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.34 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaGetCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaGet *rmaGetCallback*)

Registers the callback for the RmaGet event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
rmaGet-Callback	Function which should be called for all *RmaGet* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.35 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaGroupSyncCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaGroupSync *rmaGroupSyncCallback*)

Registers the callback for the RmaGroupSync event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
rmaGroup-SyncCall-back	Function which should be called for all *RmaGroupSync* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.36 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteBlockingCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaOpCompleteBlocking *rmaOpCompleteBlockingCallback*)

Registers the callback for the RmaOpCompleteBlocking event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaOpCompleteBlockingCallback	Function which should be called for all *RmaOpCompleteBlocking* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.37 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaOpCompleteNonBlocking *rmaOpCompleteNonBlockingCallback*)

Registers the callback for the RmaOpCompleteNonBlocking event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaOpCompleteNonBlockingCallback	Function which should be called for all *RmaOpCompleteNonBlocking* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.38 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteRemoteCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaOpCompleteRemote *rmaOpCompleteRemoteCallback*)

Registers the callback for the RmaOpCompleteRemote event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaOpCompleteRemoteCallback	Function which should be called for all *RmaOpCompleteRemote* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.39 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpTestCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaOpTest *rmaOpTestCallback*)

Registers the callback for the RmaOpTest event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaOpTestCallback	Function which should be called for all *RmaOpTest* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.40 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaPutCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaPut *rmaPutCallback*)

Registers the callback for the RmaPut event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
rmaPut-Callback	Function which should be called for all *RmaPut* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.41 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetRmaReleaseLockCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks,* OTF2_GlobalEvtReaderCallback_-RmaReleaseLock *rmaReleaseLockCallback*)

Registers the callback for the RmaReleaseLock event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
rmaRe-leaseLock-Callback	Function which should be called for all *RmaReleaseLock* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.42 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaRequestLockCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaRequestLock *rmaRequestLockCallback*)

Registers the callback for the RmaRequestLock event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaRequestLockCallback	Function which should be called for all *RmaRequestLock* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.43 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaSyncCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaSync *rmaSyncCallback*)

Registers the callback for the RmaSync event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
rmaSync-Callback	Function which should be called for all *RmaSync* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.44 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaTryLockCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaTryLock *rmaTryLockCallback*)

Registers the callback for the RmaTryLock event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
rmaTry-LockCall-back	Function which should be called for all *RmaTryLock* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.45 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaWaitChangeCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaWaitChange *rmaWaitChangeCallback*)

Registers the callback for the RmaWaitChange event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
rmaWaitChangeCallback	Function which should be called for all *RmaWaitChange* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.46 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaWinCreateCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaWinCreate *rmaWinCreateCallback*)

Registers the callback for the RmaWinCreate event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.

rmaWin-CreateCall-back	Function which should be called for all *RmaWinCreate* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.47 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaWinDestroyCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_RmaWinDestroy *rmaWinDestroyCallback*)

Registers the callback for the RmaWinDestroy event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
rmaWinDe-stroyCall-back	Function which should be called for all *RmaWinDestroy* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.48 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadAcquireLockCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadAcquireLock *threadAcquireLockCallback*)

Registers the callback for the ThreadAcquireLock event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
threadAcquireLockCallback	Function which should be called for all *ThreadAcquireLock* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.49 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadBeginCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadBegin *threadBeginCallback*)

Registers the callback for the ThreadBegin event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.

threadBeginCallback	Function which should be called for all *ThreadBegin* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.50 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadCreateCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadCreate *threadCreateCallback*)

Registers the callback for the ThreadCreate event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
threadCreateCallback	Function which should be called for all *ThreadCreate* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.51 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadEndCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadEnd *threadEndCallback*)

Registers the callback for the ThreadEnd event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
threadEnd-Callback	Function which should be called for all *ThreadEnd* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.3

Returns

OTF2_SUCCESS	if successful
OTF2_ERROR_INVALID_ARGUMENT	for an invalid `defReaderCallbacks` argument

E.21.3.52 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadForkCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadFork *threadForkCallback*)

Registers the callback for the ThreadFork event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
threadFork-Callback	Function which should be called for all *ThreadFork* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.53 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadJoinCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadJoin *threadJoinCallback*)

Registers the callback for the ThreadJoin event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
threadJoinCallback	Function which should be called for all *ThreadJoin* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.54 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadReleaseLockCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadReleaseLock *threadReleaseLockCallback*)

Registers the callback for the ThreadReleaseLock event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
thread-Release-LockCall-back	Function which should be called for all *ThreadReleaseLock* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.55 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetThreadTaskCompleteCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-ThreadTaskComplete *threadTaskCompleteCallback*)

Registers the callback for the ThreadTaskComplete event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
thread-TaskCom-pleteCall-back	Function which should be called for all *ThreadTaskComplete* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.56 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTaskCreateCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadTaskCreate *threadTaskCreateCallback*)

Registers the callback for the ThreadTaskCreate event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
threadTaskCreateCallback	Function which should be called for all *ThreadTaskCreate* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.57 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTaskSwitchCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadTaskSwitch *threadTaskSwitchCallback*)

Registers the callback for the ThreadTaskSwitch event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
threadTaskSwitchCallback	Function which should be called for all *ThreadTaskSwitch* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.58 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTeamBeginCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadTeamBegin *threadTeamBeginCallback*)

Registers the callback for the ThreadTeamBegin event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
threadTeamBeginCallback	Function which should be called for all *ThreadTeamBegin* definitions.

glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.59 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadTeamEndCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadTeamEnd *threadTeamEndCallback*)

Registers the callback for the ThreadTeamEnd event.

Parameters

glob-alEvtRead-erCallbacks	Struct for all callbacks.
threadTea-mEndCall-back	Function which should be called for all *ThreadTeamEnd* definitions.
glob-alEvtRead-erCallbacks	Struct for all callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.60 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetThreadWaitCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ThreadWait *threadWaitCallback*)

Registers the callback for the ThreadWait event.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
threadWaitCallback	Function which should be called for all *ThreadWait* definitions.
globalEvtReaderCallbacks	Struct for all callbacks.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21.3.61 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetUnknownCallback (OTF2_GlobalEvtReaderCallbacks ∗ *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_Unknown *unknownCallback*)

Registers the callback for unknown events.

Parameters

globalEvtReaderCallbacks	Struct for all callbacks.
unknownCallback	Function which should be called for all unknown events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22 otf2/OTF2_GlobalSnapReader.h File Reference

This is the global snapshot event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_SnapReader.h>
#include <otf2/OTF2_GlobalSnapReaderCallbacks.h>
```

Functions

- OTF2_ErrorCode OTF2_GlobalSnapReader_ReadSnapshots (OTF2_GlobalSnapReader ∗reader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  Reads the given number of records from the global snap event reader.
- OTF2_ErrorCode OTF2_GlobalSnapReader_SetCallbacks (OTF2_GlobalSnapReader ∗reader, const OTF2_GlobalSnapReaderCallbacks ∗callbacks, void ∗userData)

  Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

E.22.1 Detailed Description

This is the global snapshot event reader.

Since

Version 1.2

Used to read from multiple local snap event readers, and provide them in a timely ordered sequence.

E.22.2 Function Documentation

E.22.2.1 OTF2_ErrorCode OTF2_GlobalSnapReader_ReadSnapshots (OTF2_GlobalSnapReader ∗ *reader,* uint64_t *recordsToRead,* uint64_t ∗ *recordsRead*)

Reads the given number of records from the global snap event reader.

Parameters

	reader	The records of this reader will be read when the function is issued.
	record-sToRead	This variable tells the reader how much records it has to read.
`out`	*record-sRead*	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given record-sToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finnished his job by checking recordsRead < recordsToRead.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.22.2.2 OTF2_ErrorCode OTF2_GlobalSnapReader_SetCallbacks (OTF2_GlobalSnapReader ∗ *reader,* const OTF2_-GlobalSnapReaderCallbacks ∗ *callbacks,* void ∗ *userData*)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

reader	Reader object which reads the snap events from its buffer.
callbacks	Struct which holds a function pointer for each record type. *OTF2_-GlobalSnapReaderCallbacks_New*.
userData	Data passed as argument *userData* to the record callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.23 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

This defines the callbacks for the global snap reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_Enter)(OTF2_-LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_RegionRef region)

  Callback for the Enter snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MeasurementOnOff)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-MeasurementMode measurementMode)

  Callback for the MeasurementOnOff snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_Metric)(OTF2_-LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type ∗typeIDs, const OTF2_-MetricValue ∗metricValues)

  Callback for the Metric snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime)

  Callback for the MpiCollectiveBegin snap record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MpiCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

  Callback for the MpiCollectiveEnd snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MpiIrecv)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_-t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t ms-gLength, uint64_t requestID)

  Callback for the MpiIrecv snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MpiIrecvRequest)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_-t requestID)

  Callback for the MpiIrecvRequest snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MpiIsend)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_-t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t ms-gLength, uint64_t requestID)

  Callback for the MpiIsend snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MpiIsendComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_-t requestID)

  Callback for the MpiIsendComplete snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MpiRecv)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_-t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t ms-gLength)

  Callback for the MpiRecv snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MpiSend)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_-t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t ms-gLength)

  Callback for the MpiSend snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_OmpAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData,

OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_-t lockID, uint32_t acquisitionOrder)

Callback for the OmpAcquireLock snap record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_OmpFork)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_-t numberOfRequestedThreads)

  Callback for the OmpFork snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_-t taskID)

  Callback for the OmpTaskCreate snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_OmpTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_-t taskID)

  Callback for the OmpTaskSwitch snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-ParameterRef parameter, int64_t value)

  Callback for the ParameterInt snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_ParameterString)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-ParameterRef parameter, OTF2_StringRef string)

  Callback for the ParameterString snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-ParameterRef parameter, uint64_t value)

  Callback for the ParameterUnsignedInt snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_SnapshotEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t contReadPos)

  Callback for the SnapshotEnd snap record.
- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_SnapshotStart)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t numberOfRecords)

  Callback for the SnapshotStart snap record.

- typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList)

    Callback for an unknown snap record.
- typedef struct OTF2_GlobalSnapReaderCallbacks_struct OTF2_GlobalSnapReaderCallbacks

    Opaque struct which holdes all snap record callbacks.

Functions

- void OTF2_GlobalSnapReaderCallbacks_Clear (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks)

    Clears a struct for the global snap callbacks.
- void OTF2_GlobalSnapReaderCallbacks_Delete (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks)

    Deallocates a struct for the global snap callbacks.
- OTF2_GlobalSnapReaderCallbacks ∗ OTF2_GlobalSnapReaderCallbacks_-New (void)

    Allocates a new struct for the snap callbacks.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetEnterCallback (OTF2_-GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_-Enter enterCallback)

    Registers the callback for the Enter snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMeasurementOnOffCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MeasurementOnOff measurementOnOffCallback)

    Registers the callback for the MeasurementOnOff snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMetricCallback (OTF2_-GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_-Metric metricCallback)

    Registers the callback for the Metric snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MpiCollectiveBegin mpiCollectiveBeginCallback)

    Registers the callback for the MpiCollectiveBegin snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MpiCollectiveEnd mpiCollectiveEndCallback)

Registers the callback for the MpiCollectiveEnd snap.

- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MpiIrecv mpiIrecvCallback)

  Registers the callback for the MpiIrecv snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MpiIrecvRequest mpiIrecvRequestCallback)

  Registers the callback for the MpiIrecvRequest snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIsendCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MpiIsend mpiIsendCallback)

  Registers the callback for the MpiIsend snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MpiIsendComplete mpiIsendCompleteCallback)

  Registers the callback for the MpiIsendComplete snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiRecvCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MpiRecv mpiRecvCallback)

  Registers the callback for the MpiRecv snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiSendCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_MpiSend mpiSendCallback)

  Registers the callback for the MpiSend snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetOmpAcquireLockCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_OmpAcquireLock ompAcquireLockCallback)

  Registers the callback for the OmpAcquireLock snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetOmpForkCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_OmpFork ompForkCallback)

  Registers the callback for the OmpFork snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetOmpTaskCreateCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_OmpTaskCreate ompTaskCreateCallback)

  Registers the callback for the OmpTaskCreate snap.
- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetOmpTaskSwitchCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_OmpTaskSwitch ompTaskSwitchCallback)

Registers the callback for the OmpTaskSwitch snap.

- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetParameterIntCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_ParameterInt parameterIntCallback)

  Registers the callback for the ParameterInt snap.

- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetParameterStringCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_ParameterString parameterStringCallback)

  Registers the callback for the ParameterString snap.

- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetParameterUnsignedIntCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_ParameterUnsignedInt parameterUnsignedIntCall-back)

  Registers the callback for the ParameterUnsignedInt snap.

- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetSnapshotEndCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_SnapshotEnd snapshotEndCallback)

  Registers the callback for the SnapshotEnd snap.

- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetSnapshotStartCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_SnapshotStart snapshotStartCallback)

  Registers the callback for the SnapshotStart snap.

- OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetUnknownCallback (OTF2_GlobalSnapReaderCallbacks ∗globalSnapReaderCallbacks, OTF2_-GlobalSnapReaderCallback_Unknown unknownCallback)

  Registers the callback for unknown snaps.

E.23.1 Detailed Description

This defines the callbacks for the global snap reader.

Source Template:

templates/OTF2_GlobalSnapReaderCallbacks.tmpl.h

E.23.2 Typedef Documentation

E.23.2.1 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_Enter)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_RegionRef region)

Callback for the Enter snap record.

This record exists for each *Enter* event where the corresponding *Leave* event did not occur before the snapshot.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.2 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_MeasurementOnOff)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_MeasurementMode measurementMode)

Callback for the MeasurementOnOff snap record.

The last occurrence of an *MeasurementOnOff* event of this location, if any.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
measure-mentMode	Is the measurement turned on (*OTF2_MEASUREMENT_ON*) or off (*OTF2_MEASUREMENT_OFF*)?

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.3 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-Metric)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type ∗typeIDs, const OTF2_MetricValue ∗metricValues)

Callback for the Metric snap record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

As an exception for metric classes where the metric mode detontes an *OTF2_-METRIC_VALUE_RELATIVE* mode the value indicates the accumulation of all previous metric values recorded.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
metric	Could be a metric class or a metric instance. References a *MetricClass*, or a *MetricInstance* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_METRIC* is available.

numberOf-Metrics	Number of metrics with in the set.
typeIDs	List of metric types. These types must match that of the corresponding *MetricMember* definitions.
metricVal-ues	List of metric values.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.4 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-MpiCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime)

Callback for the MpiCollectiveBegin snap record.

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.5 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-MpiCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

Callback for the MpiCollectiveEnd snap record.

Indicates that this location completed a collective operation localy but not all of the participating locations completed the operation yet. The corresponding *MpiCollectiveBeginSnap* record is still in the snapshot though.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
collec-tiveOp	Determines which collective operation it is.
communi-cator	Communicator References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
root	MPI rank of root in `communicator`.
sizeSent	Size of the sent message.
sizeRe-ceived	Size of the received message.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.6 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-MpiIrecv)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIrecv snap record.

This record exists for each *MpiIrecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
sender	MPI rank of sender in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.7 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-MpiIrecvRequest)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)

Callback for the MpiIrecvRequest snap record.

This record exists for each *MpiIrecvRequest* event where an corresponding *Mpi-Irecv* or *MpiRequestCancelled* event did not occur on this location before the snap-shot. Or the corresponding *MpiIrecv* did occurred (the *MpiIrecvSnap* record exists in the snapshot) but the matching receive message event did not occur on the re-mote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
requestID	ID of the requested receive

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.8 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-MpiIsend)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIsend snap record.

This record exists for each *MpiIsend* event where an corresponding *MpiIsendCom-plete* or *MpiRequestCancelled* event did not occur on this location before the snap-shot. Or the corresponding *MpiIsendComplete* did occurred (the *MpiIsendCom-pleteSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. (This could either be an*MpiRecv* or an *MpiIrecv* event.)

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happened.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.9 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-MpiIsendComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)

Callback for the MpiIsendComplete snap record.

This record exists for each *MpiIsend* event where the corresponding *MpiIsendComplete* event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an *MpiRecv* or an *MpiIrecv* event.) .

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.

userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.10 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-MpiRecv)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiRecv snap record.

This record exists for each *MpiRecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
sender	MPI rank of sender in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.11 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-MpiSend)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiSend snap record.

This record exists for each *MpiSend* event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event. Note that it may so, that a previous *MpiIsend* with the same envelope than this one is neither completed not canceled yet, thus the matching receive may already occurred, but the matching couldn't be done yet.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.12 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-OmpAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the OmpAcquireLock snap record.

This record exists for each *OmpAcquireLock* event where the corresponding *OmpReleaseLock* did not occurred before this snapshot yet.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
lockID	ID of the lock.
acquisi-tionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.13 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-OmpFork)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t numberOfRequestedThreads)

Callback for the OmpFork snap record.

This record exists for each *OmpFork* event where the corresponding *OmpJoin* did not occurred before this snapshot.

Parameters

locationID	The location where this snap happened.

time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
num-berOfRe-quest-edThreads	Requested size of the team.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.14 typedef OTF2_CallbackCode($*$ OTF2_GlobalSnapReaderCallback_-OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void $*$userData, OTF2_AttributeList $*$attributeList, OTF2_TimeStamp origEventTime, uint64_t taskID)

Callback for the OmpTaskCreate snap record.

This record exists for each *OmpTaskCreate* event where the corresponding *Omp-TaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
taskID	Identifier of the newly created task instance.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.15 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_OmpTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_t taskID)

Callback for the OmpTaskSwitch snap record.

This record exists for each *OmpTaskSwitch* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEventTime	The original time this event happended.
taskID	Identifier of the now active task instance.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.16 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, int64_t value)

Callback for the ParameterInt snap record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.17 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-ParameterString)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, OTF2_StringRef string)

Callback for the ParameterString snap record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.

string	Value: Handle of a string definition References a *String* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_STRING* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.18 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-ParameterUnsignedInt)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, uint64_t value)

Callback for the ParameterUnsignedInt snap record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.19 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-SnapshotEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t contReadPos)

Callback for the SnapshotEnd snap record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use *OTF2_EvtReader_Seek* with `contReadPos` as the position.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
contRead-Pos	Position to continue reading in the event trace.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.20 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_-SnapshotStart)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t numberOfRecords)

Callback for the SnapshotStart snap record.

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one *SnapshotStart* record and closes with one *SnapshotEnd* record. All snapshot records inbetween are ordered by the `origEventTime`, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

Parameters

locationID	The location where this snap happened.
time	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
numberOfRecord	Number of snapshot event records in this snapshot. Excluding the *SnapshotEnd* record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.21 typedef OTF2_CallbackCode(∗ OTF2_GlobalSnapReaderCallback_Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for an unknown snap record.

Parameters

locationID	The location where this snap happened.
snapTime	The time of this snapshot.
userData	User data as set by *OTF2_Reader_RegisterGlobalSnapCallbacks* or *OTF2_GlobalSnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.23.2.22 typedef struct OTF2_GlobalSnapReaderCallbacks_struct OTF2_GlobalSnapReaderCallbacks

Opaque struct which holdes all snap record callbacks.

Since

Version 1.2

E.23.3 Function Documentation

E.23.3.1 void OTF2_GlobalSnapReaderCallbacks_Clear (OTF2_- GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*)

Clears a struct for the global snap callbacks.

Parameters

globalSnapReaderCallbacks	Handle to a struct previously allocated with *OTF2_-GlobalSnapReaderCallbacks_New*.

Since

Version 1.2

E.23.3.2 void OTF2_GlobalSnapReaderCallbacks_Delete (OTF2_- GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*)

Deallocates a struct for the global snap callbacks.

Parameters

globalSnapReaderCallbacks	Handle to a struct previously allocated with *OTF2_-GlobalSnapReaderCallbacks_New*.

Since

Version 1.2

E.23.3.3 OTF2_GlobalSnapReaderCallbacks* OTF2_GlobalSnapReaderCallbacks_- New (void)

Allocates a new struct for the snap callbacks.

Since

Version 1.2

Returns

A newly allocated struct of type *OTF2_GlobalSnapReaderCallbacks*.

E.23.3.4 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetEnterCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
enterCallback	Function which should be called for all *Enter* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.5 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMeasurementOnOffCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_MeasurementOnOff *measurementOnOffCallback*)

Registers the callback for the MeasurementOnOff snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.

measure-mentOnOff-Callback	Function which should be called for all *MeasurementOnOff* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.6 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMetricCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_Metric *metricCallback*)

Registers the callback for the Metric snap.

Parameters

global-SnapRead-erCallbacks	Struct for all callbacks.
metricCall-back	Function which should be called for all *Metric* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.7 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin *mpiCollectiveBeginCallback*)

Registers the callback for the MpiCollectiveBegin snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
mpiCollectiveBeginCallback	Function which should be called for all *MpiCollectiveBegin* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.8 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_MpiCollectiveEnd *mpiCollectiveEndCallback*)

Registers the callback for the MpiCollectiveEnd snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
mpiCollectiveEndCallback	Function which should be called for all *MpiCollectiveEnd* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.9 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_MpiIrecv *mpiIrecvCallback*)

Registers the callback for the MpiIrecv snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
mpiIrecvCallback	Function which should be called for all *MpiIrecv* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.10 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_MpiIrecvRequest *mpiIrecvRequestCallback*)

Registers the callback for the MpiIrecvRequest snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
mpiIrecvRequestCallback	Function which should be called for all *MpiIrecvRequest* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.11 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIsendCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_MpiIsend *mpiIsendCallback*)

Registers the callback for the MpiIsend snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
mpiIsendCallback	Function which should be called for all *MpiIsend* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.12 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_MpiIsendComplete *mpiIsendCompleteCallback*)

Registers the callback for the MpiIsendComplete snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
mpiIsendCompleteCallback	Function which should be called for all *MpiIsendComplete* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.13 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiRecvCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_MpiRecv *mpiRecvCallback*)

Registers the callback for the MpiRecv snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
mpiRecvCallback	Function which should be called for all *MpiRecv* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.14 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiSendCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks,* OTF2_GlobalSnapReaderCallback_MpiSend *mpiSendCallback*)

Registers the callback for the MpiSend snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
mpiSendCallback	Function which should be called for all *MpiSend* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.15 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetOmpAcquireLockCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks,* OTF2_GlobalSnapReaderCallback_OmpAcquireLock *ompAcquireLockCallback*)

Registers the callback for the OmpAcquireLock snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
ompAcquireLockCallback	Function which should be called for all *OmpAcquireLock* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.16 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetOmpForkCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks,* OTF2_GlobalSnapReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
ompForkCallback	Function which should be called for all *OmpFork* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.17 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetOmpTaskCreateCallback (OTF2_GlobalSnapReaderCallbacks ∗ *globalSnapReaderCallbacks,* OTF2_GlobalSnapReaderCallback_OmpTaskCreate *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
ompTaskCreateCallback	Function which should be called for all *OmpTaskCreate* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.18 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetOmpTaskSwitchCallback (OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_OmpTaskSwitch *ompTaskSwitchCallback*)

Registers the callback for the OmpTaskSwitch snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
ompTaskSwitchCallback	Function which should be called for all *OmpTaskSwitch* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.19 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetParameterIntCallback (OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_ParameterInt *parameterIntCallback*)

Registers the callback for the ParameterInt snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
parameterIntCallback	Function which should be called for all *ParameterInt* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.20 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetParameterStringCallback (OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_ParameterString *parameterStringCallback*)

Registers the callback for the ParameterString snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
parameterStringCallback	Function which should be called for all *ParameterString* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.21 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetParameterUnsignedIntCallback (OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_ParameterUnsignedInt *parameterUnsignedIntCallback*)

Registers the callback for the ParameterUnsignedInt snap.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
parameterUnsignedIntCallback	Function which should be called for all *ParameterUnsignedInt* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.22 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetSnapshotEndCallback (OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_SnapshotEnd *snapshotEndCallback*)

Registers the callback for the SnapshotEnd snap.

Parameters

global-SnapRead-erCallbacks	Struct for all callbacks.
snap-shotEnd-Callback	Function which should be called for all *SnapshotEnd* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.23 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_-SetSnapshotStartCallback (OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_-SnapshotStart *snapshotStartCallback*)

Registers the callback for the SnapshotStart snap.

Parameters

global-SnapRead-erCallbacks	Struct for all callbacks.
snapshot-StartCall-back	Function which should be called for all *SnapshotStart* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.23.3.24 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetUnknownCallback (OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_Unknown *unknownCallback*)

Registers the callback for unknown snaps.

Parameters

globalSnapReaderCallbacks	Struct for all callbacks.
unknownCallback	Function which should be called for all unknown snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.24 otf2/OTF2_IdMap.h File Reference

Identifier mapping data structure, based on Scalasca's epk_idmap.h.

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <otf2/OTF2_ErrorCodes.h>
```

Typedefs

- typedef struct OTF2_IdMap_struct OTF2_IdMap
- typedef void(* OTF2_IdMap_TraverseCallback)(uint64_t localId, uint64_t globalId, void *userData)

  Function prototype for use in OTF2_IdMap_Traverse.
- typedef uint8_t OTF2_IdMapMode

Enumerations

- enum OTF2_IdMapMode_enum {
  OTF2_ID_MAP_DENSE,
  OTF2_ID_MAP_SPARSE }

Functions

- OTF2_ErrorCode OTF2_IdMap_AddIdPair (OTF2_IdMap ∗instance, uint64_t localId, uint64_t globalId)
- OTF2_ErrorCode OTF2_IdMap_Clear (OTF2_IdMap ∗instance)
- OTF2_IdMap ∗ OTF2_IdMap_Create (OTF2_IdMapMode mode, uint64_t capacity)
- OTF2_IdMap ∗ OTF2_IdMap_CreateFromUint32Array (uint64_t length, const uint32_t ∗mappings, bool optimizeSize)
- OTF2_IdMap ∗ OTF2_IdMap_CreateFromUint64Array (uint64_t length, const uint64_t ∗mappings, bool optimizeSize)
- void OTF2_IdMap_Free (OTF2_IdMap ∗instance)
- OTF2_ErrorCode OTF2_IdMap_GetGlobalId (const OTF2_IdMap ∗instance, uint64_t localId, uint64_t ∗globalId)
- OTF2_ErrorCode OTF2_IdMap_GetGlobalIdSave (const OTF2_IdMap ∗instance, uint64_t localId, uint64_t ∗globalId)
- OTF2_ErrorCode OTF2_IdMap_GetMode (const OTF2_IdMap ∗instance, OTF2_IdMapMode ∗mode)
- OTF2_ErrorCode OTF2_IdMap_GetSize (const OTF2_IdMap ∗instance, uint64_t ∗size)
- OTF2_ErrorCode OTF2_IdMap_Traverse (const OTF2_IdMap ∗instance, OTF2_IdMap_TraverseCallback callback, void ∗userData)

E.24.1 Detailed Description

Identifier mapping data structure, based on Scalasca's epk_idmap.h. This file provides type definitions and function prototypes for an identifier mapping data structure which is used to store mapping tables for converting local into global identifiers.

This mapping data structure can operate in two different modes (see OTF2_IdMapMode): A dense mapping can be used if the local identifiers are consecutively enumerated from 0 to N-1. In this case, only the global identifier are stored in the table at the corresponding entry, leading to compact storage and fast look-up. By contrast, if the local identifiers can consist of arbitrary numbers, a sparse mapping is necessary. Here, (localId, globalId) tuples are stored, which requires a more complicated look-up procedure.

E.24.2 Typedef Documentation

E.24.2.1 typedef struct OTF2_IdMap_struct OTF2_IdMap

Opaque data structure representing an ID mapping table.

E.24.2.2 typedef uint8_t OTF2_IdMapMode

Wrapper around enum OTF2_IdMapMode_enum, so that it is guaranteed that it is a uint8_t

E.24.3 Enumeration Type Documentation

E.24.3.1 enum OTF2_IdMapMode_enum

Enumeration type defining the two different modes of an identifier mapping table.

Enumerator:

OTF2_ID_MAP_DENSE	Dense mapping table
OTF2_ID_MAP_SPARSE	Sparse mapping table

E.24.4 Function Documentation

E.24.4.1 OTF2_ErrorCode OTF2_IdMap_AddIdPair (OTF2_IdMap ∗ *instance,* uint64_t *localId,* uint64_t *globalId*)

Adds the given mapping from *localId* to *globalId* to the mapping table *instance*. If the current capacity does not suffice, the data structure is automatically resized.

Note

If the mapping table operates in dense mapping mode, the parameter *localId* has to correspond to the next entry in the mapping table.

Parameters

instance	Object to add the mapping to.
localId	Local identifier.
globalId	Global identifier.

Returns

OTF2_SUCCESS, or error code.

E.24.4.2 OTF2_ErrorCode OTF2_IdMap_Clear (OTF2_IdMap ∗ *instance*)

Removes all entries in the given mapping table *instance*. It can be used, e.g., to reuse an mapping table object for new input data.

Parameters

instance	Object to remove entries from.

Returns

OTF2_SUCCESS, or error code.

E.24.4.3 OTF2_IdMap∗ OTF2_IdMap_Create (OTF2_IdMapMode *mode*, uint64_t *capacity*)

Creates and returns a new instance of OTF2_IdMap with the given *mode* and initial *capacity*. If the memory allocation request cannot be fulfilled, NULL is returned.

Parameters

mode	Mapping mode.
capacity	Initial capacity.

Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.

E.24.4.4 OTF2_IdMap∗ OTF2_IdMap_CreateFromUint32Array (uint64_t *length*, const uint32_t ∗ *mappings*, bool *optimizeSize*)

Creates and returns a new instance of OTF2_IdMap from the array given by *mappings*.

Same as *OTF2_IdMap_CreateFromUint64Array*, excpet from a uint32_t array.

Parameters

length	Number of elements in the *mappings* array.

mappings	Array with a dense mapping.
optimize-Size	Creates a SPARSE mapping, if the number of non- identities is less than half the array length.

Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.

E.24.4.5 OTF2_IdMap* OTF2_IdMap_CreateFromUint64Array (uint64_t *length,* const uint64_t * *mappings,* bool *optimizeSize*)

Creates and returns a new instance of OTF2_IdMap from the array given by *mappings*.

This creates always a DENSE mapping if *optimizeSize* is false. If it is true, it creates a SPARSE mapping, if the number of non-identitiy entries in the *mappings* array (ie. mapping[i] != i) is less than half the *length*.

Returns NULL when optimizeSize is true and the number of non-identitiy entries equals zero, ie. the given map is the identity map.

Parameters

length	Number of elements in the *mappings* array.
mappings	Array with a dense mapping.
optimize-Size	Creates a SPARSE mapping, if the number of non- identities is less than half the array length.

Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.

E.24.4.6 void OTF2_IdMap_Free (OTF2_IdMap * *instance*)

Destroys the given *instance* of OTF2_IdMap and releases the allocated memory.

Parameters

instance	Object to be freed

E.24.4.7 OTF2_ErrorCode OTF2_IdMap_GetGlobalId (const OTF2_IdMap * *instance,* uint64_t *localId,* uint64_t * *globalId*)

Maps the given *localId* to the global id ansd store it in the starge provide by *globalId*.

If the given *localId* is not in the mapping, sets globalId to the localId.

Parameters

	instance	Object to add the mapping to.
	localId	Local identifier.
out	*globalId*	Global identifier.

Returns

OTF2_SUCCESS, or error code.

E.24.4.8 OTF2_ErrorCode OTF2_IdMap_GetGlobalIdSave (const OTF2_IdMap * *instance,* uint64_t *localId,* uint64_t * *globalId*)

Maps the given *localId* to the global id ansd store it in the starge provide by *globalId*.

If the given *localId* is not in the mapping, returns *OTF2_ERROR_INDEX_OUT_OF_BOUNDS*.

Parameters

	instance	Object to add the mapping to.
	localId	Local identifier.
out	*globalId*	Global identifier.

Returns

OTF2_SUCCESS, or error code.

E.24.4.9 OTF2_ErrorCode OTF2_IdMap_GetMode (const OTF2_IdMap * *instance,* OTF2_IdMapMode * *mode*)

Returns the identifier mapping mode (dense/sparse) used for the given mapping table *instance*.

Parameters

	instance	Queried object.
out	*mode*	Identifier mapping mode.

Returns

OTF2_SUCCESS, or error code.

E.24.4.10 OTF2_ErrorCode OTF2_IdMap_GetSize (const OTF2_IdMap ∗ *instance*, uint64_t ∗ *size*)

Returns the actual number of entries stored in the given OTF2_IdMap *instance*.

Parameters

	instance	Queried object.
out	*size*	Number of entries.

Returns

OTF2_SUCCESS, or error code.

E.24.4.11 OTF2_ErrorCode OTF2_IdMap_Traverse (const OTF2_IdMap ∗ *instance*, OTF2_IdMap_TraverseCallback *callback*, void ∗ *userData*)

Calls for each mapping pair the callback *callback*.

Parameters

instance	Object to add the mapping to.
callback	Callback function which is called for eaach mapping pair.
userData	Data which is passed to the *callback* function.

Returns

OTF2_SUCCESS, or error code.

E.25 otf2/OTF2_Marker.h File Reference

This file provides types and enums for markers.

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

E.25 otf2/OTF2_Marker.h File Reference

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
```

Defines

- #define OTF2_UNDEFINED_MARKER ((OTF2_MarkerRef)OTF2_UNDEFINED_-UINT32)

  The invalid value for a reference to a DefMarker definition.

Typedefs

- typedef uint32_t OTF2_MarkerRef

  Type used to indicate a reference to a DefMarker definition.
- typedef uint8_t OTF2_MarkerScope

  Wrapper for enum OTF2_MarkerScope_enum.
- typedef uint8_t OTF2_MarkerSeverity

  Wrapper for enum OTF2_MarkerSeverity_enum.

Enumerations

- enum OTF2_MarkerScope_enum {
  OTF2_MARKER_SCOPE_GLOBAL,
  OTF2_MARKER_SCOPE_LOCATION,
  OTF2_MARKER_SCOPE_LOCATION_GROUP,
  OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE,
  OTF2_MARKER_SCOPE_GROUP,
  OTF2_MARKER_SCOPE_COMM }
- enum OTF2_MarkerSeverity_enum {
  OTF2_SEVERITY_NONE,
  OTF2_SEVERITY_LOW,
  OTF2_SEVERITY_MEDIUM,
  OTF2_SEVERITY_HIGH }

E.25.1 Detailed Description

This file provides types and enums for markers.

E.25.2 Enumeration Type Documentation

E.25.2.1 enum OTF2_MarkerScope_enum

A user marker does have a scope of it validity.

Enumerator:

OTF2_MARKER_SCOPE_GLOBAL The user marker has a global scope (could also be NONE).

OTF2_MARKER_SCOPE_LOCATION The user marker has a scope of a location.

OTF2_MARKER_SCOPE_LOCATION_GROUP The user marker has a scope of a location group.

OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE The user marker has a scope of a system tree.

OTF2_MARKER_SCOPE_GROUP The user marker has a scope of a group.

OTF2_MARKER_SCOPE_COMM The user marker has a scope of a communicator.

E.25.2.2 enum OTF2_MarkerSeverity_enum

A list of possible severities of user markers.

Enumerator:

OTF2_SEVERITY_NONE The marker does not have a severity.

OTF2_SEVERITY_LOW The marker has a low severity.

OTF2_SEVERITY_MEDIUM The marker has a medium severity.

OTF2_SEVERITY_HIGH The marker has a high severity.

E.26 otf2/OTF2_MarkerReader.h File Reference

This file provides all routines that read marker records.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Marker.h>
#include <otf2/OTF2_MarkerReaderCallbacks.h>
```

Functions

- OTF2_ErrorCode OTF2_MarkerReader_ReadMarkers (OTF2_MarkerReader ∗reader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  After callback registration, the markers could be read with the following function. The user of this function tells the system how many markers it is able to handle (recordsToRead) and the function returns how many markers where in the stream (recordsRead). It should usually be the case that both values are the same. If this is not the case, then there where less records than requested in the stream.
- OTF2_ErrorCode OTF2_MarkerReader_SetCallbacks (OTF2_MarkerReader ∗reader, const OTF2_MarkerReaderCallbacks ∗callbacks, void ∗userData)

  Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

E.26.1 Detailed Description

This file provides all routines that read marker records.

E.26.2 Function Documentation

E.26.2.1 OTF2_ErrorCode OTF2_MarkerReader_ReadMarkers (OTF2_MarkerReader ∗ *reader,* uint64_t *recordsToRead,* uint64_t ∗ *recordsRead*)

After callback registration, the markers could be read with the following function. The user of this function tells the system how many markers it is able to handle (recordsToRead) and the function returns how many markers where in the stream (recordsRead). It should usually be the case that both values are the same. If this is not the case, then there where less records than requested in the stream.

Parameters

reader	Reader Object.
record-sToRead	How many records have to be read next.
record-sRead	How many records where read?

Since

Version 1.2

Returns

OTF2_ErrorCode with !=OTF2_SUCCESS if there was an error.

E.26.2.2 OTF2_ErrorCode OTF2_MarkerReader_SetCallbacks (OTF2_MarkerReader ∗ *reader*, const OTF2_MarkerReaderCallbacks ∗ *callbacks*, void ∗ *userData*)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

reader	This given reader object will be setted up with new callback functions.
callbacks	Struct which holds a function pointer for each record type. *OTF2_MarkerReaderCallbacks_New*.
userData	Data passed as argument *userData* to the record callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.27 otf2/OTF2_MarkerReaderCallbacks.h File Reference

This defines the callbacks for the marker reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_IdMap.h>
#include <otf2/OTF2_Marker.h>
```

E.27 otf2/OTF2_MarkerReaderCallbacks.h File Reference

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_MarkerReaderCallback_DefMarker)(void ∗userData, OTF2_MarkerRef self, const char ∗markerGroup, const char ∗markerCategory, OTF2_MarkerSeverity severity)

  Function pointer definition for the callback which is triggered by a DefMarker definition record.
- typedef OTF2_CallbackCode(∗ OTF2_MarkerReaderCallback_Marker)(void ∗userData, OTF2_TimeStamp timestamp, OTF2_TimeStamp duration, OTF2_-MarkerRef marker, OTF2_MarkerScope scope, uint64_t scopeRef, const char ∗text)

  Function pointer definition for the callback which is triggered by a Marker record.
- typedef OTF2_CallbackCode(∗ OTF2_MarkerReaderCallback_Unknown)(void ∗userData)

  Function pointer definition for the callback which is triggered for an unknown marker.
- typedef struct OTF2_MarkerReaderCallbacks_struct OTF2_MarkerReaderCallbacks

  Opaque struct which holdes all definition record callbacks.

Functions

- void OTF2_MarkerReaderCallbacks_Clear (OTF2_MarkerReaderCallbacks ∗markerReaderCallbacks)

  Clears a struct for the marker callbacks.
- void OTF2_MarkerReaderCallbacks_Delete (OTF2_MarkerReaderCallbacks ∗markerReaderCallbacks)

  Deallocates a struct for the marker callbacks.
- OTF2_MarkerReaderCallbacks ∗ OTF2_MarkerReaderCallbacks_New (void)

  Allocates a new struct for the marker callbacks.
- OTF2_ErrorCode OTF2_MarkerReaderCallbacks_SetDefMarkerCallback (OTF2_-MarkerReaderCallbacks ∗markerReaderCallbacks, OTF2_MarkerReaderCallback_-DefMarker defMarkerCallback)

  Registers the callback for the DefMarker definition.
- OTF2_ErrorCode OTF2_MarkerReaderCallbacks_SetMarkerCallback (OTF2_-MarkerReaderCallbacks ∗markerReaderCallbacks, OTF2_MarkerReaderCallback_-Marker markerCallback)

  Registers the callback for the Marker record.

- OTF2_ErrorCode OTF2_MarkerReaderCallbacks_SetUnknownCallback (OTF2_-MarkerReaderCallbacks ∗markerReaderCallbacks, OTF2_MarkerReaderCallback_-Unknown unknownCallback)

  Registers the callback for an unknown marker.

E.27.1 Detailed Description

This defines the callbacks for the marker reader.

E.27.2 Typedef Documentation

E.27.2.1 typedef OTF2_CallbackCode(∗ OTF2_MarkerReaderCallback_-DefMarker)(void ∗userData, OTF2_MarkerRef self, const char ∗markerGroup, const char ∗markerCategory, OTF2_MarkerSeverity severity)

Function pointer definition for the callback which is triggered by a *DefMarker* definition record.

Parameters

userData	User data as set by *OTF2_Reader_RegisterMarkerCallbacks* or *OTF2_-MarkerReader_SetCallbacks*.
self	Reference to this marker defintion.
marker-Group	Group name, e.g., "MUST", ...
markerCat-egory	Marker category, e.g., "Argument type error", ... The tuple (marker-Group, markerCategory) must be unique over all marker definitions.
severity	The severity for this marker category.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.27.2.2 typedef OTF2_CallbackCode(∗ OTF2_MarkerReaderCallback_-Marker)(void ∗userData, OTF2_TimeStamp timestamp, OTF2_TimeStamp duration, OTF2_MarkerRef marker, OTF2_MarkerScope scope, uint64_t scopeRef, const char ∗text)

Function pointer definition for the callback which is triggered by a *Marker* record.

Parameters

userData	User data as set by *OTF2_Reader_RegisterMarkerCallbacks* or *OTF2_-MarkerReader_SetCallbacks*.
timestamp	Timestamp of the marker.
duration	Duration the marker applies.
marker	Reference to the marker defintion.
scope	The type of scope of this marker instance.
scopeRef	The reference to an element of the scope of this marker. Depends on `scope`.
text	A textual description for this marker.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.27.2.3 typedef OTF2_CallbackCode(∗ OTF2_MarkerReaderCallback_-Unknown)(void ∗userData)

Function pointer definition for the callback which is triggered for an unknown marker.

Parameters

userData	User data as set by *OTF2_Reader_RegisterMarkerCallbacks* or *OTF2_-MarkerReader_SetCallbacks*.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.27.3 Function Documentation

E.27.3.1 void OTF2_MarkerReaderCallbacks_Clear (OTF2_MarkerReaderCallbacks ∗ *markerReaderCallbacks*)

Clears a struct for the marker callbacks.

Since

Version 1.2

Parameters

markerReaderCallbacks	Handle to a struct previously allocated with *OTF2_MarkerReaderCallbacks_New*.

E.27.3.2 void OTF2_MarkerReaderCallbacks_Delete (OTF2_MarkerReaderCallbacks * *markerReaderCallbacks*)

Deallocates a struct for the marker callbacks.

Since

Version 1.2

Parameters

markerReaderCallbacks	Handle to a struct previously allocated with *OTF2_MarkerReaderCallbacks_New*.

E.27.3.3 OTF2_MarkerReaderCallbacks* OTF2_MarkerReaderCallbacks_New (void)

Allocates a new struct for the marker callbacks.

Since

Version 1.2

Returns

A newly allocated struct of type *OTF2_MarkerReaderCallbacks*.

E.27.3.4 OTF2_ErrorCode OTF2_MarkerReaderCallbacks_SetDefMarkerCallback (OTF2_MarkerReaderCallbacks * *markerReaderCallbacks*, OTF2_MarkerReaderCallback_DefMarker *defMarkerCallback*)

Registers the callback for the *DefMarker* definition.

Parameters

markerReaderCallbacks	Struct for all callbacks.
defMarkerCallback	Function which should be called for all *DefMarker* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.27.3.5 OTF2_ErrorCode OTF2_MarkerReaderCallbacks_SetMarkerCallback (OTF2_MarkerReaderCallbacks ∗ *markerReaderCallbacks*, OTF2_MarkerReaderCallback_Marker *markerCallback*)

Registers the callback for the *Marker* record.

Parameters

markerReaderCallbacks	Struct for all callbacks.
markerCallback	Function which should be called for all *Marker* records.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.27.3.6 OTF2_ErrorCode OTF2_MarkerReaderCallbacks_SetUnknownCallback (OTF2_MarkerReaderCallbacks ∗ *markerReaderCallbacks*, OTF2_MarkerReaderCallback_Unknown *unknownCallback*)

Registers the callback for an unknown marker.

Parameters

markerReaderCallbacks	Struct for all callbacks.
unknownCallback	Function which should be called for all unknown definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.28 otf2/OTF2_MarkerWriter.h File Reference

This file provides all routines that write marker records.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_Marker.h>
```

Typedefs

- typedef struct OTF2_MarkerWriter_struct OTF2_MarkerWriter

  Handle definition for the external marker writer.

Functions

- OTF2_ErrorCode OTF2_MarkerWriter_WriteDefMarker (OTF2_MarkerWriter ∗writerHandle, OTF2_MarkerRef self, const char ∗markerGroup, const char ∗markerCategory, OTF2_MarkerSeverity severity)

Write a marker definition.

- OTF2_ErrorCode OTF2_MarkerWriter_WriteMarker (OTF2_MarkerWriter ∗writerHandle, OTF2_TimeStamp timestamp, OTF2_TimeStamp duration, OTF2_MarkerRef marker, OTF2_MarkerScope scope, uint64_t scopeRef, const char ∗text)

  Write a marker record.

E.28.1 Detailed Description

This file provides all routines that write marker records.

E.28.2 Function Documentation

E.28.2.1 OTF2_ErrorCode OTF2_MarkerWriter_WriteDefMarker (OTF2_MarkerWriter ∗ *writerHandle*, OTF2_MarkerRef *self*, const char ∗ *markerGroup*, const char ∗ *markerCategory*, OTF2_MarkerSeverity *severity*)

Write a marker definition.

Parameters

writerHandle	Marker writer handle.
self	Reference to this marker definition.
markerGroup	Group name, e.g., "MUST", ...
markerCategory	Marker category, e.g., "Argument type error", ... The tuple (markerGroup, markerCategory) must be unique over all marker definitions.
severity	The severity for this marker category.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.28.2.2 OTF2_ErrorCode OTF2_MarkerWriter_WriteMarker (OTF2_MarkerWriter * *writerHandle*, OTF2_TimeStamp *timestamp*, OTF2_TimeStamp *duration*, OTF2_MarkerRef *marker*, OTF2_MarkerScope *scope*, uint64_t *scopeRef*, const char * *text*)

Write a marker record.

Parameters

writerHandle	Marker writer handle.
timestamp	Time of the marker.
duration	A possible duration of this marker. May be 0.
marker	Reference to a marker definition.
scope	The type of scope of this marker instance: *OTF2_MARKER_SCOPE_GLOBAL*, *OTF2_MARKER_SCOPE_LOCATION*, *OTF2_MARKER_SCOPE_LOCATION_GROUP*, *OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE*, *OTF2_MARKER_SCOPE_GROUP*, or *OTF2_MARKER_SCOPE_COMM*.
scopeRef	The reference to an element of the scope of this marker. Depends on `scope`.
text	A textual description for this marker.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.29 otf2/OTF2_MPI_Collectives.h File Reference

MPI collectives for OTF2.

```
#include <otf2/otf2.h>
#include <mpi.h>
```

Data Structures

- struct OTF2_CollectiveContext

  Collective context which wraps an MPI communicator.
- struct OTF2_MPI_UserData

  User data structure, which will be used by the MPI collectives.

E.29 otf2/OTF2_MPI_Collectives.h File Reference

Defines

- #define OTF2_MPI_DOUBLE MPI_DOUBLE

  Define this macro to an suitable MPI datatype to be used for `double` before including the header. `MPI_DOUBLE` is used as proper default value.
- #define OTF2_MPI_FLOAT MPI_FLOAT

  Define this macro to an suitable MPI datatype to be used for `float` before including the header. `MPI_FLOAT` is used as proper default value.
- #define OTF2_MPI_INT16_T MPI_SHORT

  Define this macro to an suitable MPI datatype to be used for `int32_t` before including the header. `MPI_SHORT` or `MPI_INT32_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.
- #define OTF2_MPI_INT32_T MPI_INT

  Define this macro to an suitable MPI datatype to be used for `int32_t` before including the header. `MPI_INT` or `MPI_INT32_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.
- #define OTF2_MPI_INT64_T MPI_INT64_T

  Define this macro to an suitable MPI datatype to be used for `int64_t` before including the header. `MPI_INT64_T` is used as proper default value in case of an MPI 3.0 conforming implementation.
- #define OTF2_MPI_INT8_T MPI_CHAR

  Define this macro to an suitable MPI datatype to be used for `int8_t` before including the header. `MPI_CHAR` or `MPI_INT8_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.
- #define OTF2_MPI_UINT16_T MPI_UNSIGNED_SHORT

  Define this macro to an suitable MPI datatype to be used for `uint16_t` before including the header. `MPI_UNSIGNED_SHORT` or `MPI_UINT16_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.
- #define OTF2_MPI_UINT32_T MPI_UNSIGNED

  Define this macro to an suitable MPI datatype to be used for `uint32_t` before including the header. `MPI_UNSIGNED` or `MPI_UINT32_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.
- #define OTF2_MPI_UINT64_T MPI_UINT64_T

  Define this macro to an suitable MPI datatype to be used for `uint64_t` before including the header. `MPI_UINT64_T` is used as proper default value in case of an MPI 3.0 conforming implementation.
- #define OTF2_MPI_UINT8_T MPI_UNSIGNED_CHAR

  Define this macro to an suitable MPI datatype to be used for `uint8_t` before including the header. `MPI_UNSIGNED_CHAR` or `MPI_UINT8_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.
- #define OTF2_MPI_USE_PMPI

If you want that the collectives call the PMPI interface, define this macro before including the header.

Functions

- static OTF2_ErrorCode OTF2_MPI_Archive_SetCollectiveCallbacks (OTF2_-Archive ∗archive, MPI_Comm globalComm, MPI_Comm localComm)

  Register an MPI collective context to an OTF2 archive.
- static OTF2_ErrorCode OTF2_MPI_Archive_SetCollectiveCallbacksSplit (OTF2_-Archive ∗archive, MPI_Comm globalComm, uint32_t numberOfFiles)

  Register an MPI collective context to an OTF2 archive.
- static OTF2_ErrorCode OTF2_MPI_Reader_SetCollectiveCallbacks (OTF2_-Reader ∗reader, MPI_Comm globalComm)

  Register an MPI collective context to an OTF2 reader.

E.29.1 Detailed Description

MPI collectives for OTF2. See Usage in reading mode - MPI example and Usage in writing mode - MPI example for instruction how to use.

E.29.2 Function Documentation

E.29.2.1 static OTF2_ErrorCode OTF2_MPI_Archive_SetCollectiveCallbacks (OTF2_Archive ∗ *archive,* MPI_Comm *globalComm,* MPI_Comm *localComm*) `[static]`

Register an MPI collective context to an OTF2 archive.

Parameters

archive	The archive handle.
global-Comm	The global communicator to use. Will be duplicated.
localComm	The local communicator to use. Maybe MPI_COMM_NULL, otherwise all `localComm` must be disjoint and join to `globalComm`. Will be duplicated.

Returns

Success or error code.

E.29.2.2 static OTF2_ErrorCode OTF2_MPI_Archive_SetCollectiveCallbacksSplit (OTF2_Archive * *archive*, MPI_Comm *globalComm*, uint32_t *numberOfFiles*) [static]

Register an MPI collective context to an OTF2 archive.

Parameters

archive	The archive handle.
global-Comm	The global communicator to use. Will be duplicated.
numberOf-Files	Splits the `globalComm` into `numberOfFiles` disjoint sub-communicators and evenly distribute the ranks among them.

Returns

Success or error code.

E.29.2.3 static OTF2_ErrorCode OTF2_MPI_Reader_SetCollectiveCallbacks (OTF2_Reader * *reader*, MPI_Comm *globalComm*) [static]

Register an MPI collective context to an OTF2 reader.

Parameters

reader	The reader handle.
global-Comm	The global communicator to use. Will be duplicated.

Returns

Success or error code.

E.30 otf2/OTF2_OpenMP_Locks.h File Reference

OpenMP locks for OTF2.

```
#include <otf2/otf2.h>
#include <omp.h>
```

Data Structures

- struct OTF2_Lock

The OpenMP locking object type.

Functions

- static OTF2_ErrorCode OTF2_OpenMP_Archive_SetLockingCallbacks (OTF2_-Archive ∗archive)

  Register callbacks to use OpenMP locks for a OTF2 archive.
- static OTF2_ErrorCode OTF2_OpenMP_Reader_SetLockingCallbacks (OTF2_-Reader ∗reader)

  Register callbacks to use OpenMP locks for a OTF2 reader.

E.30.1 Detailed Description

OpenMP locks for OTF2.

E.30.2 Function Documentation

E.30.2.1 static OTF2_ErrorCode OTF2_OpenMP_Archive_SetLockingCallbacks (OTF2_Archive ∗ *archive*) `[static]`

Register callbacks to use OpenMP locks for a OTF2 archive.

Parameters

archive	The archive handle.

Since

Version 1.5

Returns

Success or error code.

E.30.2.2 static OTF2_ErrorCode OTF2_OpenMP_Reader_SetLockingCallbacks (OTF2_Reader ∗ *reader*) `[static]`

Register callbacks to use OpenMP locks for a OTF2 reader.

Parameters

reader	The reader handle.

Since

Version 1.5

Returns

Success or error code.

E.31 otf2/OTF2_Pthread_Locks.h File Reference

Pthread locks for OTF2.

```
#include <otf2/otf2.h>
#include <pthread.h>
```

Data Structures

- struct OTF2_Lock

  The OpenMP locking object type.
- struct OTF2_Pthread_UserData

  User data structure, which will be used by the Pthread locks.

Functions

- static OTF2_ErrorCode OTF2_Pthread_Archive_SetLockingCallbacks (OTF2_Archive ∗archive, const pthread_mutexattr_t ∗mutexAttribute)

  Register callbacks to use Pthread mutexes for a OTF2 archive.
- static OTF2_ErrorCode OTF2_Pthread_Reader_SetLockingCallbacks (OTF2_Reader ∗reader, const pthread_mutexattr_t ∗mutexAttribute)

  Register callbacks to use Pthread mutexes for a OTF2 reader.

E.31.1 Detailed Description

Pthread locks for OTF2.

E.31.2 Function Documentation

E.31.2.1 static OTF2_ErrorCode OTF2_Pthread_Archive_SetLockingCallbacks (OTF2_Archive ∗ *archive*, const pthread_mutexattr_t ∗ *mutexAttribute*) `[static]`

Register callbacks to use Pthread mutexes for a OTF2 archive.

Parameters

archive	The archive handle.
mutexAttribute	A possible *pthread_mutexattr_t* which will e used in all *pthread_mutex_init* calls. A corresponding *pthread_mutexattr_destroy* call is done when the archive will be closed.

Since

Version 1.5

Returns

Success or error code.

E.31.2.2 static OTF2_ErrorCode OTF2_Pthread_Reader_SetLockingCallbacks (OTF2_Reader ∗ *reader,* const pthread_mutexattr_t ∗ *mutexAttribute*) `[static]`

Register callbacks to use Pthread mutexes for a OTF2 reader.

Parameters

reader	The reader handle.
mutexAttribute	A possible *pthread_mutexattr_t* which will e used in all *pthread_mutex_init* calls. A corresponding *pthread_mutexattr_destroy* call is done when the reader will be closed.

Since

Version 1.5

Returns

Success or error code.

E.32 otf2/OTF2_Reader.h File Reference

Reading interface for OTF2 archives.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Archive.h>
```

E.32 otf2/OTF2_Reader.h File Reference

Typedefs

- typedef struct OTF2_Reader_struct OTF2_Reader

  Keeps all necessary information for the reader.

Functions

- OTF2_ErrorCode OTF2_Reader_Close (OTF2_Reader ∗reader)

  Close a reader handle.
- OTF2_ErrorCode OTF2_Reader_CloseDefFiles (OTF2_Reader ∗reader)

  Closes the local definitions file container.
- OTF2_ErrorCode OTF2_Reader_CloseDefReader (OTF2_Reader ∗reader, OTF2_DefReader ∗defReader)

  Close a local definition reader.
- OTF2_ErrorCode OTF2_Reader_CloseEvtFiles (OTF2_Reader ∗reader)

  Closes the events file container.
- OTF2_ErrorCode OTF2_Reader_CloseEvtReader (OTF2_Reader ∗reader, OTF2_-EvtReader ∗evtReader)

  Close a local event reader.
- OTF2_ErrorCode OTF2_Reader_CloseGlobalDefReader (OTF2_Reader ∗reader, OTF2_GlobalDefReader ∗globalDefReader)

  Closes the global definition reader.
- OTF2_ErrorCode OTF2_Reader_CloseGlobalEvtReader (OTF2_Reader ∗reader, OTF2_GlobalEvtReader ∗globalEvtReader)

  Closes the global event reader.
- OTF2_ErrorCode OTF2_Reader_CloseGlobalSnapReader (OTF2_Reader ∗reader, OTF2_GlobalSnapReader ∗globalSnapReader)

  Closes the global snapshot reader.
- OTF2_ErrorCode OTF2_Reader_CloseMarkerReader (OTF2_Reader ∗reader, OTF2_MarkerReader ∗markerReader)

  Closes the marker reader.
- OTF2_ErrorCode OTF2_Reader_CloseMarkerWriter (OTF2_Reader ∗reader, OTF2_MarkerWriter ∗markerWriter)

  Closes the marker writer.
- OTF2_ErrorCode OTF2_Reader_CloseSnapFiles (OTF2_Reader ∗reader)

  Closes the snapshots file container.
- OTF2_ErrorCode OTF2_Reader_CloseSnapReader (OTF2_Reader ∗reader, OTF2_SnapReader ∗snapReader)

  Close a local snapshot reader.

- OTF2_ErrorCode OTF2_Reader_CloseThumbReader (OTF2_Reader ∗reader, OTF2_ThumbReader ∗thumbReader)

  Close an opened thumbnail reader.
- OTF2_ErrorCode OTF2_Reader_GetBoolProperty (OTF2_Reader ∗reader, const char ∗name, bool ∗value)

  Get the value of the named trace file property as boolean.
- OTF2_ErrorCode OTF2_Reader_GetChunkSize (OTF2_Reader ∗reader, uint64_-t ∗chunkSizeEvents, uint64_t ∗chunkSizeDefinitions)

  Get event and definition chunk sizes.
- OTF2_ErrorCode OTF2_Reader_GetCompression (OTF2_Reader ∗reader, OTF2_Compression ∗compression)

  Get copression mode.
- OTF2_ErrorCode OTF2_Reader_GetCreator (OTF2_Reader ∗reader, char ∗∗creator)

  Get creator name.
- OTF2_DefReader ∗ OTF2_Reader_GetDefReader (OTF2_Reader ∗reader, OTF2_LocationRef location)

  Get a local definition reader.
- OTF2_ErrorCode OTF2_Reader_GetDescription (OTF2_Reader ∗reader, char ∗∗description)

  Get description.
- OTF2_EvtReader ∗ OTF2_Reader_GetEvtReader (OTF2_Reader ∗reader, OTF2_LocationRef location)

  Get a local event reader.
- OTF2_ErrorCode OTF2_Reader_GetFileSubstrate (OTF2_Reader ∗reader, OTF2_FileSubstrate ∗substrate)

  Get file substrate information.
- OTF2_GlobalDefReader ∗ OTF2_Reader_GetGlobalDefReader (OTF2_Reader ∗reader)

  Get a global definition reader.
- OTF2_GlobalEvtReader ∗ OTF2_Reader_GetGlobalEvtReader (OTF2_Reader ∗reader)

  Get a global event reader.
- OTF2_GlobalSnapReader ∗ OTF2_Reader_GetGlobalSnapReader (OTF2_-Reader ∗reader)

  Get a global snap reader.
- OTF2_ErrorCode OTF2_Reader_GetMachineName (OTF2_Reader ∗reader, char ∗∗machineName)

  Get machine name.

- OTF2_MarkerReader ∗ OTF2_Reader_GetMarkerReader (OTF2_Reader ∗reader)

    Get a marker reader.
- OTF2_MarkerWriter ∗ OTF2_Reader_GetMarkerWriter (OTF2_Reader ∗reader)

    Get a marker writer.
- OTF2_ErrorCode OTF2_Reader_GetNumberOfGlobalDefinitions (OTF2_-Reader ∗reader, uint64_t ∗numberOfDefinitions)

    Get number of global definitions.
- OTF2_ErrorCode OTF2_Reader_GetNumberOfLocations (OTF2_Reader ∗reader, uint64_t ∗numberOfLocations)

    Get number of locations.
- OTF2_ErrorCode OTF2_Reader_GetNumberOfSnapshots (OTF2_Reader ∗reader, uint32_t ∗number)

    Get number of snapshots.
- OTF2_ErrorCode OTF2_Reader_GetNumberOfThumbnails (OTF2_Reader ∗reader, uint32_t ∗number)

    Get number of thumbs.
- OTF2_ErrorCode OTF2_Reader_GetProperty (OTF2_Reader ∗reader, const char ∗name, char ∗∗value)

    Get the value of the named trace file property.
- OTF2_ErrorCode OTF2_Reader_GetPropertyNames (OTF2_Reader ∗reader, uint32_t ∗numberOfProperties, char ∗∗∗names)

    Get the names of all trace file properties.
- OTF2_SnapReader ∗ OTF2_Reader_GetSnapReader (OTF2_Reader ∗reader, OTF2_LocationRef location)

    Get a local snapshot reader.
- OTF2_ThumbReader ∗ OTF2_Reader_GetThumbReader (OTF2_Reader ∗reader, uint32_t number)

    Get a thumb reader.
- OTF2_ErrorCode OTF2_Reader_GetTraceId (OTF2_Reader ∗reader, uint64_-t ∗id)

    Get the identifier of the trace file.
- OTF2_ErrorCode OTF2_Reader_GetVersion (OTF2_Reader ∗reader, uint8_-t ∗major, uint8_t ∗minor, uint8_t ∗bugfix)

    Get OTF2 version.
- OTF2_ErrorCode OTF2_Reader_HasGlobalEvent (OTF2_Reader ∗reader, OTF2_GlobalEvtReader ∗evtReader, int ∗flag)

    Has the global event reader at least one more event to deliver.
- OTF2_Reader ∗ OTF2_Reader_Open (const char ∗anchorFilePath)

Create a new reader handle.

- OTF2_ErrorCode OTF2_Reader_OpenDefFiles (OTF2_Reader ∗reader)

  Open the local definitions file container.
- OTF2_ErrorCode OTF2_Reader_OpenEvtFiles (OTF2_Reader ∗reader)

  Open the events file container.
- OTF2_ErrorCode OTF2_Reader_OpenSnapFiles (OTF2_Reader ∗reader)

  Open the snapshots file container.
- OTF2_ErrorCode OTF2_Reader_ReadAllGlobalDefinitions (OTF2_Reader ∗reader, OTF2_GlobalDefReader ∗defReader, uint64_t ∗definitionsRead)

  Read all definitions via a global definition reader.
- OTF2_ErrorCode OTF2_Reader_ReadAllGlobalEvents (OTF2_Reader ∗reader, OTF2_GlobalEvtReader ∗evtReader, uint64_t ∗eventsRead)

  Read all events via a global event reader.
- OTF2_ErrorCode OTF2_Reader_ReadAllGlobalSnapshots (OTF2_Reader ∗reader, OTF2_GlobalSnapReader ∗snapReader, uint64_t ∗recordsRead)

  Read all records via a global snapshot reader.
- OTF2_ErrorCode OTF2_Reader_ReadAllLocalDefinitions (OTF2_Reader ∗reader, OTF2_DefReader ∗defReader, uint64_t ∗definitionsRead)

  Read all definitions via a local definition reader.
- OTF2_ErrorCode OTF2_Reader_ReadAllLocalEvents (OTF2_Reader ∗reader, OTF2_EvtReader ∗evtReader, uint64_t ∗eventsRead)

  Read all events via a local event reader.
- OTF2_ErrorCode OTF2_Reader_ReadAllLocalSnapshots (OTF2_Reader ∗reader, OTF2_SnapReader ∗snapReader, uint64_t ∗recordsRead)

  Read all records via a local snapshot reader.
- OTF2_ErrorCode OTF2_Reader_ReadAllMarkers (OTF2_Reader ∗reader, OTF2_MarkerReader ∗markerReader, uint64_t ∗markersRead)

  Read all markers via a marker reader.
- OTF2_ErrorCode OTF2_Reader_ReadGlobalDefinitions (OTF2_Reader ∗reader, OTF2_GlobalDefReader ∗defReader, uint64_t definitionsToRead, uint64_t ∗definitionsRead)

  Read a given number of definitions via a global definition reader.
- OTF2_ErrorCode OTF2_Reader_ReadGlobalEvent (OTF2_Reader ∗reader, OTF2_GlobalEvtReader ∗evtReader)

  Read an event via a global event reader.
- OTF2_ErrorCode OTF2_Reader_ReadGlobalEvents (OTF2_Reader ∗reader, OTF2_GlobalEvtReader ∗evtReader, uint64_t eventsToRead, uint64_t ∗eventsRead)

  Read a given number of events via a global event reader.

- OTF2_ErrorCode OTF2_Reader_ReadGlobalSnapshots (OTF2_Reader ∗reader, OTF2_GlobalSnapReader ∗snapReader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  Read a given number of records via a global snapshot reader.
- OTF2_ErrorCode OTF2_Reader_ReadLocalDefinitions (OTF2_Reader ∗reader, OTF2_DefReader ∗defReader, uint64_t definitionsToRead, uint64_t ∗definitionsRead)

  Read a given number of definitions via a local definition reader.
- OTF2_ErrorCode OTF2_Reader_ReadLocalEvents (OTF2_Reader ∗reader, OTF2_EvtReader ∗evtReader, uint64_t eventsToRead, uint64_t ∗eventsRead)

  Read a given number of events via a local event reader.
- OTF2_ErrorCode OTF2_Reader_ReadLocalEventsBackward (OTF2_Reader ∗reader, OTF2_EvtReader ∗evtReader, uint64_t eventsToRead, uint64_t ∗eventsRead)

  Read a given number of events via a local event reader backwards.
- OTF2_ErrorCode OTF2_Reader_ReadLocalSnapshots (OTF2_Reader ∗reader, OTF2_SnapReader ∗snapReader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  Read a given number of records via a local snapshot reader.
- OTF2_ErrorCode OTF2_Reader_ReadMarkers (OTF2_Reader ∗reader, OTF2_-MarkerReader ∗markerReader, uint64_t markersToRead, uint64_t ∗markersRead)

  Read a given number of markers via a marker reader.
- OTF2_ErrorCode OTF2_Reader_RegisterDefCallbacks (OTF2_Reader ∗reader, OTF2_DefReader ∗defReader, const OTF2_DefReaderCallbacks ∗callbacks, void ∗userData)

  Register local definition reader callbacks.
- OTF2_ErrorCode OTF2_Reader_RegisterEvtCallbacks (OTF2_Reader ∗reader, OTF2_EvtReader ∗evtReader, const OTF2_EvtReaderCallbacks ∗callbacks, void ∗userData)

  Register event reader callbacks.
- OTF2_ErrorCode OTF2_Reader_RegisterGlobalDefCallbacks (OTF2_Reader ∗reader, OTF2_GlobalDefReader ∗defReader, const OTF2_GlobalDefReaderCallbacks ∗callbacks, void ∗userData)

  Register global definition reader callbacks.
- OTF2_ErrorCode OTF2_Reader_RegisterGlobalEvtCallbacks (OTF2_Reader ∗reader, OTF2_GlobalEvtReader ∗evtReader, const OTF2_GlobalEvtReaderCallbacks ∗callbacks, void ∗userData)

  Register global event reader callbacks.

- OTF2_ErrorCode OTF2_Reader_RegisterGlobalSnapCallbacks (OTF2_Reader ∗reader, OTF2_GlobalSnapReader ∗evtReader, const OTF2_GlobalSnapReaderCallbacks ∗callbacks, void ∗userData)

  Register global event reader callbacks.
- OTF2_ErrorCode OTF2_Reader_RegisterMarkerCallbacks (OTF2_Reader ∗reader, OTF2_MarkerReader ∗markerReader, const OTF2_MarkerReaderCallbacks ∗callbacks, void ∗userData)

  Register marker reader callbacks.
- OTF2_ErrorCode OTF2_Reader_RegisterSnapCallbacks (OTF2_Reader ∗reader, OTF2_SnapReader ∗snapReader, const OTF2_SnapReaderCallbacks ∗callbacks, void ∗userData)

  Register snapshot event reader callbacks.
- OTF2_ErrorCode OTF2_Reader_SelectLocation (OTF2_Reader ∗reader, OTF2_-LocationRef location)

  Select a location to be read.
- OTF2_ErrorCode OTF2_Reader_SetCollectiveCallbacks (OTF2_Reader ∗reader, const OTF2_CollectiveCallbacks ∗collectiveCallbacks, void ∗collectiveData, OTF2_CollectiveContext ∗globalCommContext, OTF2_CollectiveContext ∗localCommContext)

  Set the collective callbacks for the reader.
- OTF2_ErrorCode OTF2_Reader_SetHint (OTF2_Reader ∗reader, OTF2_-Hint hint, void ∗value)

  Set the hint in the reader to the given value.
- OTF2_ErrorCode OTF2_Reader_SetLockingCallbacks (OTF2_Reader ∗reader, const OTF2_LockingCallbacks ∗lockingCallbacks, void ∗lockingData)

  Set the locking callbacks for the reader.
- OTF2_ErrorCode OTF2_Reader_SetSerialCollectiveCallbacks (OTF2_Reader ∗reader)

  Convenient function to set the collective callbacks to an serial implementation.

E.32.1 Detailed Description

Reading interface for OTF2 archives.

E.32.2 Function Documentation

E.32.2.1 OTF2_ErrorCode OTF2_Reader_Close (OTF2_Reader ∗ *reader*)

Close a reader handle.

Closes a reader handle and releases all associated handles. Does nothing if NULL is provided.

Parameters

reader	Reader handle.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.2 OTF2_ErrorCode OTF2_Reader_CloseDefFiles (OTF2_Reader ∗ *reader*)

Closes the local definitions file container.

This function is an collective operation.

All previously used local definition readers must be closed before this call.

Parameters

reader	Reader handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.3 OTF2_ErrorCode OTF2_Reader_CloseDefReader (OTF2_Reader ∗ *reader*, OTF2_DefReader ∗ *defReader*)

Close a local definition reader.

Parameters

reader	Valid reader handle.
defReader	Definition reader to be closed.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.4 OTF2_ErrorCode OTF2_Reader_CloseEvtFiles (OTF2_Reader ∗ *reader*)

Closes the events file container.

All previously used event readers must be closed before this call.

This function is an collective operation.

Parameters

reader	Reader handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.5 OTF2_ErrorCode OTF2_Reader_CloseEvtReader (OTF2_Reader ∗ *reader*, OTF2_EvtReader ∗ *evtReader*)

Close a local event reader.

Parameters

reader	Valid reader handle.
evtReader	Event reader to be closed.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.6 OTF2_ErrorCode OTF2_Reader_CloseGlobalDefReader (OTF2_Reader ∗ *reader*, OTF2_GlobalDefReader ∗ *globalDefReader*)

Closes the global definition reader.

Parameters

reader	Valid reader handle.
globalDef-Reader	The global definition reader.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.7 OTF2_ErrorCode OTF2_Reader_CloseGlobalEvtReader (OTF2_Reader ∗ *reader*, OTF2_GlobalEvtReader ∗ *globalEvtReader*)

Closes the global event reader.

This closes also all local event readers.

Parameters

reader	Valid reader handle.
glob-alEvtReader	The global event reader.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.8 OTF2_ErrorCode OTF2_Reader_CloseGlobalSnapReader (OTF2_Reader ∗ *reader*, OTF2_GlobalSnapReader ∗ *globalSnapReader*)

Closes the global snapshot reader.

Parameters

reader	Valid reader handle.
global-SnapReader	The global snapshot reader.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.32.2.9 OTF2_ErrorCode OTF2_Reader_CloseMarkerReader (OTF2_Reader ∗ *reader*, OTF2_MarkerReader ∗ *markerReader*)

Closes the marker reader.

Parameters

reader	Valid reader handle.
marker-Reader	The marker reader.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.10 OTF2_ErrorCode OTF2_Reader_CloseMarkerWriter (OTF2_Reader * *reader,* OTF2_MarkerWriter * *markerWriter*)

Closes the marker writer.

Parameters

reader	Valid reader handle.
marker-Writer	The marker writer.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.11 OTF2_ErrorCode OTF2_Reader_CloseSnapFiles (OTF2_Reader * *reader*)

Closes the snapshots file container.

This function is an collective operation.

All previously used snapshot readers must be closed before this call.

Parameters

reader	Reader handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.12 OTF2_ErrorCode OTF2_Reader_CloseSnapReader (OTF2_Reader ∗ *reader,* OTF2_SnapReader ∗ *snapReader*)

Close a local snapshot reader.

Parameters

reader	Valid reader handle.
snapReader	snapshot reader to be closed.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.32.2.13 OTF2_ErrorCode OTF2_Reader_CloseThumbReader (OTF2_Reader ∗ *reader,* OTF2_ThumbReader ∗ *thumbReader*)

Close an opened thumbnail reader.

Parameters

reader	Reader handle.
thumbReader	Thumbn reader handle to be closed.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.14 OTF2_ErrorCode OTF2_Reader_GetBoolProperty (OTF2_Reader ∗ *reader,* const char ∗ *name,* bool ∗ *value*)

Get the value of the named trace file property as boolean.

Parameters

	reader	Reader handle.
	name	Name of the property.
out	*value*	Returned boolean value of the property.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_PROPERTY_NOT_FOUND if the named property was not found

OTF2_ERROR_PROPERTY_VALUE_INVALID if the value could not be interpreted as an boolean value

E.32.2.15 OTF2_ErrorCode OTF2_Reader_GetChunkSize (OTF2_Reader * *reader*, uint64_t * *chunkSizeEvents*, uint64_t * *chunkSizeDefinitions*)

Get event and definition chunk sizes.

Parameters

	reader	Reader handle.
out	*chunk-SizeEvents*	Returned size of event chunks
out	*chunk-SizeDefini-tions*	Returned size of definition chunks.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.16 OTF2_ErrorCode OTF2_Reader_GetCompression (OTF2_Reader * *reader*, OTF2_Compression * *compression*)

Get copression mode.

Parameters

	reader	Reader handle.
out	*compres-sion*	Returned compression mode.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.17 OTF2_ErrorCode OTF2_Reader_GetCreator (OTF2_Reader ∗ *reader*, char ∗∗ *creator*)

Get creator name.

Parameters

	reader	Reader handle.
`out`	*creator*	Returned creator. Allocated with *malloc*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.18 OTF2_DefReader∗ OTF2_Reader_GetDefReader (OTF2_Reader ∗ *reader*, OTF2_LocationRef *location*)

Get a local definition reader.

Parameters

reader	Valid reader handle.
location	Location ID for the requested local reader.

Returns

Returns a handle to the local definition reader if successful, NULL otherwise.

E.32.2.19 OTF2_ErrorCode OTF2_Reader_GetDescription (OTF2_Reader ∗ *reader*, char ∗∗ *description*)

Get description.

Parameters

	reader	Reader handle.
`out`	*description*	Returned description. Allocated with *malloc*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.20 OTF2_EvtReader∗ OTF2_Reader_GetEvtReader (OTF2_Reader ∗ *reader*, OTF2_LocationRef *location*)

Get a local event reader.

Parameters

reader	Valid reader handle.
location	Location ID for the requested local reader.

Returns

Returns a handle to the local event reader if successful, NULL otherwise.

E.32.2.21 OTF2_ErrorCode OTF2_Reader_GetFileSubstrate (OTF2_Reader ∗ *reader*, OTF2_FileSubstrate ∗ *substrate*)

Get file substrate information.

Parameters

	reader	Reader handle.
`out`	*substrate*	Returned file substrate.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.22 OTF2_GlobalDefReader∗ OTF2_Reader_GetGlobalDefReader (OTF2_Reader ∗ *reader*)

Get a global definition reader.

Parameters

reader	Valid reader handle.

Returns

Returns a handle to the global definition reader if successful, NULL otherwise.

E.32.2.23 OTF2_GlobalEvtReader∗ OTF2_Reader_GetGlobalEvtReader (OTF2_Reader ∗ *reader*)

Get a global event reader.

Parameters

reader	Valid reader handle.

Returns

Returns a handle to the global event reader if successful, NULL otherwise.

E.32.2.24 OTF2_GlobalSnapReader∗ OTF2_Reader_GetGlobalSnapReader (OTF2_Reader ∗ *reader*)

Get a global snap reader.

Parameters

reader	Valid reader handle.

Returns

Returns a handle to the global snap reader if successful, NULL otherwise.

Since

Version 1.2

E.32.2.25 OTF2_ErrorCode OTF2_Reader_GetMachineName (OTF2_Reader ∗ *reader*, char ∗∗ *machineName*)

Get machine name.

Parameters

	reader	Reader handle.
out	*machine-Name*	Returned machine name. Allocated with *malloc*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.26 OTF2_MarkerReader∗ OTF2_Reader_GetMarkerReader (OTF2_Reader ∗ *reader*)

Get a marker reader.

Parameters

reader	Valid reader handle.

Since

Version 1.2

Returns

Returns a handle to the marker reader if successful, NULL otherwise.

E.32.2.27 OTF2_MarkerWriter∗ OTF2_Reader_GetMarkerWriter (OTF2_Reader ∗ *reader*)

Get a marker writer.

Parameters

reader	Valid reader handle.

Since

Version 1.2

Returns

Returns a handle to the marker writer if successful, NULL otherwise.

E.32.2.28 OTF2_ErrorCode OTF2_Reader_GetNumberOfGlobalDefinitions (OTF2_Reader ∗ *reader*, uint64_t ∗ *numberOfDefinitions*)

Get number of global definitions.

Parameters

	reader	Reader handle.
out	*num-berOfDefi-nitions*	Returned number of global definitions.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.29 OTF2_ErrorCode OTF2_Reader_GetNumberOfLocations (OTF2_Reader ∗ *reader*, uint64_t ∗ *numberOfLocations*)

Get number of locations.

Parameters

	reader	Reader handle.
out	*num-berOfLoca-tions*	Returned number of locations.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.30 OTF2_ErrorCode OTF2_Reader_GetNumberOfSnapshots (OTF2_Reader ∗ *reader*, uint32_t ∗ *number*)

Get number of snapshots.

Parameters

	reader	Reader handle.
out	*number*	Returned number of snapshots.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.32.2.31 OTF2_ErrorCode OTF2_Reader_GetNumberOfThumbnails (OTF2_Reader ∗ *reader*, uint32_t ∗ *number*)

Get number of thumbs.

Parameters

	reader	Reader handle.
out	*number*	Returned number of thumbs.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.32.2.32 OTF2_ErrorCode OTF2_Reader_GetProperty (OTF2_Reader ∗ *reader*, const char ∗ *name*, char ∗∗ *value*)

Get the value of the named trace file property.

Parameters

	reader	Reader handle.
	name	Name of the property.
out	*value*	Returned value of the property. Allocated with *malloc*.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_PROPERTY_NOT_FOUND if the named property was not found

E.32.2.33 OTF2_ErrorCode OTF2_Reader_GetPropertyNames (OTF2_Reader ∗ *reader*, uint32_t ∗ *numberOfProperties*, char ∗∗∗ *names*)

Get the names of all trace file properties.

Parameters

	reader	Reader handle.

out	*numberOf-Properties*	Returned number of trace file properties.
out	*names*	Returned list of property names. Allocated with *malloc*. To release memory, just pass `*names` to *free*.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.34 OTF2_SnapReader* OTF2_Reader_GetSnapReader (OTF2_Reader * *reader*, OTF2_LocationRef *location*)

Get a local snapshot reader.

Parameters

reader	Valid reader handle.
location	Location ID for the requested local reader.

Returns

Returns a handle to the local event reader if successful, NULL otherwise.

Since

Version 1.2

E.32.2.35 OTF2_ThumbReader* OTF2_Reader_GetThumbReader (OTF2_Reader * *reader*, uint32_t *number*)

Get a thumb reader.

Parameters

reader	Reader handle.
number	Thumbnail number.

Since

Version 1.2

Returns

Returns a global definition writer handle if successful, NULL if an error occurs.

E.32.2.36 OTF2_ErrorCode OTF2_Reader_GetTraceId (OTF2_Reader ∗ *reader*, uint64_t ∗ *id*)

Get the identifier of the trace file.

Parameters

	reader	Reader handle.
out	*id*	Trace identifier.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.37 OTF2_ErrorCode OTF2_Reader_GetVersion (OTF2_Reader ∗ *reader*, uint8_t ∗ *major*, uint8_t ∗ *minor*, uint8_t ∗ *bugfix*)

Get OTF2 version.

Parameters

	reader	Valid reader handle.
out	*major*	Major version.
out	*minor*	Minor version.
out	*bugfix*	Bugfix revision.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.38 OTF2_ErrorCode OTF2_Reader_HasGlobalEvent (OTF2_Reader ∗ *reader*, OTF2_GlobalEvtReader ∗ *evtReader*, int ∗ *flag*)

Has the global event reader at least one more event to deliver.

Parameters

	reader	Global event reader handle.
	evtReader	Global event reader handle.

out	*flag*	In case of success, the flag will be set to 1 when there is at least more more event to read. To 0 if not. Otherwise the value is undefined.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.39 OTF2_Reader∗ OTF2_Reader_Open (const char ∗ *anchorFilePath*)

Create a new reader handle.

Creates a new reader handle, opens an according archive handle, and calls a routine to register all necessary function pointers.

Parameters

anchor-FilePath	Path to the anchor file e.g. 'trace.otf2'. This can be a relative as well as an absolute path.

Returns

Returns a handle to the reader if successful, NULL otherwise.

E.32.2.40 OTF2_ErrorCode OTF2_Reader_OpenDefFiles (OTF2_Reader ∗ *reader*)

Open the local definitions file container.

This function is an collective operation.

Parameters

reader	Reader handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.41 OTF2_ErrorCode OTF2_Reader_OpenEvtFiles (OTF2_Reader ∗ *reader*)

Open the events file container.

This function is an collective operation.

Parameters

reader	Reader handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.42 OTF2_ErrorCode OTF2_Reader_OpenSnapFiles (OTF2_Reader ∗ *reader*)

Open the snapshots file container.

This function is an collective operation.

Parameters

reader	Reader handle.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.43 OTF2_ErrorCode OTF2_Reader_ReadAllGlobalDefinitions (OTF2_Reader ∗ *reader,* OTF2_GlobalDefReader ∗ *defReader,* uint64_t ∗ *definitionsRead*)

Read all definitions via a global definition reader.

Parameters

	reader	Reader handle.

	defReader	Global definition reader handle.
out	*definition-sRead*	Return pointer to the number of definitions actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.44 OTF2_ErrorCode OTF2_Reader_ReadAllGlobalEvents (OTF2_Reader ∗ *reader*, OTF2_GlobalEvtReader ∗ *evtReader*, uint64_t ∗ *eventsRead*)

Read all events via a global event reader.

Parameters

	reader	Reader handle.
	evtReader	Global event reader handle.
out	*eventsRead*	Return pointer to the number of events actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.45 OTF2_ErrorCode OTF2_Reader_ReadAllGlobalSnapshots (OTF2_Reader ∗ *reader*, OTF2_GlobalSnapReader ∗ *snapReader*, uint64_t ∗ *recordsRead*)

Read all records via a global snapshot reader.

Parameters

	reader	Reader handle.
	snapReader	Global snapshot reader handle.
out	*record-sRead*	Return pointer to the number of records

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.32.2.46 OTF2_ErrorCode OTF2_Reader_ReadAllLocalDefinitions (OTF2_Reader ∗ *reader*, OTF2_DefReader ∗ *defReader*, uint64_t ∗ *definitionsRead*)

Read all definitions via a local definition reader.

Parameters

	reader	Reader handle.
	defReader	Local definition reader handle.
`out`	*definitionsRead*	Return pointer to the number of definitions actually read.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INTERRUPTED_BY_CALLBACK if an user supplied callback returned OTF2_CALLBACK_INTERRUPT

OTF2_ERROR_DUPLICATE_MAPPING_TABLE if an duplicate mapping table definition was read

otherwise the error code

E.32.2.47 OTF2_ErrorCode OTF2_Reader_ReadAllLocalEvents (OTF2_Reader ∗ *reader*, OTF2_EvtReader ∗ *evtReader*, uint64_t ∗ *eventsRead*)

Read all events via a local event reader.

Parameters

	reader	Reader handle.
	evtReader	Local event reader handle.
`out`	*eventsRead*	Return pointer to the number of events actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.48 OTF2_ErrorCode OTF2_Reader_ReadAllLocalSnapshots (OTF2_Reader ∗ *reader*, OTF2_SnapReader ∗ *snapReader*, uint64_t ∗ *recordsRead*)

Read all records via a local snapshot reader.

Parameters

	reader	Reader handle.
	snapReader	Local snapshot reader handle.
out	*record-sRead*	Return pointer to the number of records

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.32.2.49 OTF2_ErrorCode OTF2_Reader_ReadAllMarkers (OTF2_Reader ∗ *reader,* OTF2_MarkerReader ∗ *markerReader,* uint64_t ∗ *markersRead*)

Read all markers via a marker reader.

Parameters

	reader	Reader handle.
	marker-Reader	Marker reader handle.
out	*marker-sRead*	Return pointer to the number of markers actually read.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.50 OTF2_ErrorCode OTF2_Reader_ReadGlobalDefinitions (OTF2_Reader ∗ *reader,* OTF2_GlobalDefReader ∗ *defReader,* uint64_t *definitionsToRead,* uint64_t ∗ *definitionsRead*)

Read a given number of definitions via a global definition reader.

Parameters

	reader	Reader handle.

	defReader	Global definition reader handle.
	definition-sToRead	Number definitions to be read.
out	*definition-sRead*	Return pointer to the number of definitions actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.51 OTF2_ErrorCode OTF2_Reader_ReadGlobalEvent (OTF2_Reader ∗ *reader*, OTF2_GlobalEvtReader ∗ *evtReader*)

Read an event via a global event reader.

Parameters

reader	Reader handle.
evtReader	Global event reader handle.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.52 OTF2_ErrorCode OTF2_Reader_ReadGlobalEvents (OTF2_Reader ∗ *reader*, OTF2_GlobalEvtReader ∗ *evtReader*, uint64_t *eventsToRead*, uint64_t ∗ *eventsRead*)

Read a given number of events via a global event reader.

Parameters

	reader	Reader handle.
	evtReader	Global event reader handle.
	eventsToReac	Number events to be read.
out	*eventsRead*	Return pointer to the number of events actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.53 OTF2_ErrorCode OTF2_Reader_ReadGlobalSnapshots (OTF2_Reader ∗ *reader,* OTF2_GlobalSnapReader ∗ *snapReader,* uint64_t *recordsToRead,* uint64_t ∗ *recordsRead*)

Read a given number of records via a global snapshot reader.

Parameters

	reader	Reader handle.
	snapReader	Global snapshot reader handle.
	record-sToRead	Number records to be read.
out	*record-sRead*	Return pointer to the number of records actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.32.2.54 OTF2_ErrorCode OTF2_Reader_ReadLocalDefinitions (OTF2_Reader ∗ *reader,* OTF2_DefReader ∗ *defReader,* uint64_t *definitionsToRead,* uint64_t ∗ *definitionsRead*)

Read a given number of definitions via a local definition reader.

Parameters

	reader	Reader handle.
	defReader	Local definition reader handle.
	definition-sToRead	Number definitions to be read.
out	*definition-sRead*	Return pointer to the number of definitions actually read.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INTERRUPTED_BY_CALLBACK if an user supplied call-back returned OTF2_CALLBACK_INTERRUPT

OTF2_ERROR_DUPLICATE_MAPPING_TABLE if an duplicate mapping table definition was read

otherwise the error code

E.32.2.55 OTF2_ErrorCode OTF2_Reader_ReadLocalEvents (OTF2_Reader ∗ *reader,* OTF2_EvtReader ∗ *evtReader,* uint64_t *eventsToRead,* uint64_t ∗ *eventsRead*)

Read a given number of events via a local event reader.

Parameters

reader	Reader handle.
evtReader	Local event reader handle.
eventsToReac	Number events to be read.
eventsRead	Return pointer to the number of events actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.56 OTF2_ErrorCode OTF2_Reader_ReadLocalEventsBackward (OTF2_Reader ∗ *reader,* OTF2_EvtReader ∗ *evtReader,* uint64_t *eventsToRead,* uint64_t ∗ *eventsRead*)

Read a given number of events via a local event reader backwards.

Parameters

	reader	Reader handle.
	evtReader	Local event reader handle.
	eventsToReac	Number events to be read.
out	*eventsRead*	Return pointer to the number of events actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.57 OTF2_ErrorCode OTF2_Reader_ReadLocalSnapshots (OTF2_Reader ∗ *reader,* OTF2_SnapReader ∗ *snapReader,* uint64_t *recordsToRead,* uint64_t ∗ *recordsRead*)

Read a given number of records via a local snapshot reader.

Parameters

reader	Reader handle.
snapReader	Local snapshot reader handle.
record-sToRead	Number records to be read.
record-sRead	Return pointer to the number of records actually read.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Since

Version 1.2

E.32.2.58 OTF2_ErrorCode OTF2_Reader_ReadMarkers (OTF2_Reader ∗ *reader,* OTF2_MarkerReader ∗ *markerReader,* uint64_t *markersToRead,* uint64_t ∗ *markersRead*)

Read a given number of markers via a marker reader.

Parameters

	reader	Reader handle.
	marker-Reader	Marker reader handle.
	marker-sToRead	Number markers to be read.
out	*marker-sRead*	Return pointer to the number of markers actually read.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.59 OTF2_ErrorCode OTF2_Reader_RegisterDefCallbacks (OTF2_Reader ∗ *reader,* OTF2_DefReader ∗ *defReader,* const OTF2_DefReaderCallbacks ∗ *callbacks,* void ∗ *userData*)

Register local definition reader callbacks.

Parameters

reader	OTF2_Reader handle.
defReader	Local definition reader handle.
callbacks	Callbacks for the local definition readers.
userData	Addition user data.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.60 OTF2_ErrorCode OTF2_Reader_RegisterEvtCallbacks (OTF2_Reader ∗ *reader,* OTF2_EvtReader ∗ *evtReader,* const OTF2_EvtReaderCallbacks ∗ *callbacks,* void ∗ *userData*)

Register event reader callbacks.

Parameters

reader	OTF2_Reader handle.
evtReader	Local event reader handle.
callbacks	Callbacks for the event readers.
userData	Addition user data.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.61 OTF2_ErrorCode OTF2_Reader_RegisterGlobalDefCallbacks (OTF2_Reader ∗ *reader*, OTF2_GlobalDefReader ∗ *defReader*, const OTF2_GlobalDefReaderCallbacks ∗ *callbacks*, void ∗ *userData*)

Register global definition reader callbacks.

Parameters

reader	OTF2_Reader handle.
defReader	Global definition reader handle.
callbacks	Callbacks for the global definition readers.
userData	Addition user data.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.62 OTF2_ErrorCode OTF2_Reader_RegisterGlobalEvtCallbacks (OTF2_Reader ∗ *reader*, OTF2_GlobalEvtReader ∗ *evtReader*, const OTF2_GlobalEvtReaderCallbacks ∗ *callbacks*, void ∗ *userData*)

Register global event reader callbacks.

Parameters

reader	OTF2_Reader handle.
evtReader	Global event reader handle.
callbacks	Callbacks for the global event reader.
userData	Addition user data.

Returns

Returns *OTF2_SUCCESS* if successful, an error code if an error occurs.

E.32.2.63 OTF2_ErrorCode OTF2_Reader_RegisterGlobalSnapCallbacks (OTF2_Reader ∗ *reader*, OTF2_GlobalSnapReader ∗ *evtReader*, const OTF2_GlobalSnapReaderCallbacks ∗ *callbacks*, void ∗ *userData*)

Register global event reader callbacks.

Parameters

reader	OTF2_Reader handle.

evtReader	Global event reader handle.
callbacks	Callbacks for the global event reader.
userData	Addition user data.

Since

Version 1.2

Returns

Returns *OTF2_SUCCESS* if successful, an error code if an error occurs.

E.32.2.64 OTF2_ErrorCode OTF2_Reader_RegisterMarkerCallbacks (OTF2_Reader ∗ *reader,* OTF2_MarkerReader ∗ *markerReader,* const OTF2_MarkerReaderCallbacks ∗ *callbacks,* void ∗ *userData*)

Register marker reader callbacks.

Parameters

reader	OTF2_Reader handle.
marker-Reader	Marker reader handle.
callbacks	Callbacks for the marker reader.
userData	Addition user data.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.65 OTF2_ErrorCode OTF2_Reader_RegisterSnapCallbacks (OTF2_Reader ∗ *reader,* OTF2_SnapReader ∗ *snapReader,* const OTF2_SnapReaderCallbacks ∗ *callbacks,* void ∗ *userData*)

Register snapshot event reader callbacks.

Parameters

reader	OTF2_Reader handle.

snapReader	Local snap reader handle.
callbacks	Callbacks for the event readers.
userData	Addition user data.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.66 OTF2_ErrorCode OTF2_Reader_SelectLocation (OTF2_Reader ∗ *reader*, OTF2_LocationRef *location*)

Select a location to be read.

Parameters

reader	Reader handle.
location	Location ID.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.67 OTF2_ErrorCode OTF2_Reader_SetCollectiveCallbacks (OTF2_Reader ∗ *reader*, const OTF2_CollectiveCallbacks ∗ *collectiveCallbacks*, void ∗ *collectiveData*, OTF2_CollectiveContext ∗ *globalCommContext*, OTF2_CollectiveContext ∗ *localCommContext*)

Set the collective callbacks for the reader.

The reader has as the default the serial collectives set.

This function is an collective operation.

Parameters

reader	Reader handle.

collective-Callbacks	Struct holding the collective callback functions.
collective-Data	Data passed to the collective callbacks in the `userData` argument.
global-CommCon-text	Global communication context.
local-CommCon-text	Local communication context. Unsued in reading mode. A local communication context may be created via the callbacks which fits the one used when the given trace was written.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.32.2.68 OTF2_ErrorCode OTF2_Reader_SetHint (OTF2_Reader * *reader*, OTF2_Hint *hint*, void * *value*)

Set the *hint* in the *reader* to the given *value*.

Hints can only be set once and only before OTF2 itself uses the hint the first time.

Parameters

reader	Reader handle.
hint	Name of the hint.
value	Reference to the hint value.

Since

Version 1.5

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT in case of NULL pointers for *archive* or *value*, or an unknown *hint* value

OTF2_ERROR_HINT_INVALID in case the hint is not valid for this handle

OTF2_ERROR_HINT_LOCKED in case the hint was already set or was queried at least once by the handle

OTF2_ERROR_HINT_INVALID_VALUE in case the provided value is invalid for this hint

E.32.2.69 OTF2_ErrorCode OTF2_Reader_SetLockingCallbacks (OTF2_Reader ∗ *reader,* const OTF2_LockingCallbacks ∗ *lockingCallbacks,* void ∗ *lockingData*)

Set the locking callbacks for the reader.

Can be called any time, but only once. Before this call no thread-safety is guaranteed.

Parameters

reader	Reader handle.
locking-Callbacks	Struct holding the locking callback functions.
lockingData	Data passed to the locking callbacks in the `userData` argument.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT in case of NULL pointers for *archive* or *lockingCallbacks*, or mandatory callbacks in *lockingCallbacks* are missing

OTF2_ERROR_INVALID_CALL in case there were locking callbacks already set

E.32.2.70 OTF2_ErrorCode OTF2_Reader_SetSerialCollectiveCallbacks (OTF2_Reader ∗ *reader*)

Convenient function to set the collective callbacks to an serial implementation.

Parameters

reader	Reader handle.

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.33 otf2/OTF2_SnapReader.h File Reference

This is the local snap reader, which reads snapshot events from one location.

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

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_SnapReaderCallbacks.h>
```

Functions

- OTF2_ErrorCode OTF2_SnapReader_GetLocationID (const OTF2_SnapReader ∗reader, OTF2_LocationRef ∗location)

  Return the location ID of the reading related location.
- OTF2_ErrorCode OTF2_SnapReader_ReadSnapshots (OTF2_SnapReader ∗reader, uint64_t recordsToRead, uint64_t ∗recordsRead)

  After callback registration, the local events could be read with the following function. Readn reads recordsToRead records. The reader indicates that it reached the end of the trace by just reading less records than requested.
- OTF2_ErrorCode OTF2_SnapReader_Seek (OTF2_SnapReader ∗reader, uint64_t req_time, bool ∗found)

  Seek jumps to start of latest snaphot that was made before a given time 'req_time'.
- OTF2_ErrorCode OTF2_SnapReader_SetCallbacks (OTF2_SnapReader ∗reader, const OTF2_SnapReaderCallbacks ∗callbacks, void ∗userData)

  Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

E.33.1 Detailed Description

This is the local snap reader, which reads snapshot events from one location.

E.33.2 Function Documentation

E.33.2.1 OTF2_ErrorCode OTF2_SnapReader_GetLocationID (const OTF2_SnapReader ∗ *reader*, OTF2_LocationRef ∗ *location*)

Return the location ID of the reading related location.

Parameters

	reader	Reader object which reads the snapshot events from its buffer.
out	*location*	ID of the location.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.33.2.2 OTF2_ErrorCode OTF2_SnapReader_ReadSnapshots (OTF2_SnapReader ∗ *reader,* uint64_t *recordsToRead,* uint64_t ∗ *recordsRead*)

After callback registration, the local events could be read with the following function. Readn reads *recordsToRead* records. The reader indicates that it reached the end of the trace by just reading less records than requested.

Parameters

	reader	Reader object which reads the events from its buffer.
	record-sToRead	How many records can be read next.
`out`	*record-sRead*	Return how many records where really read.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.33.2.3 OTF2_ErrorCode OTF2_SnapReader_Seek (OTF2_SnapReader ∗ *reader,* uint64_t *req_time,* bool ∗ *found*)

Seek jumps to start of latest snaphot that was made before a given time 'req_time'.

Parameters

reader	Reader object which reads the events from its buffer.
req_time	Requested time (see above)
found	returns if a matching snapshot was found

Since

Version 1.2

Returns

OTF2_ErrorCode with !=OTF2_SUCCESS if there was an error.

E.33.2.4 OTF2_ErrorCode OTF2_SnapReader_SetCallbacks (OTF2_SnapReader ∗ *reader*, const OTF2_SnapReaderCallbacks ∗ *callbacks*, void ∗ *userData*)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

These callbacks are ignored, if the events are read by an global event reader.

Parameters

reader	Reader object which reads the events from its buffer.
callbacks	Struct which holds a function pointer for each record type. *OTF2_-SnapReaderCallbacks_New*.
userData	Data passed as argument *userData* to the record callbacks.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.34 otf2/OTF2_SnapReaderCallbacks.h File Reference

This defines the callbacks for the snap reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

Typedefs

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_Enter)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_RegionRef region)

  Callback for the Enter snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MeasurementOnOff)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-MeasurementMode measurementMode)

  Callback for the MeasurementOnOff snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_Metric)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type ∗typeIDs, const OTF2_-MetricValue ∗metricValues)

  Callback for the Metric snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime)

  Callback for the MpiCollectiveBegin snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

  Callback for the MpiCollectiveEnd snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MpiIrecv)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_-t requestID)

  Callback for the MpiIrecv snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_-t requestID)

  Callback for the MpiIrecvRequest snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MpiIsend)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t re-

ceiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIsend snap event record.

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_-t requestID)

  Callback for the MpiIsendComplete snap event record.

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MpiRecv)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Callback for the MpiRecv snap event record.

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_MpiSend)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t re-ceiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Callback for the MpiSend snap event record.

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_OmpAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_-t lockID, uint32_t acquisitionOrder)

  Callback for the OmpAcquireLock snap event record.

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_OmpFork)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t num-berOfRequestedThreads)

  Callback for the OmpFork snap event record.

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_-t taskID)

  Callback for the OmpTaskCreate snap event record.

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_-t taskID)

  Callback for the OmpTaskSwitch snap event record.

- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_ParameterInt)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, int64_t value)

  Callback for the ParameterInt snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-ParameterRef parameter, OTF2_StringRef string)

  Callback for the ParameterString snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_-ParameterRef parameter, uint64_t value)

  Callback for the ParameterUnsignedInt snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_SnapshotEnd)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList, uint64_t contReadPos)

  Callback for the SnapshotEnd snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_SnapshotStart)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, uint64_t numberOfRecords)

  Callback for the SnapshotStart snap event record.
- typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_Unknown)(OTF2_-LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_-AttributeList ∗attributeList)

  Callback for an unknown snap event record.
- typedef struct OTF2_SnapReaderCallbacks_struct OTF2_SnapReaderCallbacks

  Opaque struct which holdes all snap event record callbacks.

Functions

- void OTF2_SnapReaderCallbacks_Clear (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks)

  Clears a struct for the sanp event callbacks.
- void OTF2_SnapReaderCallbacks_Delete (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks)

  Deallocates a struct for the snap event callbacks.
- OTF2_SnapReaderCallbacks ∗ OTF2_SnapReaderCallbacks_New (void)

Allocates a new struct for the snap event callbacks.

- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetEnterCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-Enter enterCallback)

  Registers the callback for the Enter snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMeasurementOnOffCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MeasurementOnOff measurementOnOffCallback)

  Registers the callback for the MeasurementOnOff snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMetricCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-Metric metricCallback)

  Registers the callback for the Metric snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MpiCollectiveBegin mpiCollectiveBeginCallback)

  Registers the callback for the MpiCollectiveBegin snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MpiCollectiveEnd mpiCollectiveEndCallback)

  Registers the callback for the MpiCollectiveEnd snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIrecvCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MpiIrecv mpiIrecvCallback)

  Registers the callback for the MpiIrecv snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MpiIrecvRequest mpiIrecvRequestCallback)

  Registers the callback for the MpiIrecvRequest snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIsendCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MpiIsend mpiIsendCallback)

  Registers the callback for the MpiIsend snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MpiIsendComplete mpiIsendCompleteCallback)

  Registers the callback for the MpiIsendComplete snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiRecvCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MpiRecv mpiRecvCallback)

  Registers the callback for the MpiRecv snap event.

- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiSendCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-MpiSend mpiSendCallback)

  Registers the callback for the MpiSend snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpAcquireLockCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-OmpAcquireLock ompAcquireLockCallback)

  Registers the callback for the OmpAcquireLock snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpForkCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-OmpFork ompForkCallback)

  Registers the callback for the OmpFork snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpTaskCreateCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-OmpTaskCreate ompTaskCreateCallback)

  Registers the callback for the OmpTaskCreate snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpTaskSwitchCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-OmpTaskSwitch ompTaskSwitchCallback)

  Registers the callback for the OmpTaskSwitch snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetParameterIntCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-ParameterInt parameterIntCallback)

  Registers the callback for the ParameterInt snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetParameterStringCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-ParameterString parameterStringCallback)

  Registers the callback for the ParameterString snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetParameterUnsignedIntCallback (OTF2_SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-ParameterUnsignedInt parameterUnsignedIntCallback)

  Registers the callback for the ParameterUnsignedInt snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetSnapshotEndCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-SnapshotEnd snapshotEndCallback)

  Registers the callback for the SnapshotEnd snap event.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetSnapshotStartCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-SnapshotStart snapshotStartCallback)

  Registers the callback for the SnapshotStart snap event.

- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetUnknownCallback (OTF2_-SnapReaderCallbacks ∗snapReaderCallbacks, OTF2_SnapReaderCallback_-Unknown unknownCallback)

  Registers the callback for the Unknown snap event.

E.34.1 Detailed Description

This defines the callbacks for the snap reader.

Source Template:

templates/OTF2_SnapReaderCallbacks.tmpl.h

E.34.2 Typedef Documentation

E.34.2.1 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-Enter)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_RegionRef region)

Callback for the Enter snap event record.

This record exists for each *Enter* event where the corresponding *Leave* event did not occur before the snapshot.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.2 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MeasurementOnOff)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_MeasurementMode measurementMode)

Callback for the MeasurementOnOff snap event record.

The last occurrence of an *MeasurementOnOff* event of this location, if any.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
measure-mentMode	Is the measurement turned on (*OTF2_MEASUREMENT_ON*) or off (*OTF2_MEASUREMENT_OFF*)?

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.3 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-Metric)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type ∗typeIDs, const OTF2_MetricValue ∗metricValues)

Callback for the Metric snap event record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

As an exception for metric classes where the metric mode detontes an *OTF2_-METRIC_VALUE_RELATIVE* mode the value indicates the accumulation of all previous metric values recorded.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
metric	Could be a metric class or a metric instance. References a *MetricClass*, or a *MetricInstance* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_METRIC* is available.
numberOf-Metrics	Number of metrics with in the set.
typeIDs	List of metric types. These types must match that of the corresponding *MetricMember* definitions.
metricVal-ues	List of metric values.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.4 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime)

Callback for the MpiCollectiveBegin snap event record.

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.5 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

Callback for the MpiCollectiveEnd snap event record.

Indicates that this location completed a collective operation localy but not all of the participating locations completed the operation yet. The corresponding *Mpi-CollectiveBeginSnap* record is still in the snapshot though.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
collec-tiveOp	Determines which collective operation it is.
communi-cator	Communicator References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
root	MPI rank of root in `communicator`.
sizeSent	Size of the sent message.
sizeRe-ceived	Size of the received message.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.6 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MpiIrecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIrecv snap event record.

This record exists for each *MpiIrecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
sender	MPI rank of sender in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.7 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)

Callback for the MpiIrecvRequest snap event record.

This record exists for each *MpiIrecvRequest* event where an corresponding *Mpi-Irecv* or *MpiRequestCancelled* event did not occur on this location before the snap-shot. Or the corresponding *MpiIrecv* did occurred (the *MpiIrecvSnap* record exists in the snapshot) but the matching receive message event did not occur on the re-mote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
requestID	ID of the requested receive

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.8 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MpiIsend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIsend snap event record.

This record exists for each *MpiIsend* event where an corresponding *MpiIsendCom-plete* or *MpiRequestCancelled* event did not occur on this location before the snap-shot. Or the corresponding *MpiIsendComplete* did occurred (the *MpiIsendCom-pleteSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. (This could either be an*MpiRecv* or an *MpiIrecv* event.)

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happened.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.9 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)

Callback for the MpiIsendComplete snap event record.

This record exists for each *MpiIsend* event where the corresponding *MpiIsendComplete* event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an *MpiRecv* or an *MpiIrecv* event.) .

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.

userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.10 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MpiRecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiRecv snap event record.

This record exists for each *MpiRecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
sender	MPI rank of sender in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.11 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-MpiSend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiSend snap event record.

This record exists for each *MpiSend* event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event. Note that it may so, that a previous *MpiIsend* with the same envelope than this one is neither completed not canceled yet, thus the matching receive may already occurred, but the matching couldn't be done yet.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.12 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-OmpAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the OmpAcquireLock snap event record.

This record exists for each *OmpAcquireLock* event where the corresponding *OmpReleaseLock* did not occurred before this snapshot yet.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
lockID	ID of the lock.
acquisi-tionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.13 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-OmpFork)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint32_t numberOfRequestedThreads)

Callback for the OmpFork snap event record.

This record exists for each *OmpFork* event where the corresponding *OmpJoin* did not occurred before this snapshot.

Parameters

location	The location where this snap happened.

snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
num-berOfRe-quest-edThreads	Requested size of the team.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.14 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_t taskID)

Callback for the OmpTaskCreate snap event record.

This record exists for each *OmpTaskCreate* event where the corresponding *Omp-TaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
taskID	Identifier of the newly created task instance.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.15 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_- OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, uint64_t taskID)

Callback for the OmpTaskSwitch snap event record.

This record exists for each *OmpTaskSwitch* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
taskID	Identifier of the now active task instance.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.16 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_- ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, int64_t value)

Callback for the ParameterInt snap event record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.17 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, OTF2_StringRef string)

Callback for the ParameterString snap event record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.

string	Value: Handle of a string definition References a *String* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_STRING* is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.18 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-ParameterUnsignedInt)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, uint64_t value)

Callback for the ParameterUnsignedInt snap event record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
origEvent-Time	The original time this event happended.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.19 typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_-SnapshotEnd)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t contReadPos)

Callback for the SnapshotEnd snap event record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use *OTF2_EvtReader_Seek* with `contReadPos` as the position.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
contRead-Pos	Position to continue reading in the event trace.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.20 typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_-SnapshotStart)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t numberOfRecords)

Callback for the SnapshotStart snap event record.

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one *SnapshotStart* record and closes with one *SnapshotEnd* record. All snapshot records inbetween are ordered by the `origEventTime`, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

Parameters

location	The location where this snap happened.
snapTime	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this snap.
num-berOfRecord	Number of snapshot event records in this snapshot. Excluding the *Snap-shotEnd* record.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.21 typedef OTF2_CallbackCode(∗ OTF2_SnapReaderCallback_-Unknown)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void ∗userData, OTF2_AttributeList ∗attributeList)

Callback for an unknown snap event record.

Parameters

location	The location where this event happened.
time	Snapshot time.
userData	User data as set by *OTF2_Reader_RegisterSnapCallbacks* or *OTF2_-SnapReader_SetCallbacks*.
attributeList	Additional attributes for this event.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.34.2.22 typedef struct OTF2_SnapReaderCallbacks_struct OTF2_SnapReaderCallbacks

Opaque struct which holdes all snap event record callbacks.

Since

Version 1.2

E.34.3 Function Documentation

E.34.3.1 void OTF2_SnapReaderCallbacks_Clear (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*)

Clears a struct for the sanp event callbacks.

Parameters

snapReaderCallbacks	Handle to a struct previously allocated with *OTF2_SnapReaderCallbacks_New*.

Since

Version 1.2

E.34.3.2 void OTF2_SnapReaderCallbacks_Delete (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*)

Deallocates a struct for the snap event callbacks.

Parameters

snapReaderCallbacks	Handle to a struct previously allocated with *OTF2_SnapReaderCallbacks_New*.

Since

Version 1.2

E.34.3.3 OTF2_SnapReaderCallbacks∗ OTF2_SnapReaderCallbacks_New (void)

Allocates a new struct for the snap event callbacks.

Since

Version 1.2

Returns

A newly allocated struct of type *OTF2_SnapReaderCallbacks*.

E.34.3.4 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetEnterCallback (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*, OTF2_SnapReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
enterCallback	Function which should be called for all *Enter* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.5 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMeasurementOnOffCallback (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*, OTF2_SnapReaderCallback_MeasurementOnOff *measurementOnOffCallback*)

Registers the callback for the MeasurementOnOff snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
measurementOnOffCallback	Function which should be called for all *MeasurementOnOff* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.6 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMetricCallback (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*, OTF2_SnapReaderCallback_Metric *metricCallback*)

Registers the callback for the Metric snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
metricCallback	Function which should be called for all *Metric* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.7 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*, OTF2_SnapReaderCallback_MpiCollectiveBegin *mpiCollectiveBeginCallback*)

Registers the callback for the MpiCollectiveBegin snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
mpiCollectiveBeginCallback	Function which should be called for all *MpiCollectiveBegin* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.8 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_MpiCollectiveEnd *mpiCollectiveEndCallback*)

Registers the callback for the MpiCollectiveEnd snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
mpiCollectiveEndCallback	Function which should be called for all *MpiCollectiveEnd* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.9 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIrecvCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_MpiIrecv *mpiIrecvCallback*)

Registers the callback for the MpiIrecv snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
mpiIrecvCallback	Function which should be called for all *MpiIrecv* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.10 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*, OTF2_SnapReaderCallback_MpiIrecvRequest *mpiIrecvRequestCallback*)

Registers the callback for the MpiIrecvRequest snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
mpiIrecvRequestCallback	Function which should be called for all *MpiIrecvRequest* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.11 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIsendCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_MpiIsend *mpiIsendCallback*)

Registers the callback for the MpiIsend snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
mpiIsendCallback	Function which should be called for all *MpiIsend* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.12 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_MpiIsendComplete *mpiIsendCompleteCallback*)

Registers the callback for the MpiIsendComplete snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
mpiIsendCompleteCallback	Function which should be called for all *MpiIsendComplete* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.13 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiRecvCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_MpiRecv *mpiRecvCallback*)

Registers the callback for the MpiRecv snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
mpiRecvCallback	Function which should be called for all *MpiRecv* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.14 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiSendCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_MpiSend *mpiSendCallback*)

Registers the callback for the MpiSend snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
mpiSendCallback	Function which should be called for all *MpiSend* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.15 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpAcquireLockCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_OmpAcquireLock *ompAcquireLockCallback*)

Registers the callback for the OmpAcquireLock snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
ompAcquireLockCallback	Function which should be called for all *OmpAcquireLock* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.16 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpForkCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
ompForkCallback	Function which should be called for all *OmpFork* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.17 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpTaskCreateCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_OmpTaskCreate *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
ompTaskCreateCallback	Function which should be called for all *OmpTaskCreate* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.18 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpTaskSwitchCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_OmpTaskSwitch *ompTaskSwitchCallback*)

Registers the callback for the OmpTaskSwitch snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
ompTaskSwitchCallback	Function which should be called for all *OmpTaskSwitch* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.19 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetParameterIntCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_ParameterInt *parameterIntCallback*)

Registers the callback for the ParameterInt snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
parameterIntCallback	Function which should be called for all *ParameterInt* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.20 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetParameterStringCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_ParameterString *parameterStringCallback*)

Registers the callback for the ParameterString snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
parameterStringCallback	Function which should be called for all *ParameterString* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.21 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetParameterUnsignedIntCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_ParameterUnsignedInt *parameterUnsignedIntCallback*)

Registers the callback for the ParameterUnsignedInt snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
parameterUnsignedIntCallback	Function which should be called for all *ParameterUnsignedInt* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.22 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetSnapshotEndCallback (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*, OTF2_SnapReaderCallback_SnapshotEnd *snapshotEndCallback*)

Registers the callback for the SnapshotEnd snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.
snapshotEndCallback	Function which should be called for all *SnapshotEnd* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.23 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetSnapshotStartCallback (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*, OTF2_SnapReaderCallback_SnapshotStart *snapshotStartCallback*)

Registers the callback for the SnapshotStart snap event.

Parameters

snapReaderCallbacks	Struct for all callbacks.

snapshot-StartCall-back	Function which should be called for all *SnapshotStart* definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.34.3.24 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetUnknownCallback (OTF2_SnapReaderCallbacks ∗ *snapReaderCallbacks*, OTF2_SnapReaderCallback_Unknown *unknownCallback*)

Registers the callback for the Unknown snap event.

Parameters

snapRead-erCallbacks	Struct for all callbacks.
unknown-Callback	Function which should be called for all unknown snap events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.35 otf2/OTF2_SnapWriter.h File Reference

This lowest user-visible layer provides write routines to write snapshot records for a single location.

`#include <stdint.h>`

E.35 otf2/OTF2_SnapWriter.h File Reference

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_AttributeList.h>
```

Typedefs

- typedef struct OTF2_SnapWriter_struct OTF2_SnapWriter

  Keeps all necessary information about the snap writer. See OTF2_SnapWriter_-struct for detailed information.

Functions

- OTF2_ErrorCode OTF2_SnapWriter_Enter (OTF2_SnapWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_RegionRef region)

  Records an Enter snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_GetLocationID (const OTF2_SnapWriter ∗writer, OTF2_LocationRef ∗locationID)

  Function to get the location ID of a snap writer object.
- OTF2_ErrorCode OTF2_SnapWriter_MeasurementOnOff (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_-TimeStamp origEventTime, OTF2_MeasurementMode measurementMode)

  Records an MeasurementOnOff snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_Metric (OTF2_SnapWriter ∗writer, OTF2_-AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type ∗typeIDs, const OTF2_MetricValue ∗metricValues)

  Records an Metric snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_MpiCollectiveBegin (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_-TimeStamp origEventTime)

  Records an MpiCollectiveBegin snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_MpiCollectiveEnd (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_-TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)

  Records an MpiCollectiveEnd snapshot record.

- OTF2_ErrorCode OTF2_SnapWriter_MpiIrecv (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

  Records an MpiIrecv snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_MpiIrecvRequest (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_-TimeStamp origEventTime, uint64_t requestID)

  Records an MpiIrecvRequest snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_MpiIsend (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

  Records an MpiIsend snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_MpiIsendComplete (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_-TimeStamp origEventTime, uint64_t requestID)

  Records an MpiIsendComplete snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_MpiRecv (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Records an MpiRecv snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_MpiSend (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

  Records an MpiSend snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_OmpAcquireLock (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_-TimeStamp origEventTime, uint32_t lockID, uint32_t acquisitionOrder)

  Records an OmpAcquireLock snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_OmpFork (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint32_t numberOfRequestedThreads)

  Records an OmpFork snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_OmpTaskCreate (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint64_t taskID)

  Records an OmpTaskCreate snapshot record.

- OTF2_ErrorCode OTF2_SnapWriter_OmpTaskSwitch (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint64_t taskID)

  Records an OmpTaskSwitch snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_ParameterInt (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, int64_t value)

  Records an ParameterInt snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_ParameterString (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, OTF2_StringRef string)

  Records an ParameterString snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_ParameterUnsignedInt (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, OTF2_-TimeStamp origEventTime, OTF2_ParameterRef parameter, uint64_t value)

  Records an ParameterUnsignedInt snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_SnapshotEnd (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, uint64_-t contReadPos)

  Records an SnapshotEnd snapshot record.
- OTF2_ErrorCode OTF2_SnapWriter_SnapshotStart (OTF2_SnapWriter ∗writer, OTF2_AttributeList ∗attributeList, OTF2_TimeStamp snapTime, uint64_-t numberOfRecords)

  Records an SnapshotStart snapshot record.

E.35.1 Detailed Description

This lowest user-visible layer provides write routines to write snapshot records for a single location.

Source Template:

templates/OTF2_SnapWriter.tmpl.h

E.35.2 Typedef Documentation

E.35.2.1 typedef struct OTF2_SnapWriter_struct OTF2_SnapWriter

Keeps all necessary information about the snap writer. See OTF2_SnapWriter_-struct for detailed information.

Since

Version 1.2

E.35.3 Function Documentation

E.35.3.1 OTF2_ErrorCode OTF2_SnapWriter_Enter (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, OTF2_RegionRef *region*)

Records an Enter snapshot record.

This record exists for each *Enter* event where the corresponding *Leave* event did not occur before the snapshot.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
region	Needs to be defined in a definition record References a *Region* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_REGION* is available.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.2 OTF2_ErrorCode OTF2_SnapWriter_GetLocationID (const OTF2_SnapWriter ∗ *writer*, OTF2_LocationRef ∗ *locationID*)

Function to get the location ID of a snap writer object.

Parameters

writer	Snap writer object of interest
locationID	Pointer to a variable where the ID is returned in

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.3 OTF2_ErrorCode OTF2_SnapWriter_MeasurementOnOff (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, OTF2_MeasurementMode *measurementMode*)

Records an MeasurementOnOff snapshot record.

The last occurrence of an *MeasurementOnOff* event of this location, if any.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
measure-mentMode	Is the measurement turned on (*OTF2_MEASUREMENT_ON*) or off (*OTF2_MEASUREMENT_OFF*)?

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.4 OTF2_ErrorCode OTF2_SnapWriter_Metric (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, OTF2_MetricRef *metric*, uint8_t *numberOfMetrics*, const OTF2_Type ∗ *typeIDs*, const OTF2_MetricValue ∗ *metricValues*)

Records an Metric snapshot record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

As an exception for metric classes where the metric mode detontes an *OTF2_-METRIC_VALUE_RELATIVE* mode the value indicates the accumulation of all previous metric values recorded.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
metric	Could be a metric class or a metric instance. References a *MetricClass*, or a *MetricInstance* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_METRIC* is available.
numberOf-Metrics	Number of metrics with in the set.
typeIDs	List of metric types. These types must match that of the corresponding *MetricMember* definitions.
metricVal-ues	List of metric values.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.5 OTF2_ErrorCode OTF2_SnapWriter_MpiCollectiveBegin (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*)

Records an MpiCollectiveBegin snapshot record.

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.6 OTF2_ErrorCode OTF2_SnapWriter_MpiCollectiveEnd (OTF2_SnapWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, OTF2_CollectiveOp *collectiveOp*, OTF2_CommRef *communicator*, uint32_t *root*, uint64_t *sizeSent*, uint64_t *sizeReceived*)

Records an MpiCollectiveEnd snapshot record.

Indicates that this location completed a collective operation localy but not all of the participating locations completed the operation yet. The corresponding *MpiCollectiveBeginSnap* record is still in the snapshot though.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEventTime	The original time this event happended.
collectiveOp	Determines which collective operation it is.
communicator	Communicator References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_COMM* is available.
root	MPI rank of root in `communicator`.
sizeSent	Size of the sent message.
sizeReceived	Size of the received message.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.7 OTF2_ErrorCode OTF2_SnapWriter_MpiIrecv (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, uint32_t *sender*, OTF2_CommRef *communicator*, uint32_t *msgTag*, uint64_t *msgLength*, uint64_t *requestID*)

Records an MpiIrecv snapshot record.

This record exists for each *MpiIrecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
sender	MPI rank of sender in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.8 OTF2_ErrorCode OTF2_SnapWriter_MpiIrecvRequest (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, uint64_t *requestID*)

Records an MpiIrecvRequest snapshot record.

This record exists for each *MpiIrecvRequest* event where an corresponding *Mpi-Irecv* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIrecv* did occurred (the *MpiIrecvSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
requestID	ID of the requested receive

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.9 OTF2_ErrorCode OTF2_SnapWriter_MpiIsend (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, uint32_t *receiver*, OTF2_CommRef *communicator*, uint32_t *msgTag*, uint64_t *msgLength*, uint64_t *requestID*)

Records an MpiIsend snapshot record.

This record exists for each *MpiIsend* event where an corresponding *MpiIsendComplete* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIsendComplete* did occurred (the *MpiIsendCompleteSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. (This could either be an*MpiRecv* or an *MpiIrecv* event.)

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.

msgTag	Message tag
msgLength	Message length
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.10 OTF2_ErrorCode OTF2_SnapWriter_MpiIsendComplete (OTF2_SnapWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *snapTime,* OTF2_TimeStamp *origEventTime,* uint64_t *requestID*)

Records an MpiIsendComplete snapshot record.

This record exists for each *MpiIsend* event where the corresponding *MpiIsendComplete* event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an *MpiRecv* or an *MpiIrecv* event.) .

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEventTime	The original time this event happended.
requestID	ID of the related request

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.11 OTF2_ErrorCode OTF2_SnapWriter_MpiRecv (OTF2_SnapWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, uint32_t *sender*, OTF2_CommRef *communicator*, uint32_t *msgTag*, uint64_t *msgLength*)

Records an MpiRecv snapshot record.

This record exists for each *MpiRecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
sender	MPI rank of sender in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.12 OTF2_ErrorCode OTF2_SnapWriter_MpiSend (OTF2_SnapWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, uint32_t *receiver*, OTF2_CommRef *communicator*, uint32_t *msgTag*, uint64_t *msgLength*)

Records an MpiSend snapshot record.

This record exists for each *MpiSend* event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event. Note that it may so, that a previous *MpiIsend* with the same envelope than this one is neither completed not canceled yet, thus the matching receive may already occurred, but the matching couldn't be done yet.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
receiver	MPI rank of receiver in `communicator`.
communi-cator	Communicator ID. References a *Comm* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-COMM* is available.
msgTag	Message tag
msgLength	Message length

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.13 OTF2_ErrorCode OTF2_SnapWriter_OmpAcquireLock (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, uint32_t *lockID*, uint32_t *acquisitionOrder*)

Records an OmpAcquireLock snapshot record.

This record exists for each *OmpAcquireLock* event where the corresponding *OmpReleaseLock* did not occurred before this snapshot yet.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
lockID	ID of the lock.

acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.14 OTF2_ErrorCode OTF2_SnapWriter_OmpFork (OTF2_SnapWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *snapTime,* OTF2_TimeStamp *origEventTime,* uint32_t *numberOfRequestedThreads*)

Records an OmpFork snapshot record.

This record exists for each *OmpFork* event where the corresponding *OmpJoin* did not occurred before this snapshot.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEventTime	The original time this event happended.
numberOfRequestedThreads	Requested size of the team.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.15 OTF2_ErrorCode OTF2_SnapWriter_OmpTaskCreate (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, uint64_t *taskID*)

Records an OmpTaskCreate snapshot record.

This record exists for each *OmpTaskCreate* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEventTime	The original time this event happended.
taskID	Identifier of the newly created task instance.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.16 OTF2_ErrorCode OTF2_SnapWriter_OmpTaskSwitch (OTF2_SnapWriter ∗ *writer*, OTF2_AttributeList ∗ *attributeList*, OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*, uint64_t *taskID*)

Records an OmpTaskSwitch snapshot record.

This record exists for each *OmpTaskSwitch* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEventTime	The original time this event happended.
taskID	Identifier of the now active task instance.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.17 OTF2_ErrorCode OTF2_SnapWriter_ParameterInt (OTF2_SnapWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *snapTime,* OTF2_TimeStamp *origEventTime,* OTF2_ParameterRef *parameter,* int64_t *value*)

Records an ParameterInt snapshot record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.18 OTF2_ErrorCode OTF2_SnapWriter_ParameterString (OTF2_SnapWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *snapTime,* OTF2_TimeStamp *origEventTime,* OTF2_ParameterRef *parameter,* OTF2_StringRef *string*)

Records an ParameterString snapshot record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.
parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
string	Value: Handle of a string definition References a *String* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_STRING* is available.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.19 OTF2_ErrorCode OTF2_SnapWriter_ParameterUnsignedInt (OTF2_SnapWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *snapTime,* OTF2_TimeStamp *origEventTime,* OTF2_ParameterRef *parameter,* uint64_t *value*)

Records an ParameterUnsignedInt snapshot record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
origEvent-Time	The original time this event happended.

parameter	Parameter ID. References a *Parameter* definition and will be mapped to the global definition if a mapping table of type *OTF2_MAPPING_-PARAMETER* is available.
value	Value of the recorded parameter.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.20 OTF2_ErrorCode OTF2_SnapWriter_SnapshotEnd (OTF2_SnapWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *snapTime,* uint64_t *contReadPos*)

Records an SnapshotEnd snapshot record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use *OTF2_EvtReader_Seek* with `contReadPos` as the position.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
contRead-Pos	Position to continue reading in the event trace.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.35.3.21 OTF2_ErrorCode OTF2_SnapWriter_SnapshotStart (OTF2_SnapWriter ∗ *writer,* OTF2_AttributeList ∗ *attributeList,* OTF2_TimeStamp *snapTime,* uint64_t *numberOfRecords*)

Records an SnapshotStart snapshot record.

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one *SnapshotStart* record and closes with one *SnapshotEnd* record. All snapshot records inbetween are ordered by the `origEventTime`, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

Parameters

writer	Writer object.
attributeList	Generic attributes for the snap.
snapTime	Snapshot time.
num-berOfRecord	Number of snapshot event records in this snapshot. Excluding the *Snap-shotEnd* record.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.36 otf2/OTF2_Thumbnail.h File Reference

This lowest user-visible layer provides write routines to read and write thumbnail data.

`#include <stdint.h>`

`#include <otf2/OTF2_GeneralDefinitions.h>`

Typedefs

- typedef struct OTF2_ThumbReader_struct OTF2_ThumbReader

  Keeps all necessary information about the event reader. See OTF2_ThumbReader_-struct for detailed information.

- typedef struct OTF2_ThumbWriter_struct OTF2_ThumbWriter

  Keeps all necessary information about the thumb writer. See OTF2_ThumbWriter_-struct for detailed information.

E.36 otf2/OTF2_Thumbnail.h File Reference

Functions

- OTF2_ErrorCode OTF2_ThumbReader_GetHeader (OTF2_ThumbReader ∗reader, char ∗∗const name, char ∗∗const description, OTF2_ThumbnailType ∗type, uint32_t ∗numberOfSamples, uint32_t ∗numberOfMetrics, uint64_t ∗∗refsToDefs)

  Reads a thumbnail header.
- OTF2_ErrorCode OTF2_ThumbReader_ReadSample (OTF2_ThumbReader ∗reader, uint64_t ∗baseline, uint32_t numberOfMetrics, uint64_t ∗metricSamples)

  Reads a thumbnail sample.
- OTF2_ErrorCode OTF2_ThumbWriter_WriteSample (OTF2_ThumbWriter ∗writer, uint64_t baseline, uint32_t numberOfMetrics, const uint64_t ∗metricSamples)

  Writes a thumbnail sample.

E.36.1 Detailed Description

This lowest user-visible layer provides write routines to read and write thumbnail data.

E.36.2 Function Documentation

E.36.2.1 OTF2_ErrorCode OTF2_ThumbReader_GetHeader (OTF2_ThumbReader ∗ *reader,* char ∗∗const *name,* char ∗∗const *description,* OTF2_ThumbnailType ∗ *type,* uint32_t ∗ *numberOfSamples,* uint32_t ∗ *numberOfMetrics,* uint64_t ∗∗ *refsToDefs*)

Reads a thumbnail header.

A thumbnail header contains some meta information for a thumbnail.

Parameters

	reader	Reader object.
out	*name*	Name of thumbnail.
out	*description*	Description of thumbnail.
out	*type*	Type of thumbnail.
out	*numberOf-Samples*	Number of samples.
out	*numberOf-Metrics*	Number of metrics.
out	*refsToDefs*	The sorted set of references to definitions used in an thumbnail sample.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.36.2.2 OTF2_ErrorCode OTF2_ThumbReader_ReadSample (OTF2_ThumbReader ∗ *reader*, uint64_t ∗ *baseline*, uint32_t *numberOfMetrics*, uint64_t ∗ *metricSamples*)

Reads a thumbnail sample.

Parameters

	reader	Reader object.
out	*baseline*	Baseline for this sample. If zero, the baseline is the sum of all metric values in this sample.
	numberOf-Metrics	Number of metric sample values.
out	*metricSam-ples*	Metric sample values.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

E.36.2.3 OTF2_ErrorCode OTF2_ThumbWriter_WriteSample (OTF2_ThumbWriter ∗ *writer*, uint64_t *baseline*, uint32_t *numberOfMetrics*, const uint64_t ∗ *metricSamples*)

Writes a thumbnail sample.

Parameters

writer	Writer object.
baseline	Baseline for this sample. If zero, the baseline is the sum of all metric values in this sample.
numberOf-Metrics	Number of metric sample values.

metricSamples	Metric sample values.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful, an error code if an error occurs.

Index