

OPEN TRACE FORMAT 2 USER MANUAL

1.4 (revision 3826)

Fri Aug 29 2014 12:03:38

OTF2 LICENSE AGREEMENT

COPYRIGHT ©2009-2012,
RWTH Aachen University, Germany
COPYRIGHT ©2009-2012,
Gesellschaft fuer numerische Simulation mbH, Germany
COPYRIGHT ©2009-2013,
Technische Universitaet Dresden, Germany
COPYRIGHT ©2009-2012,
University of Oregon, Eugene, USA
COPYRIGHT ©2009-2013,
Forschungszentrum Juelich GmbH, Germany
COPYRIGHT ©2009-2013,
German Research School for Simulation Sciences GmbH, Germany
COPYRIGHT ©2009-2012,
Technische Universitaet Muenchen, Germany

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the names of
RWTH Aachen University,
Gesellschaft fuer numerische Simulation mbH Braunschweig,
Technische Universitaet Dresden,
University of Oregon, Eugene,
Forschungszentrum Juelich GmbH,
German Research School for Simulation Sciences GmbH, or the
Technische Universitaet Muenchen,
nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

	Page
Contents	v
1 Open Trace Format 2	1
1.1 Introduction	1
1.2 Get started	1
Appendix A OTF2 INSTALL	5
Appendix B Deprecated List	15
Appendix C Module Documentation	17
C.1 Usage of OTF2 tools	17
C.2 OTF2 config tool	17
C.3 OTF2 print tool	18
C.4 OTF2 snapshots tool	19
C.5 OTF2 marker tool	19
C.6 OTF2 estimator tool	20
C.7 OTF2 records	21
C.8 List of all definition records	22
C.9 ClockProperties	22
C.10 MappingTable	23
C.11 ClockOffset	23
C.12 String	24
C.13 Attribute	24
C.14 SystemTreeNode	25
C.15 LocationGroup	25
C.16 Location	26
C.17 Region	27
C.18 Callsite	27
C.19 Callpath	28
C.20 Group	28
C.21 MetricMember	29
C.22 MetricClass	30
C.23 MetricInstance	31

CONTENTS

C.24 Comm	32
C.25 Parameter	33
C.26 RmaWin	33
C.27 MetricClassRecorder	34
C.28 SystemTreeNodeProperty	34
C.29 SystemTreeNodeDomain	35
C.30 LocationGroupProperty	35
C.31 LocationProperty	36
C.32 CartDimension	36
C.33 CartTopology	37
C.34 CartCoordinate	38
C.35 List of all event records	39
C.36 BufferFlush	39
C.37 MeasurementOnOff	39
C.38 Enter	40
C.39 Leave	40
C.40 MpiSend	41
C.41 MpiIsend	41
C.42 MpiIsendComplete	42
C.43 MpiIrecvRequest	42
C.44 MpiRecv	43
C.45 MpiIrecv	44
C.46 MpiRequestTest	44
C.47 MpiRequestCancelled	45
C.48 MpiCollectiveBegin	45
C.49 MpiCollectiveEnd	46
C.50 OmpFork	46
C.51 OmpJoin	47
C.52 OmpAcquireLock	47
C.53 OmpReleaseLock	48
C.54 OmpTaskCreate	49
C.55 OmpTaskSwitch	49
C.56 OmpTaskComplete	50
C.57 Metric	51
C.58 ParameterString	51
C.59 ParameterInt	52
C.60 ParameterUnsignedInt	53
C.61 RmaWinCreate	53
C.62 RmaWinDestroy	54
C.63 RmaCollectiveBegin	54
C.64 RmaCollectiveEnd	55
C.65 RmaGroupSync	55
C.66 RmaRequestLock	56
C.67 RmaAcquireLock	57

CONTENTS

C.68 RmaTryLock	57
C.69 RmaReleaseLock	58
C.70 RmaSync	59
C.71 RmaWaitChange	59
C.72 RmaPut	60
C.73 RmaGet	60
C.74 RmaAtomic	61
C.75 RmaOpCompleteBlocking	62
C.76 RmaOpCompleteNonBlocking	62
C.77 RmaOpTest	63
C.78 RmaOpCompleteRemote	63
C.79 ThreadFork	64
C.80 ThreadJoin	65
C.81 ThreadTeamBegin	65
C.82 ThreadTeamEnd	66
C.83 ThreadAcquireLock	66
C.84 ThreadReleaseLock	67
C.85 ThreadTaskCreate	67
C.86 ThreadTaskSwitch	68
C.87 ThreadTaskComplete	69
C.88 ThreadCreate	69
C.89 ThreadBegin	70
C.90 ThreadWait	70
C.91 ThreadEnd	71
C.92 List of all marker records	72
C.93 DefMarker	72
C.94 Marker	72
C.95 List of all snapshot records	73
C.96 SnapshotStart	73
C.97 SnapshotEnd	73
C.98 MeasurementOnOffSnap	74
C.99 EnterSnap	74
C.100MpiSendSnap	75
C.101MpiIsendSnap	76
C.102MpiIsendCompleteSnap	77
C.103MpiRecvSnap	77
C.104MpiIrecvRequestSnap	78
C.105MpiIrecvSnap	79
C.106MpiCollectiveBeginSnap	79
C.107MpiCollectiveEndSnap	80
C.108OmpForkSnap	81
C.109OmpAcquireLockSnap	81
C.110OmpTaskCreateSnap	82
C.111OmpTaskSwitchSnap	83

C.112MetricSnap	83
C.113ParameterStringSnap	84
C.114ParameterIntSnap	85
C.115ParameterUnsignedIntSnap	85
C.116OTF2 usage examples	86
C.117Usage in writing mode - a simple example	86
C.118How to use the attribute list for writing additional attributes to event records	91
C.119OTF2 callbacks	91
C.120Controlling OTF2 flush behavior in writing mode	91
C.120.1Detailed Description	92
C.120.2Typedef Documentation	92
C.121Memory pooling for OTF2	93
C.121.1Detailed Description	94
C.121.2Typedef Documentation	94
C.122Operating OTF2 in an collective context	95
C.122.1Detailed Description	97
C.122.2Typedef Documentation	97
C.123Usage in reading mode - MPI example	101
C.124Usage in writing mode - MPI example	108
C.125Usage in reading mode - a simple example	115
Appendix D Data Structure Documentation	121
D.1 OTF2_AttributeValue Union Reference	121
D.1.1 Detailed Description	122
D.2 OTF2_CollectiveCallbacks Struct Reference	123
D.2.1 Detailed Description	123
D.3 OTF2_CollectiveContext Struct Reference	123
D.3.1 Detailed Description	123
D.4 OTF2_FlushCallbacks Struct Reference	123
D.4.1 Detailed Description	124
D.5 OTF2_MemoryCallbacks Struct Reference	124
D.5.1 Detailed Description	124
D.6 OTF2_MetricValue Union Reference	125
D.6.1 Detailed Description	125
D.7 OTF2_MPI_UserData Struct Reference	125
D.7.1 Detailed Description	125
Appendix E File Documentation	127
E.1 otf2/OTF2_ErrorCodes.h File Reference	127
E.1.1 Detailed Description	131
E.1.2 Typedef Documentation	131
E.1.3 Enumeration Type Documentation	131
E.1.4 Function Documentation	135

CONTENTS

E.2	otf2/otf2.h File Reference	136
E.2.1	Detailed Description	136
E.3	otf2/OTF2_Archive.h File Reference	136
E.3.1	Detailed Description	142
E.3.2	Define Documentation	142
E.3.3	Typedef Documentation	142
E.3.4	Function Documentation	143
E.4	otf2/OTF2_AttributeList.h File Reference	167
E.4.1	Detailed Description	172
E.4.2	Function Documentation	172
E.5	otf2/OTF2_Callbacks.h File Reference	192
E.5.1	Detailed Description	194
E.6	otf2/OTF2_Definitions.h File Reference	195
E.6.1	Detailed Description	200
E.6.2	Enumeration Type Documentation	200
E.7	otf2/OTF2_DefReader.h File Reference	209
E.7.1	Detailed Description	210
E.7.2	Function Documentation	210
E.8	otf2/OTF2_DefReaderCallbacks.h File Reference	212
E.8.1	Detailed Description	218
E.8.2	Typedef Documentation	218
E.8.3	Function Documentation	236
E.9	otf2/OTF2_DefWriter.h File Reference	250
E.9.1	Detailed Description	253
E.9.2	Function Documentation	253
E.10	otf2/OTF2_Events.h File Reference	269
E.10.1	Detailed Description	272
E.10.2	Enumeration Type Documentation	272
E.11	otf2/OTF2_EventSizeEstimator.h File Reference	275
E.11.1	Detailed Description	281
E.11.2	Function Documentation	281
E.12	otf2/OTF2_EvtReader.h File Reference	310
E.12.1	Detailed Description	311
E.12.2	Function Documentation	311
E.13	otf2/OTF2_EvtReaderCallbacks.h File Reference	315
E.13.1	Detailed Description	328
E.13.2	Typedef Documentation	328
E.13.3	Function Documentation	370
E.14	otf2/OTF2_EvtWriter.h File Reference	402
E.14.1	Detailed Description	409
E.14.2	Function Documentation	409
E.15	otf2/OTF2_GeneralDefinitions.h File Reference	448
E.15.1	Detailed Description	455
E.15.2	Enumeration Type Documentation	455

E.16	otf2/OTF2_GlobalDefReader.h File Reference	461
E.16.1	Detailed Description	462
E.16.2	Function Documentation	462
E.17	otf2/OTF2_GlobalDefReaderCallbacks.h File Reference	463
E.17.1	Detailed Description	469
E.17.2	Typedef Documentation	469
E.17.3	Function Documentation	486
E.18	otf2/OTF2_GlobalDefWriter.h File Reference	502
E.18.1	Detailed Description	505
E.18.2	Function Documentation	505
E.19	otf2/OTF2_GlobalEvtReader.h File Reference	522
E.19.1	Detailed Description	523
E.19.2	Function Documentation	523
E.20	otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference	525
E.20.1	Detailed Description	538
E.20.2	Typedef Documentation	538
E.20.3	Function Documentation	577
E.21	otf2/OTF2_GlobalSnapReader.h File Reference	611
E.21.1	Detailed Description	612
E.21.2	Function Documentation	612
E.22	otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference	613
E.22.1	Detailed Description	619
E.22.2	Typedef Documentation	619
E.22.3	Function Documentation	636
E.23	otf2/OTF2_IdMap.h File Reference	649
E.23.1	Detailed Description	650
E.23.2	Typedef Documentation	650
E.23.3	Enumeration Type Documentation	651
E.23.4	Function Documentation	651
E.24	otf2/OTF2_Marker.h File Reference	655
E.24.1	Detailed Description	656
E.24.2	Enumeration Type Documentation	656
E.25	otf2/OTF2_MarkerReader.h File Reference	657
E.25.1	Detailed Description	658
E.25.2	Function Documentation	658
E.26	otf2/OTF2_MarkerReaderCallbacks.h File Reference	659
E.26.1	Detailed Description	660
E.26.2	Typedef Documentation	661
E.26.3	Function Documentation	662
E.27	otf2/OTF2_MarkerWriter.h File Reference	665
E.27.1	Detailed Description	666
E.27.2	Function Documentation	666
E.28	otf2/OTF2_MPI_Collectives.h File Reference	667
E.28.1	Detailed Description	669

CONTENTS

E.28.2	Function Documentation	669
E.29	otf2/OTF2_Reader.h File Reference	670
E.29.1	Detailed Description	676
E.29.2	Function Documentation	676
E.30	otf2/OTF2_SnapReader.h File Reference	704
E.30.1	Detailed Description	705
E.30.2	Function Documentation	705
E.31	otf2/OTF2_SnapReaderCallbacks.h File Reference	707
E.31.1	Detailed Description	712
E.31.2	Typedef Documentation	713
E.31.3	Function Documentation	729
E.32	otf2/OTF2_SnapWriter.h File Reference	741
E.32.1	Detailed Description	744
E.32.2	Typedef Documentation	744
E.32.3	Function Documentation	745
E.33	otf2/OTF2_Thumbnail.h File Reference	759
E.33.1	Detailed Description	760
E.33.2	Function Documentation	760

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.

Configuration of OTF2

'configure' configures scorep 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'.

APPENDIX A. OTF2 INSTALL

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-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

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>.

APPENDIX A. OTF2 INSTALL

Installation Instructions

Copyright (C) 1994, 1995, 1996, 1999, 2000, 2001, 2002, 2004, 2005,
2006, 2007, 2008, 2009 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved. This file is offered as-is,
without warranty of any kind.

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 be 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
```

APPENDIX A. OTF2 INSTALL

*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'.

APPENDIX A. OTF2 INSTALL

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

APPENDIX A. OTF2 INSTALL

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 *estimator
In version 1.2

APPENDIX B. DEPRECATED LIST

Global **OTF2_EventSizeEstimator_GetSizeOfOmpTaskSwitchEvent**(OTF2_EventSizeEstimator *estimator,

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 *attributeList,

In version 1.2

Global **OTF2_EvtWriter_OmpTaskCreate**(OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, C

In version 1.2

Global **OTF2_EvtWriter_OmpTaskSwitch**(OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, C

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: <ul style="list-style-type: none">* substrates* compressions
--help	prints this usage information

APPENDIX C. MODULE DOCUMENTATION

```
--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:

```
--backend
                on systems, which required cross-compiling, this flag
                specifies that the information for the backend is displayed.
                By default the information for the frontend is displayed.
                On non-cross compiling systems, this flag is ignored
--nvcc
                specifies that the required flags are for the CUDA compiler
                nvcc
```

C.3 OTF2 print tool

A call to otf2-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 from 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
-d, --debug          turn on debug mode
-V, --version        print version information
-h, --help           print this help information
```

C.4 OTF2 snapshots tool

C.4 OTF2 snapshots tool

A call to oft2-snapshots has the following syntax:

Usage: oft2-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: oft2-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>

APPENDIX C. MODULE DOCUMENTATION

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 otf2-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.
------------------	-----------------------------------

C.7 OTF2 records

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

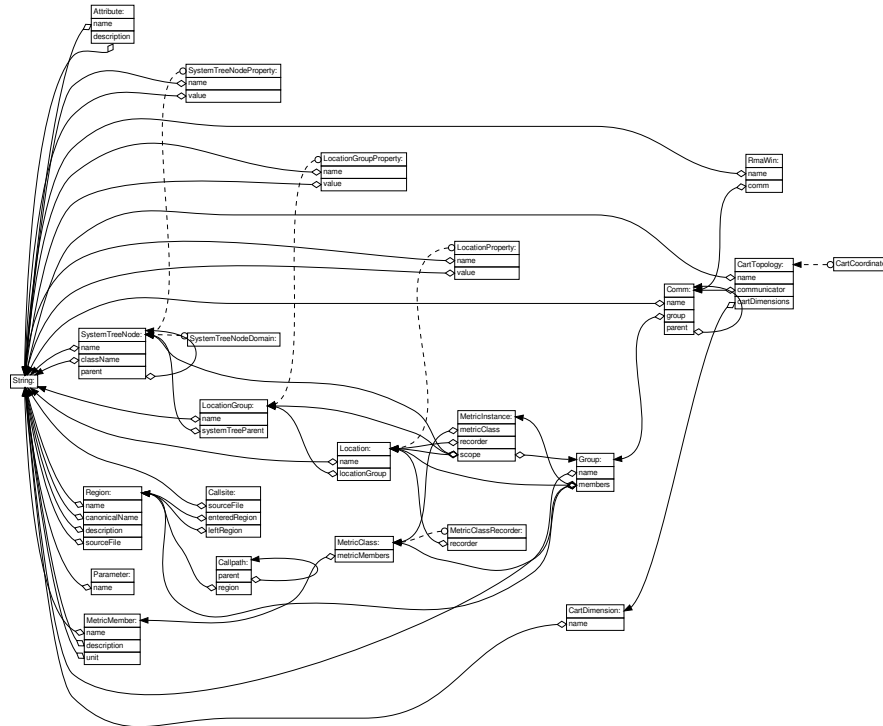
```
cat <<EOC | otf2-estimator
set Region 4
set Metric 1
get Timestamp
get Enter
get Leave
get Metric 4
exit
EOC
Timestamp 9
Enter 3
Leave 3
Metric 4 44
```

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	traceLength	A timespan which includes the timespan between the smallest and greatest timestamp of all event timestamps.

C.11 ClockOffset

See also

[OTF2_GlobalDefWriter_WriteClockProperties\(\)](#)

Since

Version 1.0

C.10 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.11 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 <code>time</code> .
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.12 [OTF2_StringRef](#) String

The string definitions.

Attributes

OTF2_StringRef	const char*	string	The string, null terminated.
--------------------------------	-------------	--------	------------------------------

See also

[OTF2_GlobalDefWriter_WriteString\(\)](#)

[OTF2_DefWriter_WriteString\(\)](#)

Since

Version 1.0

C.13 [OTF2_AttributeRef](#) Attribute

The attribute definition.

Attributes

OTF2_AttributeRef	name	Name of the attribute. References a String definition.
OTF2_AttributeRef	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\(\)](#)

C.15 LocationGroup

Since

Version 1.0

C.14 *OTF2_SystemTreeNodeRef* SystemTreeNode

The system tree node definition.

Attributes

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

Supplements

SystemTreeNodeProperty
SystemTreeNodeDomain

See also

OTF2_GlobalDefWriter_WriteSystemTreeNode()
OTF2_DefWriter_WriteSystemTreeNode()

Since

Version 1.0

C.15 *OTF2_LocationGroupRef* LocationGroup

The location group definition.

Attributes

<i>OTF2_StringRef</i>	name	Name of the group. References a <i>String</i> definition.
<i>OTF2_LocationGroupType</i>	location-GroupType	Type of this group.

APPENDIX C. MODULE DOCUMENTATION

<i>OTF2_-SystemTreeNodeRef</i>	sys- temTreePar- ent	Parent of this location group in the sys- tem tree. References a <i>SystemTreeNode</i> definition.
--	----------------------------	--

Supplements

[*LocationGroupProperty*](#)

See also

[OTF2_GlobalDefWriter_WriteLocationGroup\(\)](#)
[OTF2_DefWriter_WriteLocationGroup\(\)](#)

Since

Version 1.0

C.16 [*OTF2_LocationRef*](#) Location

The location definition.

Attributes

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

Supplements

[*LocationProperty*](#)

See also

[OTF2_GlobalDefWriter_WriteLocation\(\)](#)
[OTF2_DefWriter_WriteLocation\(\)](#)

C.18 Callsite

Since

Version 1.0

C.17 *OTF2_RegionRef* Region

The region definition.

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteRegion\(\)](#)
[OTF2_DefWriter_WriteRegion\(\)](#)

Since

Version 1.0

C.18 *OTF2_CallsiteRef* Callsite

The callsite definition.

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteCallsite\(\)](#)
[OTF2_DefWriter_WriteCallsite\(\)](#)

Since

Version 1.0

C.19 [*OTF2_CallpathRef*](#) Callpath

The callpath definition.

Attributes

<i>OTF2_CallpathRef</i>	parent	The parent of this callpath. References a <i>Callpath</i> definition.
<i>OTF2_RegionRef</i>	region	The region of this callpath. References a <i>Region</i> definition.

See also

[OTF2_GlobalDefWriter_WriteCallpath\(\)](#)
[OTF2_DefWriter_WriteCallpath\(\)](#)

Since

Version 1.0

C.20 [*OTF2_GroupRef*](#) Group

The group definition.

C.21 MetricMember

Attributes

<i>OTF2_StringRef</i>	name	Name of this group References a <i>String</i> definition.
<i>OTF2_GroupType</i>	groupType	The type of this group. Since version 1.2.
<i>OTF2_Paradigm</i>	paradigm	The paradigm of this communication group. Since version 1.2.
<i>OTF2_GroupFlag</i>	groupFlags	Flags for this group. Since version 1.2.
uint32_t	num-berOfMem-bers	The number of members in this group.
uint64_t	members [num-berOfMem-bers]	The identifiers of the group members.

See also

[OTF2_GlobalDefWriter_WriteGroup\(\)](#)
[OTF2_DefWriter_WriteGroup\(\)](#)

Since

Version 1.0

C.21 [*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

<i>OTF2_StringRef</i>	name	Name of the metric. References a <i>String</i> definition.
<i>OTF2_StringRef</i>	description	Description of the metric. References a <i>String</i> definition.
<i>OTF2_MetricType</i>	metricType	Metric type: PAPI, etc.
<i>OTF2_MetricMode</i>	metric-Mode	Metric mode: accumulative, fix, relative, etc.

APPENDIX C. MODULE DOCUMENTATION

<i>OTF2_Type</i>	valueType	Type of the value. Only <i>OTF2_TYPE_INT64</i> , <i>OTF2_TYPE_UINT64</i> , and <i>OTF2_TYPE_DOUBLE</i> are valid types. If this metric member is recorded in an <i>Metric</i> event, than this type and the type in the event must match.
<i>OTF2_MetricBase</i>	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 $\text{base}^{\text{exponent}}$, to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be <i>OTF2_BASE_BINARY</i> 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.
<i>OTF2_StringRef</i>	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 <i>String</i> definition.

See also

[*OTF2_GlobalDefWriter_WriteMetricMember\(\)*](#)
[*OTF2_DefWriter_WriteMetricMember\(\)*](#)

Since

Version 1.0

C.22 [*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.

C.23 MetricInstance

Attributes

<code>uint8_t</code>	<code>numberOfMetrics</code>	Number of metrics within the set.
<i>OTF2_MetricMemberRef</i>	<code>metricMembers</code> [<code>numberOfMetrics</code>]	List of metric members. References a <i>MetricMember</i> definition.
<i>OTF2_MetricOccurrence</i>	<code>metricOccurrence</code>	Defines occurrence of a metric set.
<i>OTF2_RecorderKind</i>	<code>recorderKind</code>	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.23 [*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

<i>OTF2_MetricRef</i>	<code>metricClass</code>	The instanced <i>MetricClass</i> . This metric class must be of kind <i>OTF2_RECORDER_KIND_ABSTRACT</i> . References a <i>MetricClass</i> definition.
<i>OTF2_LocationRef</i>	<code>recorder</code>	Recorder of the metric: location ID. References a <i>Location</i> definition.

APPENDIX C. MODULE DOCUMENTATION

<i>OTF2_MetricScope</i>	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.24 [*OTF2_CommRef*](#) Comm

The communicator definition.

Attributes

<i>OTF2_StringRef</i>	name	The name given by calling <code>MPI_Comm_set_name</code> on this communicator. Or the empty name to indicate that no name was given. References a <i>String</i> definition.
<i>OTF2_GroupRef</i>	group	The describing MPI group of this MPI communicator The group needs to be of type <i>OTF2_GROUP_TYPE_COMM_GROUP</i> or <i>OTF2_GROUP_TYPE_COMM_SELF</i> . References a <i>Group</i> definition.
<i>OTF2_CommRef</i>	parent	The parent MPI communicator from which this communicator was created, if any. Use <i>OTF2_UNDEFINED_COMM</i> to indicate no parent. References a <i>Comm</i> definition.

See also

[OTF2_GlobalDefWriter_WriteComm\(\)](#)
[OTF2_DefWriter_WriteComm\(\)](#)

C.26 RmaWin

Since

Version 1.0

C.25 *OTF2_ParameterRef* Parameter

The parameter definition.

Attributes

<i>OTF2_StringRef</i>	name	Name of the parameter (variable name etc.) References a <i>String</i> definition.
<i>OTF2_ParameterType</i>	parameter-Type	Type of the parameter, <i>OTF2_ParameterType</i> for possible types.

See also

[OTF2_GlobalDefWriter_WriteParameter\(\)](#)
[OTF2_DefWriter_WriteParameter\(\)](#)

Since

Version 1.0

C.26 *OTF2_RmaWinRef* RmaWin

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

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteRmaWin\(\)](#)
[OTF2_DefWriter_WriteRmaWin\(\)](#)

Since

Version 1.2

C.27 MetricClassRecorder

The metric class recorder definition.

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteMetricClassRecorder\(\)](#)
[OTF2_DefWriter_WriteMetricClassRecorder\(\)](#)

Since

Version 1.2

C.28 SystemTreeNodeProperty

An arbitrary key/value property for a [*SystemTreeNode*](#) definition.

Attributes

<i>OTF2_SystemTreeNodeRef</i>	systemTreeNode	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>OTF2_StringRef</i>	name	Name of the property. References a <i>String</i> definition.
<i>OTF2_StringRef</i>	value	Property value. References a <i>String</i> definition.

C.30 LocationGroupProperty

See also

[OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty\(\)](#)
[OTF2_DefWriter_WriteSystemTreeNodeProperty\(\)](#)

Since

Version 1.2

C.29 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_-SystemTreeNodeDomain	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.30 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.
--	--------------------	---

<i>OTF2_StringRef</i>	name	Name of the property. References a <i>String</i> definition.
<i>OTF2_StringRef</i>	value	Property value. References a <i>String</i> definition.

See also

[OTF2_GlobalDefWriter_WriteLocationGroupProperty\(\)](#)
[OTF2_DefWriter_WriteLocationGroupProperty\(\)](#)

Since

Version 1.3

C.31 LocationProperty

An arbitrary key/value property for a [*Location*](#) definition.

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteLocationProperty\(\)](#)
[OTF2_DefWriter_WriteLocationProperty\(\)](#)

Since

Version 1.3

C.32 [*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.

C.33 CartTopology

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.33 [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.34 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

<i>OTF2-CartTopologyRef</i>	cartTopology	Parent <i>CartTopology</i> definition to which this one is a supplementary definition. References a <i>CartTopology</i> 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 MeasurementOnOff

C.35 List of all event records

C.36 BufferFlush

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

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	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.37 MeasurementOnOff

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

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	timestamp	The time when this event happened.
<i>OTF2_MeasurementMode</i>	measurementMode	Is the measurement turned on (<i>OTF2_MEASUREMENT_ON</i>) or off (<i>OTF2_MEASUREMENT_OFF</i>)?

See also

[*OTF2_EvtWriter_MeasurementOnOff\(\)*](#)

Since

Version 1.0

C.38 Enter

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

Attributes

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

See also

[*OTF2_EvtWriter_Enter\(\)*](#)

Since

Version 1.0

C.39 Leave

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

Attributes

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

See also

[*OTF2_EvtWriter_Leave\(\)*](#)

Since

Version 1.0

C.41 MpISend

C.40 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	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

[OTF2_EvtWriter_MpiSend\(\)](#)

Since

Version 1.0

C.41 MpISend

A MpISend 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.

<i>OTF2_CommRef</i>	communi- cator	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length
uint64_t	requestID	ID of the related request

See also

[*OTF2_EvtWriter_MpiSend\(\)*](#)

Since

Version 1.0

C.42 MpiSendComplete

Signals the completion of non-blocking send request.

Attributes

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

See also

[*OTF2_EvtWriter_MpiSendComplete\(\)*](#)

Since

Version 1.0

C.43 MpiRecvRequest

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

C.44 MpiRecv

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_MpiRecvRequest\(\)](#)

Since

Version 1.0

C.44 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	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

[OTF2_EvtWriter_MpiRecv\(\)](#)

Since

Version 1.0

C.45 MpiIrecv

A MpiIrecv record indicates that a MPI message was received (MPI_I_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	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_MpiIrecv\(\)](#)

Since

Version 1.0

C.46 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\(\)](#)

C.48 **MpiCollectiveBegin**

Since

Version 1.0

C.47 **MpiRequestCancelled**

This events appears if the program canceled a request.

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	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.48 **MpiCollectiveBegin**

A **MpiCollectiveBegin** record marks the begin of an MPI collective operation (MPI_GATHER, MPI_SCATTER etc.).

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	timestamp	The time when this event happened.

See also

[*OTF2_EvtWriter_MpiCollectiveBegin\(\)*](#)

Since

Version 1.0

C.49 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.50 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.

C.52 OmpAcquireLock

See also

[OTF2_EvtWriter_OmpFork\(\)](#)

Since

Version 1.0

Deprecated

In version 1.2

C.51 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.52 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

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	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.53 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

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	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.

C.55 OmpTaskSwitch

See also

[OTF2_EvtWriter_OmpReleaseLock\(\)](#)

Since

Version 1.0

Deprecated

In version 1.2

C.54 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.55 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.56 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.58 ParameterString

C.57 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.58 ParameterString

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

APPENDIX C. MODULE DOCUMENTATION

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	timestamp	The time when this event happened.
<i>OTF2_- ParameterRef</i>	parameter	Parameter ID. References a <i>Parameter</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_PARAMETER</i> is available.
<i>OTF2_StringRef</i>	string	Value: Handle of a string definition References a <i>String</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_- STRING</i> is available.

See also

[*OTF2_EvtWriter_ParameterString\(\)*](#)

Since

Version 1.0

C.59 ParameterInt

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

Attributes

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

See also

[*OTF2_EvtWriter_ParameterInt\(\)*](#)

C.61 RmaWinCreate

Since

Version 1.0

C.60 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.61 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.62 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 <i>RmaWin</i> 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.63 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\(\)](#)

C.65 RmaGroupSync

Since

Version 1.2

C.64 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.65 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.
----------------------------------	----------	---

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

See also

[*OTF2_EvtWriter_RmaGroupSync\(\)*](#)

Since

Version 1.2

C.66 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

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

C.68 RmaTryLock

See also

[OTF2_EvtWriter_RmaRequestLock\(\)](#)

Since

Version 1.2

C.67 RmaAcquireLock

An RmaAcquireLock record denotes the time a lock was aquired 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 aquired, if multiple locks are defined on a window.
OTF2_LockType	lockType	Type of lock aquired.

See also

[OTF2_EvtWriter_RmaAcquireLock\(\)](#)

Since

Version 1.2

C.68 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.
----------------------------------	----------	---

<i>OTF2_TimeStamp</i>	timestamp	The time when this event happened.
<i>OTF2_RmaWinRef</i>	win	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
uint32_t	remote	Rank of the locked remote process.
uint64_t	lockId	ID of the lock aquired, if multiple locks are defined on a window.
<i>OTF2_LockType</i>	lockType	Type of lock aquired.

See also

[*OTF2_EvtWriter_RmaTryLock\(\)*](#)

Since

Version 1.2

C.69 RmaReleaseLock

An RmaReleaseLock record denotes the time the lock was released.

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	timestamp	The time when this event happened.
<i>OTF2_RmaWinRef</i>	win	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> 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.71 RmaWaitChange

C.70 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.71 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.72 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 appropriate completion record.

See also

[OTF2_EvtWriter_RmaPut\(\)](#)

Since

Version 1.2

C.73 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.

C.74 RmaAtomic

uint32_t	remote	Rank of the target process.
uint64_t	bytes	Bytes received from target.
uint64_t	matchingId	ID used for matching the appropriate completion record.

See also

[OTF2_EvtWriter_RmaGet\(\)](#)

Since

Version 1.2

C.74 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	bytesReceived	Bytes received from target.
uint64_t	matchingId	ID used for matching the appropriate completion record.

See also

[OTF2_EvtWriter_RmaAtomic\(\)](#)

Since

Version 1.2

C.75 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 appropriate completion record.

See also

[OTF2_EvtWriter_RmaOpCompleteBlocking\(\)](#)

Since

Version 1.2

C.76 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 appropriate completion record.

C.78 RmaOpCompleteRemote

See also

[OTF2_EvtWriter_RmaOpCompleteNonBlocking\(\)](#)

Since

Version 1.2

C.77 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 appropriate completion record.

See also

[OTF2_EvtWriter_RmaOpTest\(\)](#)

Since

Version 1.2

C.78 RmaOpCompleteRemote

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

Attributes

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

<i>OTF2_RmaWinRef</i>	win	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
uint64_t	matchingId	ID used for matching the appropriate completion record.

See also

[*OTF2_EvtWriter_RmaOpCompleteRemote\(\)*](#)

Since

Version 1.2

C.79 ThreadFork

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

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	timestamp	The time when this event happened.
<i>OTF2_Paradigm</i>	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.81 ThreadTeamBegin

C.80 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.81 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.82 ThreadTeamEnd

The current location leaves the specified thread team.

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	timestamp	The time when this event happened.
<i>OTF2_CommRef</i>	threadTeam	Thread team References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.

See also

[OTF2_EvtWriter_ThreadTeamEnd\(\)](#)

Since

Version 1.2

C.83 ThreadAcquireLock

An ThreadAcquireLock record marks that a thread acquires an lock.

Attributes

<i>OTF2_LocationRef</i>	location	The location where this event happened.
<i>OTF2_TimeStamp</i>	timestamp	The time when this event happened.
<i>OTF2_Paradigm</i>	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\(\)](#)

C.85 ThreadTaskCreate

Since

Version 1.2

C.84 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.85 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	creat- ingThread	Creating thread of this task.
uint32_t	genera- tionNumber	Thread-private generation number of task's creating thread.

See also

[OTF2_EvtWriter_ThreadTaskCreate\(\)](#)

Since

Version 1.2

C.86 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	creat- ingThread	Creating thread of this task.
uint32_t	genera- tionNumber	Thread-private generation number of task's creating thread.

See also

[OTF2_EvtWriter_ThreadTaskSwitch\(\)](#)

Since

Version 1.2

C.88 ThreadCreate

C.87 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.88 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.89 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.90 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.

C.91 ThreadEnd

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.91 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	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 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.92 List of all marker records

C.93 *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", ...
<i>OTF2_MarkerSeverity</i>	severity	The severity for these markers.

See also

[*OTF2_MarkerWriter_WriteDefMarker\(\)*](#)

Since

Version 1.2

C.94 Marker

A user marker instance, with implied time stamp.

Attributes

<i>OTF2_TimeStamp</i>	timestamp	The time when this marker happened.
<i>OTF2_TimeStamp</i>	duration	A possible duration of this marker. May be 0.
<i>OTF2_MarkerRef</i>	marker	Groups this marker by name and severity. References a <i>DefMarker</i> definition.
<i>OTF2_MarkerScope</i>	scope	The type of scope of this marker instance.
uint64_t	scopeRef	The scope instance of this marker. Depends on <i>scope</i> .
const char*	text	A textual description for this marker.

See also

[*OTF2_MarkerWriter_WriteMarker\(\)*](#)

C.97 SnapshotEnd

Since

Version 1.2

C.95 List of all snapshot records

C.96 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	num-berOfRecord	Number of snapshot event records in this snapshot. Excluding the SnapshotEnd record.

See also

[OTF2_SnapWriter_SnapshotStart\(\)](#)

Since

Version 1.2

C.97 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.

APPENDIX C. MODULE DOCUMENTATION

uint64_t	contRead-Pos	Position to continue reading in the event trace.
----------	--------------	--

See also

[OTF2_SnapWriter_SnapshotEnd\(\)](#)

Since

Version 1.2

C.98 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 happened.
OTF2-MeasurementMode	measurementMode	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.99 EnterSnap

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

Attributes

C.100 MpiSendSnap

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 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

[Enter](#) event

[OTF2_SnapWriter_Enter\(\)](#)

Since

Version 1.2

C.100 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 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

[MpiSend](#) event
[OTF2_SnapWriter_MpiSend\(\)](#)

Since

Version 1.2

C.101 MpisendSnap

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 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

[MpiIsend](#) event
[OTF2_SnapWriter_MpiIsend\(\)](#)

Since

Version 1.2

C.103 MpiRecvSnap

C.102 MpisendCompleteSnap

This record exists for each *MpiSend* event where the corresponding *MpiSendComplete* 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

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	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

MpiSendComplete event
OTF2_SnapWriter_MpiSendComplete()

Since

Version 1.2

C.103 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 *MpiSendComplete* 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

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	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.
<i>OTF2_CommRef</i>	communi-cator	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.

uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length

See also

[MpiRecv](#) event
[OTF2_SnapWriter_MpiRecv\(\)](#)

Since

Version 1.2

C.104 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	origEvent-Time	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.106 MpiCollectiveBeginSnap

C.105 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

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.
uint32_t	sender	MPI rank of sender in communicator.
<i>OTF2_CommRef</i>	communi-cator	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> 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.106 MpiCollectiveBeginSnap

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

Attributes

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
-------------------------	----------	-------------------------------

<i>OTF2_TimeStamp</i>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.

See also

[*MpiCollectiveBegin*](#) event
[OTF2_SnapWriter_MpiCollectiveBegin\(\)](#)

Since

Version 1.2

C.107 MpiCollectiveEndSnap

Indicates that this location completed a collective operation locally but not all of the participating locations completed the operation yet. The corresponding [*MpiCollectiveBeginSnap*](#) record is still in the snapshot though.

Attributes

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

See also

[*MpiCollectiveEnd*](#) event
[OTF2_SnapWriter_MpiCollectiveEnd\(\)](#)

C.109 OmpAcquireLockSnap

Since

Version 1.2

C.108 OmpForkSnap

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

Attributes

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	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.109 OmpAcquireLockSnap

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

Attributes

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	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	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

[*OmpAcquireLock*](#) event
[OTF2_SnapWriter_OmpAcquireLock\(\)](#)

Since

Version 1.2

C.110 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

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	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.112 MetricSnap

C.111 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	origEvent-Time	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.112 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 <i>MetricClass</i> , or a <i>MetricInstance</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING-METRIC</i> is available.

APPENDIX C. MODULE DOCUMENTATION

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 <i>MetricMember</i> definitions.
OTF2_MetricValue	metricValues [numberOfMetrics]	List of metric values.

See also

Metric event
`OTF2_SnapWriter_Metric()`

Since

Version 1.2

C.113 ParameterStringSnap

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

Attributes

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.
<i>OTF2-ParameterRef</i>	parameter	Parameter ID. References a <i>Parameter</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_PARAMETER</i> is available.
<i>OTF2_StringRef</i>	string	Value: Handle of a string definition References a <i>String</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_STRING</i> is available.

C.115 ParameterUnsignedIntSnap

See also

[ParameterString](#) event
[OTF2_SnapWriter_ParameterString\(\)](#)

Since

Version 1.2

C.114 ParameterIntSnap

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest 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 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

[ParameterInt](#) event
[OTF2_SnapWriter_ParameterInt\(\)](#)

Since

Version 1.2

C.115 ParameterUnsignedIntSnap

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

Attributes

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.
<i>OTF2_ParameterRef</i>	parameter	Parameter ID. References a <i>Parameter</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_PARAMETER</i> is available.
uint64_t	value	Value of the recorded parameter.

See also

[*ParameterUnsignedInt*](#) event
[OTF2_SnapWriter_ParameterUnsignedInt\(\)](#)

Since

Version 1.2

C.116 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.117 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`.

First include the OTF2 header.

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

For this example an additional include statement is necessary.

C.117 Usage in writing mode - a simple example

```
#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 suppress 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 spent 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,
```

APPENDIX C. MODULE DOCUMENTATION

```
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 collectively.

C.117 Usage in writing mode - a simple example

```
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( archive );
```

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 */ );
```

APPENDIX C. MODULE DOCUMENTATION

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

C.118 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_UINT8, 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.119 OTF2 callbacks

Modules

- [Controlling OTF2 flush behavior in writing mode](#)
- [Memory pooling for OTF2](#)
- [Operating OTF2 in an collective context](#)

C.120 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.120.1 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.120.2 Typedef Documentation

C.120.2.1 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

<i>userData</i>	Data passed to the call OTF2_Archive_SetFlushCallbacks .
<i>fileType</i>	The file type for which the flush has happened.
<i>location</i>	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.121 Memory pooling for OTF2

C.120.2.2 `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

<i>userData</i>	Data passed to the call OTF2_Archive_SetFlushCallbacks .
<i>fileType</i>	The type of file for what this buffer holds data.
<i>location</i>	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 <i>callerData</i> argument.
<i>callerData</i>	Depending of the fileType, this can be an OTF2_EvtWriter , OTF2_GlobalDefWriter , OTF2_DefWriter .
<i>final</i>	Indicates whether this is the final flush when closing the writer objects.

Returns

Returns [OTF2_FLUSH](#) or [OTF2_NO_FLUSH](#).

C.121 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.

APPENDIX C. MODULE DOCUMENTATION

- `typedef void(* OTF2_MemoryFreeAll)(void *userData, OTF2_FileType fileType, OTF2_LocationRef location, void **perBufferData, bool final)`

Function pointer to release all allocated chunks.

C.121.1 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.

C.121.2 Typedef Documentation

- C.121.2.1 `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 'chunkSize' provided with [OTF2_Archive_Open\(\)](#).

Parameters

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

Returns

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

C.122 Operating OTF2 in an collective context

C.121.2.2 `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

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

C.122 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.

C.122 Operating OTF2 in an collective context

- 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.122.1 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.

C.122.2 Typedef Documentation

C.122.2.1 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.122.2.2 `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.122.2.3 `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.122 Operating OTF2 in an collective context

C.122.2.4 `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.122.2.5 `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.122.2.6 `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.122.2.7 `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.122.2.8 `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.122.2.9 `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.123 Usage in reading mode - MPI example

C.122.2.10 `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.122.2.11 `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.123 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>
```

APPENDIX C. MODULE DOCUMENTATION

And then 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 assume 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 describing `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 );
}
```

C.123 Usage in reading mode - MPI example

```
        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 definitions 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 environment and query the size and rank.

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

APPENDIX C. MODULE DOCUMENTATION

```
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( reader );
```

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 );
```

C.123 Usage in reading mode - MPI example

Read all global definitions. Everytime a location definition is read, the previously 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 supposed 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 definition 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 definitions 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 )
    {
```

APPENDIX C. MODULE DOCUMENTATION

```
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 occurred in the trace. Everytime an enter or leave event is read, the

C.123 Usage in reading mode - MPI example

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 environment.

```
    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.124 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 then 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 assume 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

C.124 Usage in writing mode - MPI example

to return either OTF2_FLUSH to flush the recorded data to a file or OTF2_NO_FLUSH to suppress 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 spent 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 environment 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",
```

APPENDIX C. MODULE DOCUMENTATION

```
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.

C.124 Usage in writing mode - MPI example

```
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 collectively.

```
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.

APPENDIX C. MODULE DOCUMENTATION

```
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 );
```

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 */,
```

C.124 Usage in writing mode - MPI example

```
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 location 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.

APPENDIX C. MODULE DOCUMENTATION

```
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 enumerating 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` communicator. 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 environment, and exit.

C.125 Usage in reading mode - a simple example

```
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.125 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,
```

APPENDIX C. MODULE DOCUMENTATION

```
        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 definitions 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;
}
```


C.125 Usage in reading mode - a simple example

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( reader );
```

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,
```

APPENDIX C. MODULE DOCUMENTATION

```
                                global_def_callbacks,  
                                locations );  
OTF2_GlobalDefReaderCallbacks_Delete( global_def_callbacks );
```

Read all global definitions. Everytime a location definition is read, the previously 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 supposed 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 definition reader handle can be requested. In this example for all. To apply mapping tables stored in the local definitions, the local definitions 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 );  
        }  
    }  
}
```

C.125 Usage in reading mode - a simple example

```
    }  
  }  
  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( 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 occurred 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.

APPENDIX C. MODULE DOCUMENTATION

```
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_AttributeList.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_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_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_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_AttributeList.h](#)

D.2 OTF2_CollectiveCallbacks Struct Reference

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_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.5.1 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.6 OTF2_MetricValue Union Reference

D.6 OTF2_MetricValue Union Reference

Metric value.

```
#include <otf2/OTF2_Events.h>
```

D.6.1 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.7 OTF2_MPI_UserData Struct Reference

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

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

D.7.1 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](#)

APPENDIX D. DATA STRUCTURE DOCUMENTATION

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,

E.1 otf2/OTF2_ErrorCodes.h File Reference

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,

APPENDIX E. FILE DOCUMENTATION

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 }

E.1 otf2/OTF2_ErrorCodes.h File Reference

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

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

OTF2_ERROR_EINPROGRESS Operation is work in progress

OTF2_ERROR_EINTR Interruption 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 Too many layers of symbolic links

OTF2_ERROR_EMFILE Too many opened files

E.1 otf2/OTF2_ErrorCodes.h File Reference

OTF2_ERROR_EMLINK To many links

OTF2_ERROR_MSGSIZE 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

APPENDIX E. FILE DOCUMENTATION

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 Timeout 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

E.1 otf2/OTF2_ErrorCodes.h File Reference

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

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

<i>errorCode</i>	: 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

<i>errorCode</i>	: 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

<i>errorCallbackIn</i>	: Function will be called instead of printing a default message to standard error.
<i>userData</i>	: 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>
```

E.3 otf2/OTF2_Archive.h File Reference

```
#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_StatusCode OTF2_Archive_Close (OTF2_Archive *archive)`
Close an opened archive.
- `OTF2_StatusCode OTF2_Archive_CloseDefFiles (OTF2_Archive *archive)`
Closes the local definitions file container.
- `OTF2_StatusCode OTF2_Archive_CloseDefReader (OTF2_Archive *archive, OTF2_DefReader *reader)`
Close an opened local definition reader.
- `OTF2_StatusCode OTF2_Archive_CloseDefWriter (OTF2_Archive *archive, OTF2_DefWriter *writer)`
Close an opened local definition writer.
- `OTF2_StatusCode OTF2_Archive_CloseEvtFiles (OTF2_Archive *archive)`
Closes the events file container.
- `OTF2_StatusCode 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)

E.3 otF2/OTF2_Archive.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- [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)

E.3 otf2/OTF2_Archive.h File Reference

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_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 otf2/OTF2_Archive.h File Reference

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

<i>archive</i>	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

<i>archive</i>	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

<i>archive</i>	Archive handle.
<i>reader</i>	Reader handle to be closed.

APPENDIX E. FILE DOCUMENTATION

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.3.4.4 **OTF2_StatusCode** **OTF2_Archive_CloseDefWriter** (**OTF2_Archive** * *archive*,
OTF2_DefWriter * *writer*)

Close an opened local definition writer.

Parameters

<i>archive</i>	Archive handle.
<i>writer</i>	Writer handle to be closed.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.3.4.5 **OTF2_StatusCode** **OTF2_Archive_CloseEvtFiles** (**OTF2_Archive** * *archive*)

Closes the events file container.

This function is an collective operation.

Parameters

<i>archive</i>	Archive handle.
----------------	-----------------

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.3.4.6 **OTF2_StatusCode** **OTF2_Archive_CloseEvtReader** (**OTF2_Archive** * *archive*,
OTF2_EvtReader * *reader*)

Close an opened local event reader.

Parameters

<i>archive</i>	Archive handle.
<i>reader</i>	Reader handle to be closed.

E.3 otf2/OTF2_Archive.h File Reference

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

<i>archive</i>	Archive handle.
<i>writer</i>	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

<i>archive</i>	Archive handle.
<i>globalDef-Reader</i>	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

<i>archive</i>	Archive handle.
<i>writer</i>	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

<i>archive</i>	Archive handle.
<i>globalEvtReader</i>	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

<i>archive</i>	Archive handle.
<i>globalSnapReader</i>	Reader handle to be closed.

Since

Version 1.2

Returns

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

E.3 otf2/OTF2_Archive.h File Reference

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

Closes the marker reader.

Parameters

<i>archive</i>	Archive handle.
<i>marker-Reader</i>	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

<i>archive</i>	Archive handle.
<i>writer</i>	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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>archive</i>	Archive handle.
----------------	-----------------

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.3.4.15 **OTF2_StatusCode** **OTF2_Archive_CloseSnapReader** (**OTF2_Archive ***
archive, **OTF2_SnapReader *** ***reader***)

Close an opened local snap reader.

Parameters

<i>archive</i>	Archive handle.
<i>reader</i>	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_StatusCode** **OTF2_Archive_CloseSnapWriter** (**OTF2_Archive ***
archive, **OTF2_SnapWriter *** ***writer***)

Close an opened local snap writer.

Parameters

<i>archive</i>	Archive handle.
<i>writer</i>	Writer handle to be closed.

Since

Version 1.2

E.3 otf2/OTF2_Archive.h File Reference

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

<i>archive</i>	Archive handle.
<i>reader</i>	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

	<i>archive</i>	Archive handle.
out	<i>chunk-SizeEvents</i>	Chunk size for event files.
out	<i>chunk-SizeDefs</i>	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)

APPENDIX E. FILE DOCUMENTATION

Parameters

	<i>archive</i>	Archive handle.
out	<i>compression</i>	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

	<i>archive</i>	Archive handle.
out	<i>creator</i>	Returned creator. Allocated with <i>malloc</i> .

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

<i>archive</i>	Archive handle.
<i>location</i>	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.

E.3 otf2/OTF2_Archive.h File Reference

Parameters

<i>archive</i>	Archive handle.
<i>location</i>	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

	<i>archive</i>	Archive handle.
out	<i>description</i>	Returned description. Allocated with <i>malloc</i> .

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

<i>archive</i>	Archive handle.
<i>location</i>	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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>archive</i>	Archive handle.
<i>location</i>	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

	<i>archive</i>	Archive handle.
out	<i>substrate</i>	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

<i>archive</i>	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.

E.3 otf2/OTF2_Archive.h File Reference

Parameters

<i>archive</i>	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

<i>archive</i>	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

<i>archive</i>	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.

APPENDIX E. FILE DOCUMENTATION

Parameters

	<i>archive</i>	Archive handle.
out	<i>machine-Name</i>	Returned machine name. Allocated with <i>malloc</i> .

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

<i>archive</i>	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

<i>archive</i>	Archive handle.
----------------	-----------------

Since

Version 1.2

Returns

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

E.3 otf2/OTF2_Archive.h File Reference

E.3.4.34 **OTF2_ErrorCode** **OTF2_Archive_GetNumberOfGlobalDefinitions** (**OTF2_Archive** * *archive*, uint64_t * *numberOfDefinitions*)

Get the number of global definitions.

Parameters

	<i>archive</i>	Archive handle.
out	<i>numberOfDefinitions</i>	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

	<i>archive</i>	Archive handle.
out	<i>numberOfLocations</i>	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

<i>archive</i>	Archive handle.
<i>number</i>	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

<i>archive</i>	Archive handle.
<i>number</i>	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

	<i>archive</i>	Archive handle.
	<i>name</i>	Name of the property.
out	<i>value</i>	Returned value of the property. Allocated with <i>malloc</i> .

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_PROPERTY_NOT_FOUND if the named property was not
found

E.3 otf2/OTF2_Archive.h File Reference

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

	<i>archive</i>	Archive handle.
out	<i>numberOfProperties</i>	Returned number of trace file properties.
out	<i>names</i>	Returned list of property names. Allocated with <i>malloc</i> . To release memory, just pass <i>*names</i> to <i>free</i> .

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

<i>archive</i>	Archive handle.
<i>location</i>	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

<i>archive</i>	Archive handle.
<i>location</i>	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

<i>archive</i>	Archive handle.
<i>number</i>	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

<i>archive</i>	Archive handle.
<i>name</i>	Name of thumb.
<i>description</i>	Description of thumb.
<i>type</i>	Type of thumb.
<i>numberOfSamples</i>	Number of samples.
<i>numberOfMetrics</i>	Number of metrics.
<i>refsToDefs</i>	<i>numberOfMetrics</i> references to definition matching the thumbnail type.

E.3 otf2/OTF2_Archive.h File Reference

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_GetTraceld (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

	<i>archive</i>	Archive handle.
out	<i>id</i>	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

	<i>archive</i>	Archive handle
out	<i>major</i>	Major version number
out	<i>minor</i>	Minor version number
out	<i>bugfix</i>	Bugfix revision

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

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

<i>archivePath</i>	Path to the archive i.e. the directory where the anchor file is located.
<i>archive-Name</i>	Name of the archive. It is used to generate sub pathes e.g. 'archive-Name.otf2'.
<i>fileMode</i>	Determines if in reading or writing mode. Available values are OTF2_FILEMODE_WRITE or OTF2_FILEMODE_READ .
<i>chunk-SizeEvents</i>	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.
<i>chunk-SizeDefs</i>	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.
<i>fileSubstrate</i>	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 suppress file writing at all. In reading mode this value is ignored because the correct file substrated is extracted from the anchor file.
<i>compression</i>	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.

E.3 otf2/OTF2_Archive.h File Reference

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

<i>archive</i>	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

<i>archive</i>	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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>archive</i>	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

<i>archive</i>	Archive handle.
<i>location</i>	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

<i>archive</i>	Archive handle.
<i>name</i>	Name of the trace file property (case insensitive, [A-Z0-9_]).
<i>value</i>	Boolean value of property (e.g. true or false).
<i>overwrite</i>	If true a previous trace file property with the same name <i>name</i> will be overwritten.

E.3 otf2/OTF2_Archive.h File Reference

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

<i>archive</i>	Archive handle.
<i>collective-Callbacks</i>	Struct holding the collective callback functions.
<i>collective-Data</i>	Data passed to the collective callbacks in the <code>userData</code> argument.
<i>global-CommContext</i>	Global communication context.
<i>local-CommContext</i>	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

APPENDIX E. FILE DOCUMENTATION

only needs to be set for an archive handle marked as 'master' or does not need to be set at all.

Parameters

<i>archive</i>	Archive handle.
<i>creator</i>	Creator information.

Returns

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

E.3.4.54 `OTF2_StatusCode 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

<i>archive</i>	Archive handle.
<i>description</i>	Description.

Returns

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

E.3.4.55 `OTF2_StatusCode OTF2_Archive_SetFlushCallbacks (OTF2_Archive *
archive, const OTF2_FlushCallbacks * flushCallbacks, void * flushData)`

Set the flush callbacks for the archive.

Parameters

<i>archive</i>	Archive handle.
<i>flushCallbacks</i>	Struct holding the flush callback functions.
<i>flushData</i>	Data passed to the flush callbacks in the <code>userData</code> argument.

Returns

OTF2_StatusCode, or error code.

E.3 otf2/OTF2_Archive.h File Reference

E.3.4.56 **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

<i>archive</i>	Archive handle.
<i>machine-Name</i>	Machine name.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.3.4.57 **OTF2_ErrorCode** **OTF2_Archive_SetMemoryCallbacks** (**OTF2_Archive** * *archive*, const **OTF2_MemoryCallbacks** * *memoryCallbacks*, void * *memoryData*)

Set the memory callbacks for the archive.

Parameters

<i>archive</i>	Archive handle.
<i>memoryCallbacks</i>	Struct holding the memory callback functions.
<i>memoryData</i>	Data passed to the memory callbacks in the <code>userData</code> argument.

Returns

OTF2_ErrorCode, or error code.

E.3.4.58 **OTF2_ErrorCode** **OTF2_Archive_SetNumberOfSnapshots** (**OTF2_Archive** * *archive*, uint32_t *number*)

Set the number of snapshots.

Parameters

APPENDIX E. FILE DOCUMENTATION

<i>archive</i>	Archive handle.
<i>number</i>	Snapshot number.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.3.4.59 `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 then.

Note

This call is only allowed when the archive was opened with mode [*OTF2_FILEMODE_WRITE*](#).

Parameters

<i>archive</i>	Archive handle.
<i>name</i>	Name of the trace file property (case insensitive, [A-Z0-9_]).
<i>value</i>	Value of property.
<i>overwrite</i>	If true a previous trace file property with the same name <code>name</code> 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.4 otf2/OTF2_AttributeList.h File Reference

E.3.4.60 OTF2_ErrorCode OTF2_Archive_SetSerialCollectiveCallbacks (OTF2_Archive * *archive*)

Convenient function to set the collective callbacks to an serial implementation.

Parameters

<i>archive</i>	Archive handle.
----------------	-----------------

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.3.4.61 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

<i>archive</i>	Archive handle.
<i>newFile-Mode</i>	New <i>OTF2_FileMode</i> 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>
```

APPENDIX E. FILE DOCUMENTATION

```
#include <otf2/OTF2_GeneralDefinitions.h>
```

Data Structures

- union [OTF2_AttributeValue](#)
Value container for an attributes.

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](#) attribute-Value)
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_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.

E.4 otf2/OTF2_AttributeList.h File Reference

- [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_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_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.

APPENDIX E. FILE DOCUMENTATION

- [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_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)

E.4 otf2/OTF2_AttributeList.h File Reference

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_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_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.

APPENDIX E. FILE DOCUMENTATION

- [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.

E.4 oftf2/OTF2_AttributeList.h File Reference

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>type</i>	Type of the attribute.
<i>attribute-Value</i>	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.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>attributeRef</i>	Reference to Attribute definition.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.3 OTF2_ErrorCode OTF2_AttributeList_AddCommRef (OTF2_AttributeList
*** *attributeList*, OTF2_AttributeRef *attribute*, OTF2_CommRef *commRef*)**

Add an OTF2_TYPE_COMM attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>commRef</i>	Reference to Comm definition.

Returns

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

E.4.2.4 `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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>float64Value</i>	Value of the attribute.

Returns

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

E.4.2.5 `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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>float32Value</i>	Value of the attribute.

Returns

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

E.4.2.6 `OTF2_ErrorCode OTF2_AttributeList_AddGroupRef (OTF2_AttributeList
* attributeList, OTF2_AttributeRef attribute, OTF2_GroupRef groupRef)`

Add an OTF2_TYPE_GROUP attribute to an attribute list.

E.4 oftf2/OTF2_AttributeList.h File Reference

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>groupRef</i>	Reference to Group definition.

Returns

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

E.4.2.7 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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>int16Value</i>	Value of the attribute.

Returns

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

E.4.2.8 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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>int32Value</i>	Value of the attribute.

Returns

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

APPENDIX E. FILE DOCUMENTATION

E.4.2.9 **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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>int64Value</i>	Value of the attribute.

Returns

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

E.4.2.10 **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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>int8Value</i>	Value of the attribute.

Returns

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

E.4.2.11 **OTF2_ErrorCode** **OTF2_AttributeList_AddLocationRef** (**OTF2_AttributeList** * *attributeList*, **OTF2_AttributeRef** *attribute*, **OTF2_LocationRef** *locationRef*)

Add an OTF2_TYPE_LOCATION attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
----------------------	------------------------

E.4 otf2/OTF2_AttributeList.h File Reference

<i>attribute</i>	Reference to Attribute definition.
<i>locationRef</i>	Reference to Location definition.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.12 **OTF2_ErrorCode** **OTF2_AttributeList_AddMetricRef** (**OTF2_AttributeList**
* *attributeList*, **OTF2_AttributeRef** *attribute*, **OTF2_MetricRef** *metricRef*
)

Add an OTF2_TYPE_METRIC attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>metricRef</i>	Reference to Metric definition.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.13 **OTF2_ErrorCode** **OTF2_AttributeList_AddParameterRef** (
OTF2_AttributeList * *attributeList*, **OTF2_AttributeRef** *attribute*,
OTF2_ParameterRef *parameterRef*)

Add an OTF2_TYPE_PARAMETER attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>parameter-Ref</i>	Reference to Parameter definition.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.14 **OTF2_ErrorCode** **OTF2_AttributeList_AddRegionRef** (
 OTF2_AttributeList * *attributeList*, **OTF2_AttributeRef** *attribute*,
 OTF2_RegionRef *regionRef*)

Add an OTF2_TYPE_REGION attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>regionRef</i>	Reference to Region definition.

Returns

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

E.4.2.15 **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.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>rmaWinRef</i>	Reference to RmaWin definition.

Returns

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

E.4.2.16 **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.

E.4 of2/OTF2_AttributeList.h File Reference

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>stringRef</i>	Reference to String definition.

Returns

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

E.4.2.17 `OTF2_ErrorCode OTF2_AttributeList_AddStringRef (OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute, OTF2_StringRef stringRef)`

Add an OTF2_TYPE_STRING attribute to an attribute list.

Convenient function around *OTF2_AttributeList_AddAttribute*.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.
<i>stringRef</i>	Reference to String definition.

Returns

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

E.4.2.18 `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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>uint16Value</i>	Value of the attribute.

Returns

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

APPENDIX E. FILE DOCUMENTATION

E.4.2.19 **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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>uint32Value</i>	Value of the attribute.

Returns

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

E.4.2.20 **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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>uint64Value</i>	Value of the attribute.

Returns

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

E.4.2.21 **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

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to attribute definition.
<i>uint8Value</i>	Value of the attribute.

E.4 oftf2/OTF2_AttributeList.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.22 **OTF2_ErrorCode** OTF2_AttributeList_Delete (OTF2_AttributeList * *attributeList*)

Delete an attribute list handle.

Deletes an attribute list handle and releases all associated resources.

Parameters

<i>attributeList</i>	Attribute list handle.
----------------------	------------------------

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.23 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>type</i>	Returned type of the attribute.
out	<i>attribute-Value</i>	Returned value of the attribute.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.24 **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.

APPENDIX E. FILE DOCUMENTATION

Parameters

	<i>attributeList</i>	Attribute list handle.
	<i>index</i>	Position of the attribute in the attribute list.
out	<i>attribute</i>	Returned attribute reference.
out	<i>type</i>	Returned type of the attribute.
out	<i>attribute-Value</i>	Returned value of the attribute.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.25 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>attributeRef</i>	Returned attribute value.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.26 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>commRef</i>	Returned comm value.

E.4 otf2/OTF2_AttributeList.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.27 `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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>float64Value</i>	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_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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>float32Value</i>	Returned value of the attribute.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.4.2.29 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>groupRef</i>	Returned group value.

Returns

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

E.4.2.30 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>int16Value</i>	Returned value of the attribute.

Returns

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

E.4.2.31 **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*.

E.4 of2/OTF2_AttributeList.h File Reference

Parameters

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>int32Value</i>	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_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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>int64Value</i>	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_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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>int8Value</i>	Returned value of the attribute.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.4.2.34 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>locationRef</i>	Returned location value.

Returns

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

E.4.2.35 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>metricRef</i>	Returned metric value.

Returns

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

E.4.2.36 **uint32_t** **OTF2_AttributeList_GetNumberOfElements** (**const**
OTF2_AttributeList * *attributeList*)

Get the number of entries in an attribute list.

Parameters

<i>attributeList</i>	Attribute list handle.
----------------------	------------------------

E.4 of2/OTF2_AttributeList.h File Reference

Returns

Returns the number of elements in the list. Returns zero if the list does not exist.

E.4.2.37 `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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>parameter-Ref</i>	Returned parameter value.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.38 `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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>regionRef</i>	Returned region value.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.4.2.39 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>rmaWinRef</i>	Returned rmaWin value.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.40 **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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>stringRef</i>	Returned string value.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4 of2/OTF2_AttributeList.h File Reference

E.4.2.41 `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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to attribute definition.
out	<i>stringRef</i>	Returned string value.

Returns

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

E.4.2.42 `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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>uint16Value</i>	Returned value of the attribute.

Returns

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

E.4.2.43 `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*.

APPENDIX E. FILE DOCUMENTATION

Parameters

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>uint32Value</i>	Returned value of the attribute.

Returns

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

E.4.2.44 `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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>uint64Value</i>	Returned value of the attribute.

Returns

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

E.4.2.45 `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

	<i>attributeList</i>	Attribute list handle.
	<i>attribute</i>	Reference to Attribute definition.
out	<i>uint8Value</i>	Returned value of the attribute.

Returns

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

E.4 of2/OTF2_AttributeList.h File Reference

E.4.2.46 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.4.2.47 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

	<i>attributeList</i>	Attribute list handle.
out	<i>attribute</i>	Returned attribute reference.
out	<i>type</i>	Returned type of the attribute.
out	<i>attribute-Value</i>	Returned value of the attribute.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.48 OTF2_ErrorCode OTF2_AttributeList.RemoveAllAttributes (OTF2_AttributeList * *attributeList*)

Remove all attributes from an attribute list.

Parameters

<i>attributeList</i>	Attribute list handle.
----------------------	------------------------

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.49 `OTF2_ErrorCode OTF2_AttributeList_RemoveAttribute (`
`OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute)`

Remove an attribute from an attribute list.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.4.2.50 `bool OTF2_AttributeList_TestAttributeByID (const OTF2_AttributeList *`
`attributeList, OTF2_AttributeRef attribute)`

Test if an attribute is in the attribute list.

Parameters

<i>attributeList</i>	Attribute list handle.
<i>attribute</i>	Reference to Attribute definition.

Returns

True if the id is in the list, else false.

E.5 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.

E.5 otf2/OTF2_Callbacks.h File Reference

- 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)

APPENDIX E. FILE DOCUMENTATION

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 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.

- 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.

E.5.1 Detailed Description

This header file provides all user callbacks.

E.6 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.

E.6 otf2/OTF2_Definitions.h File Reference

- 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) }`

E.6 otf2/OTF2_Definitions.h File Reference

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 }
```

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.6.1 Detailed Description

Data types used in the definition records.

Source Template:

templates/OTF2_Definitions.tmpl.h

E.6.2 Enumeration Type Documentation

E.6.2.1 `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.6 otf2/OTF2_Definitions.h File Reference

E.6.2.2 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 [OTF2_GROUP_TYPE_COMM_LOCATIONS](#) group.

E.6.2.3 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 paradigm 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 exist per paradigm.

E.6.2.4 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.6.2.5 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.6 otf2/OTF2_Definitions.h File Reference

E.6.2.6 enum OTF2_MetricBase_enum

Metric base types.

Since

Version 1.0

Enumerator:

OTF2_BASE_BINARY Binary base.

OTF2_BASE_DECIMAL Decimal base.

E.6.2.7 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.6.2.8 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.6.2.9 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.6.2.10 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

E.6 oftf2/OTF2_Definitions.h File Reference

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.6.2.11 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.6.2.12 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.6.2.13 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.6.2.14 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.6 otf2/OTF2_Definitions.h File Reference

E.6.2.15 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.6.2.16 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.

APPENDIX E. FILE DOCUMENTATION

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.

E.7 otf2/OTF2_DefReader.h File Reference

Since

Version 1.3.

OTF2_REGION_ROLE_THREAD_WAIT A function which waits for the completion of one thread.

Since

Version 1.3.

E.6.2.17 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.7 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.

APPENDIX E. FILE DOCUMENTATION

```
#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.7.1 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.7.2 Function Documentation

E.7.2.1 [OTF2_ErrorCode OTF2_DefReader_GetLocationID](#) (const [OTF2_DefReader](#) * reader, [OTF2_LocationRef](#) * location)

Get the location ID of this reader object.

Parameters

<i>reader</i>	This given reader object will be deleted.
<i>location</i>	Pointer to the variable where the location ID is returned in.

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.7 otf2/OTF2_DefReader.h File Reference

E.7.2.2 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

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.
out	<i>recordsRead</i>	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given recordsToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finished 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.7.2.3 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

<i>reader</i>	This given reader object will be setted up with new callback functions.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <i>OTF2_DefReaderCallbacks_New</i> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.8 otF2/OTF2_DefReaderCallbacks.h File Reference

This defines the callbacks for the definition reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#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_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.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

- 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_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 [Location-Group](#) 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 [Location-GroupProperty](#) definition record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_DefReaderCallback_LocationProperty](#))(void *userData, [OTF2_LocationRef](#) location, [OTF2_StringRef](#) name, [OTF2_StringRef](#) value)

APPENDIX E. FILE DOCUMENTATION

Function pointer definition for the callback which is triggered by a [Location-Property](#) 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.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

- 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_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 [SystemTreeNode](#) 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 [SystemTreeNodeDomain](#) 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 [SystemTreeNodeProperty](#) 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 holds 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_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_SetLocationCallback \(OTF2_-DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_Location locationCallback\)](#)

Registers the callback for the [Location](#) definition.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

- [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_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.8.1 Detailed Description

This defines the callbacks for the definition reader.

Source Template:

templates/OTF2_DefReaderCallbacks.tmpl.h

E.8.2 Typedef Documentation

E.8.2.1 `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.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Attribute definition.
<i>name</i>	Name of the attribute. References a String definition.
<i>description</i>	Description of the attribute. References a String definition. Since version 1.4.
<i>type</i>	Type of the attribute value.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.2 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Callpath definition.
<i>parent</i>	The parent of this callpath. References a Callpath definition.
<i>region</i>	The region of this callpath. References a Region definition.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

APPENDIX E. FILE DOCUMENTATION

E.8.2.3 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Callsite definition.
<i>sourceFile</i>	The source file where this call was made. References a String definition.
<i>lineNumber</i>	Line number in the source file where this call was made.
<i>enteredRegion</i>	The region which was called. References a Region definition.
<i>leftRegion</i>	The region which made the call. References a Region definition.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.4 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>cartTopology</i>	Parent CartTopology definition to which this one is a supplementary definition. References a CartTopology definition.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

<i>rank</i>	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
<i>numberOfDimensions</i>	Number of dimensions.
<i>coordinates</i>	Coordinates, indexed by dimension.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.8.2.5 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterDefCallbacks</i> or <i>OTF2_DefReader_SetCallbacks</i> .
<i>self</i>	The unique identifier for this <i>CartDimension</i> definition.
<i>name</i>	The name of the cartesian topology dimension. References a <i>String</i> definition.
<i>size</i>	The size of the cartesian topology dimension.
<i>cartPeriodicity</i>	Periodicity of the cartesian topology dimension.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.8.2.6 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterDefCallbacks</i> or <i>OTF2_DefReader_SetCallbacks</i> .
<i>self</i>	The unique identifier for this <i>CartTopology</i> definition.
<i>name</i>	The name of the topology. References a <i>String</i> definition.
<i>communi- cator</i>	Communicator object used to create the topology. References a <i>Comm</i> definition.
<i>num- berOfDi- mensions</i>	Number of dimensions.
<i>cartDimen- sions</i>	The dimensions of this topology. References a <i>CartDimension</i> definition.

Since

Version 1.3

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.8.2.7 `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

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>time</i>	Time when this offset was determined.
<i>offset</i>	The offset to the global clock which was determined at <i>time</i> .
<i>standard-Deviation</i>	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.8.2.8 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Comm definition.
<i>name</i>	The name given by calling <code>MPI_Comm_set_name</code> on this communicator. Or the empty name to indicate that no name was given. References a String definition.
<i>group</i>	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.
<i>parent</i>	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.8.2.9 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Group definition.
<i>name</i>	Name of this group References a String definition.
<i>groupType</i>	The type of this group. Since version 1.2.
<i>paradigm</i>	The paradigm of this communication group. Since version 1.2.
<i>groupFlags</i>	Flags for this group. Since version 1.2.
<i>numberOfMembers</i>	The number of members in this group.
<i>members</i>	The identifiers of the group members.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.10 `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.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

The location definition.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Location definition.
<i>name</i>	Name of the location References a String definition.
<i>location-Type</i>	Location type.
<i>numberOfEvents</i>	Number of events this location has recorded.
<i>location-Group</i>	Location group which includes this location. References a Location-Group definition.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.11 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this LocationGroup definition.
<i>name</i>	Name of the group. References a String definition.
<i>location-GroupType</i>	Type of this group.
<i>systemTreeParent</i>	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.8.2.12 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterDefCallbacks</i> or <i>OTF2_-DefReader_SetCallbacks</i> .
<i>location-Group</i>	Parent <i>LocationGroup</i> definition to which this one is a supplementary definition. References a <i>LocationGroup</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.8.2.13 `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

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>location</i>	Parent Location definition to which this one is a supplementary definition. References a Location definition.
<i>name</i>	Name of the property. References a String definition.
<i>value</i>	Property value. References a String definition.

Since

Version 1.3

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.14 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>mapping-Type</i>	Says to what type of ID the mapping table has to be applied.
<i>idMap</i>	Mapping table.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.15 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this MetricClass definition.
<i>numberOfMetrics</i>	Number of metrics within the set.
<i>metricMembers</i>	List of metric members. References a MetricMember definition.
<i>metricOccurrence</i>	Defines occurrence of a metric set.
<i>recorderKind</i>	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.8.2.16 `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

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>metricClass</i>	Parent MetricClass definition to which this one is a supplementary definition. References a MetricClass definition.
<i>recorder</i>	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.8.2.17 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this MetricClass definition.
<i>metricClass</i>	The instanced MetricClass . This metric class must be of kind OTF2_RECORDER_KIND_ABSTRACT . References a MetricClass definition.
<i>recorder</i>	Recorder of the metric: location ID. References a Location definition.
<i>metric-Scope</i>	Defines type of scope: location, location group, system tree node, or a generic group of locations.
<i>scope</i>	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.8.2.18 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this MetricMember definition.
<i>name</i>	Name of the metric. References a String definition.
<i>description</i>	Description of the metric. References a String definition.
<i>metricType</i>	Metric type: PAPI, etc.
<i>metricMode</i>	Metric mode: accumulative, fix, relative, etc.
<i>valueType</i>	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.
<i>metricBase</i>	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.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$, 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.
<i>unit</i>	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.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.19 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Parameter definition.
<i>name</i>	Name of the parameter (variable name etc.) References a String definition.
<i>parameter-Type</i>	Type of the parameter, OTF2_ParameterType for possible types.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.20 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Region definition.
<i>name</i>	Name of the region (demangled name if available). References a String definition.
<i>canonical-Name</i>	Alternative name of the region (e.g. mangled name). References a String definition. Since version 1.1.
<i>description</i>	A more detailed description of this region. References a String definition.
<i>regionRole</i>	Region role. Since version 1.1.
<i>paradigm</i>	Paradigm. Since version 1.1.
<i>regionFlags</i>	Region flags. Since version 1.1.
<i>sourceFile</i>	The source file where this region was declared. References a String definition.
<i>beginLineNumber</i>	Starting line number of this region in the source file.
<i>endLineNumber</i>	Ending line number of this region in the source file.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.21 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this RmaWin definition.
<i>name</i>	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a String definition.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

<i>comm</i>	Communicator object used to create the window. References a Comm definition.
-------------	--

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.22 `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 definitions.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this String definition.
<i>string</i>	The string, null terminated.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8.2.23 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>self</i>	The unique identifier for this SystemTreeNode definition.
<i>name</i>	Free form instance name of this node. References a String definition.
<i>className</i>	Free form class name of this node References a String definition.
<i>parent</i>	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.8.2.24 `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 [SystemTreeNodeDomain](#) definition record.

The system tree node domain definition.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
<i>systemTreeNode</i>	Parent SystemTreeNode definition to which this one is a supplementary definition. References a SystemTreeNode definition.
<i>systemTreeDomain</i>	The domain in which the referenced SystemTreeNode operates in.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

E.8.2.25 `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 *SystemTreeNodeProperty* definition record.

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

Parameters

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterDefCallbacks</i> or <i>OTF2_DefReader_SetCallbacks</i> .
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.8.2.26 `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_
Unknown)(void *userData)`

Function pointer definition for the callback which is triggered for an unknown definition.

Parameters

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterDefCallbacks</i> or <i>OTF2_DefReader_SetCallbacks</i> .
-----------------	---

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.8.3 Function Documentation

E.8.3.1 `void OTF2_DefReaderCallbacks_Clear (OTF2_DefReaderCallbacks *
defReaderCallbacks)`

Clears a struct for the definition callbacks.

Parameters

<i>defReader- Callbacks</i>	Handle to a struct previously allocated with OTF2_- DefReaderCallbacks_New .
---------------------------------	--

E.8.3.2 `void OTF2_DefReaderCallbacks_Delete (OTF2_DefReaderCallbacks *
defReaderCallbacks)`

Deallocates a struct for the definition callbacks.

Parameters

<i>defReader- Callbacks</i>	Handle to a struct previously allocated with OTF2_- DefReaderCallbacks_New .
---------------------------------	--

E.8.3.3 `OTF2_DefReaderCallbacks* OTF2_DefReaderCallbacks_New (void)`

Allocates a new struct for the definition callbacks.

Returns

A newly allocated struct of type [OTF2_DefReaderCallbacks](#).

E.8.3.4 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetAttributeCallback
(OTF2_DefReaderCallbacks * defReaderCallbacks,
OTF2_DefReaderCallback_Attribute attributeCallback)`

Registers the callback for the [Attribute](#) definition.

Parameters

<i>defReader- Callbacks</i>	Struct for all callbacks.
<i>attribute- Callback</i>	Function which should be called for all Attribute definitions.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful

[*OTF2_ERROR_INVALID_ARGUMENT*](#) for an invalid `defReaderCallbacks` argument

E.8.3.5 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallpathCallback`
(`OTF2_DefReaderCallbacks * defReaderCallbacks`,
`OTF2_DefReaderCallback_Callpath callpathCallback`)

Registers the callback for the [*Callpath*](#) definition.

Parameters

<i>defReader-Callbacks</i>	Struct for all callbacks.
<i>callpath-Callback</i>	Function which should be called for all <i>Callpath</i> definitions.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful

[*OTF2_ERROR_INVALID_ARGUMENT*](#) for an invalid `defReaderCallbacks` argument

E.8.3.6 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallsiteCallback`
(`OTF2_DefReaderCallbacks * defReaderCallbacks`,
`OTF2_DefReaderCallback_Callsite callsiteCallback`)

Registers the callback for the [*Callsite*](#) definition.

Parameters

<i>defReader-Callbacks</i>	Struct for all callbacks.
<i>callsite-Callback</i>	Function which should be called for all <i>Callsite</i> definitions.

APPENDIX E. FILE DOCUMENTATION

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.8.3.7 **OTF2_StatusCode** **OTF2_DefReaderCallbacks_SetCartCoordinateCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_CartCoordinate *cartCoordinateCallback*)

Registers the callback for the *CartCoordinate* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>cartCoordinateCallback</i>	Function which should be called for all <i>CartCoordinate</i> definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.8.3.8 **OTF2_StatusCode** **OTF2_DefReaderCallbacks_SetCartDimensionCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_CartDimension *cartDimensionCallback*)

Registers the callback for the *CartDimension* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
---------------------------	---------------------------

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

<i>cartDimensionCallback</i>	Function which should be called for all <i>CartDimension</i> definitions.
------------------------------	---

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.8.3.9 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCartTopologyCallback (OTF2_DefReaderCallbacks * defReaderCallbacks, OTF2_DefReaderCallback_CartTopology cartTopologyCallback)`

Registers the callback for the *CartTopology* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>cartTopologyCallback</i>	Function which should be called for all <i>CartTopology</i> definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.8.3.10 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetClockOffsetCallback (OTF2_DefReaderCallbacks * defReaderCallbacks, OTF2_DefReaderCallback_ClockOffset clockOffsetCallback)`

Registers the callback for the *ClockOffset* definition.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>clockOffsetCallback</i>	Function which should be called for all <i>ClockOffset</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.11 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetCommCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_Comm *commCallback*)

Registers the callback for the *Comm* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>commCallback</i>	Function which should be called for all <i>Comm</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.12 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetGroupCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_Group *groupCallback*)

Registers the callback for the *Group* definition.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>groupCallback</i>	Function which should be called for all <i>Group</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.13 *OTF2_StatusCode* *OTF2_DefReaderCallbacks_SetLocationCallback*
(*OTF2_DefReaderCallbacks* * *defReaderCallbacks*,
OTF2_DefReaderCallback_Location *locationCallback*)

Registers the callback for the *Location* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>locationCallback</i>	Function which should be called for all <i>Location</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.14 *OTF2_StatusCode* *OTF2_DefReaderCallbacks_SetLocationGroupCallback*
(*OTF2_DefReaderCallbacks* * *defReaderCallbacks*,
OTF2_DefReaderCallback_LocationGroup *locationGroupCallback*)

Registers the callback for the *LocationGroup* definition.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>locationGroupCallback</i>	Function which should be called for all <i>LocationGroup</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.15 **OTF2_StatusCode** **OTF2_DefReaderCallbacks_-**
SetLocationGroupPropertyCallback (**OTF2_DefReaderCallbacks**
***** *defReaderCallbacks***,** **OTF2_DefReaderCallback_-**
LocationGroupProperty *locationGroupPropertyCallback*
)

Registers the callback for the *LocationGroupProperty* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>locationGroupPropertyCallback</i>	Function which should be called for all <i>LocationGroupProperty</i> definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

E.8.3.16 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetLocationPropertyCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
 OTF2_DefReaderCallback_LocationProperty *locationPropertyCallback*
)

Registers the callback for the *LocationProperty* definition.

Parameters

<i>defReader-Callbacks</i>	Struct for all callbacks.
<i>location-Property-Callback</i>	Function which should be called for all <i>LocationProperty</i> definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.17 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetMappingTableCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
 OTF2_DefReaderCallback_MappingTable *mappingTableCallback*)

Registers the callback for the *MappingTable* definition.

Parameters

<i>defReader-Callbacks</i>	Struct for all callbacks.
<i>map-pingTable-Callback</i>	Function which should be called for all <i>MappingTable</i> definitions.

Since

Version 1.0

APPENDIX E. FILE DOCUMENTATION

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.8.3.18 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetMetricClassCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_MetricClass *metricClassCallback*)

Registers the callback for the *MetricClass* definition.

Parameters

<i>defReader-Callbacks</i>	Struct for all callbacks.
<i>metric-ClassCallback</i>	Function which should be called for all <i>MetricClass</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.8.3.19 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetMetricClassRecorderCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_MetricClassRecorder
metricClassRecorderCallback)

Registers the callback for the *MetricClassRecorder* definition.

Parameters

<i>defReader-Callbacks</i>	Struct for all callbacks.
<i>metric-Class-Recorder-Callback</i>	Function which should be called for all <i>MetricClassRecorder</i> definitions.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful

[*OTF2_ERROR_INVALID_ARGUMENT*](#) for an invalid `defReaderCallbacks` argument

E.8.3.20 `OTF2_StatusCode OTF2_DefReaderCallbacks_SetMetricInstanceCallback`
(`OTF2_DefReaderCallbacks * defReaderCallbacks`,
`OTF2_DefReaderCallback_MetricInstance metricInstanceCallback`)

Registers the callback for the [*MetricInstance*](#) definition.

Parameters

<i>defReader-Callbacks</i>	Struct for all callbacks.
<i>metricInstanceCallback</i>	Function which should be called for all <i>MetricInstance</i> definitions.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful

[*OTF2_ERROR_INVALID_ARGUMENT*](#) for an invalid `defReaderCallbacks` argument

E.8.3.21 `OTF2_StatusCode OTF2_DefReaderCallbacks_SetMetricMemberCallback`
(`OTF2_DefReaderCallbacks * defReaderCallbacks`,
`OTF2_DefReaderCallback_MetricMember metricMemberCallback`)

Registers the callback for the [*MetricMember*](#) definition.

Parameters

<i>defReader-Callbacks</i>	Struct for all callbacks.
----------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>metricMemberCallback</i>	Function which should be called for all <i>MetricMember</i> definitions.
-----------------------------	--

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.22 *OTF2_StatusCode* *OTF2_DefReaderCallbacks_SetParameterCallback*
(*OTF2_DefReaderCallbacks* * *defReaderCallbacks*,
OTF2_DefReaderCallback_Parameter *parameterCallback*)

Registers the callback for the *Parameter* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>parameterCallback</i>	Function which should be called for all <i>Parameter</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.23 *OTF2_StatusCode* *OTF2_DefReaderCallbacks_SetRegionCallback*
(*OTF2_DefReaderCallbacks* * *defReaderCallbacks*,
OTF2_DefReaderCallback_Region *regionCallback*)

Registers the callback for the *Region* definition.

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>regionCallback</i>	Function which should be called for all <i>Region</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.24 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetRmaWinCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_RmaWin *rmaWinCallback*)

Registers the callback for the *RmaWin* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWinCallback</i>	Function which should be called for all <i>RmaWin</i> definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.25 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetStringCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_String *stringCallback*)

Registers the callback for the *String* definition.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>stringCallback</i>	Function which should be called for all <i>String</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8.3.26 *OTF2_*ErrorCode *OTF2_DefReaderCallbacks_SetSystemTreeNodeCallback*
(*OTF2_DefReaderCallbacks* * *defReaderCallbacks*,
OTF2_DefReaderCallback_SystemTreeNode *systemTreeNodeCallback*)

Registers the callback for the *SystemTreeNode* definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>systemTreeNodeCallback</i>	Function which should be called for all <i>SystemTreeNode</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.8 otf2/OTF2_DefReaderCallbacks.h File Reference

E.8.3.27 **OTF2_ErrorCode** **OTF2.DefReaderCallbacks_-**
SetSystemTreeNodeDomainCallback (**OTF2_DefReaderCallbacks**
* **defReaderCallbacks**, **OTF2_DefReaderCallback_-**
SystemTreeNodeDomain **systemTreeNodeDomainCallback**
)

Registers the callback for the [SystemTreeNodeDomain](#) definition.

Parameters

<i>defReader- Callbacks</i>	Struct for all callbacks.
<i>sys- temTreeN- odeDo- mainCall- back</i>	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.8.3.28 **OTF2_ErrorCode** **OTF2.DefReaderCallbacks_-**
SetSystemTreeNodePropertyCallback (**OTF2_DefReaderCallbacks**
* **defReaderCallbacks**, **OTF2_DefReaderCallback_-**
SystemTreeNodeProperty **systemTreeNodePropertyCallback**
)

Registers the callback for the [SystemTreeNodeProperty](#) definition.

Parameters

<i>defReader- Callbacks</i>	Struct for all callbacks.
<i>sys- temTreeN- odeProp- ertyCallback</i>	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.8.3.29 **OTF2_StatusCode** **OTF2_DefReaderCallbacks_SetUnknownCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_Unknown *unknownCallback*)

Registers the callback for an unknown definition.

Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>unknownCallback</i>	Function which should be called for all unknown definitions.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.9 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.

E.9 otf2/OTF2_DefWriter.h File Reference

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_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_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)

E.9 otf2/OTF2_DefWriter.h File Reference

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_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.9.1 Detailed Description

This file provides all routines that write definition records of a single location.

Source Template:

templates/OTF2_DefWriter.tmpl.h

E.9.2 Function Documentation

E.9.2.1 [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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>writer</i>	Writer object.
<i>location</i>	Return location reference.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9.2.2 **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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <i>Attribute</i> definition.
<i>name</i>	Name of the attribute. References a <i>String</i> definition.
<i>description</i>	Description of the attribute. References a <i>String</i> definition. Since version 1.4.
<i>type</i>	Type of the attribute value.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9.2.3 **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

<i>writer</i>	Writer object.
---------------	----------------

E.9 otf2/OTF2_DefWriter.h File Reference

<i>self</i>	The unique identifier for this Callpath definition.
<i>parent</i>	The parent of this callpath. References a Callpath definition.
<i>region</i>	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.9.2.4 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this Callsite definition.
<i>sourceFile</i>	The source file where this call was made. References a String definition.
<i>lineNumber</i>	Line number in the source file where this call was made.
<i>enteredRegion</i>	The region which was called. References a Region definition.
<i>leftRegion</i>	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.9.2.5 `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.

APPENDIX E. FILE DOCUMENTATION

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

<i>writer</i>	Writer object.
<i>cartTopology</i>	Parent CartTopology definition to which this one is a supplementary definition. References a CartTopology definition.
<i>rank</i>	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
<i>numberOfDimensions</i>	Number of dimensions.
<i>coordinates</i>	Coordinates, indexed by dimension.

Since

Version 1.3

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.9.2.6 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this CartDimension definition.
<i>name</i>	The name of the cartesian topology dimension. References a String definition.
<i>size</i>	The size of the cartesian topology dimension.
<i>cartPeriodicity</i>	Periodicity of the cartesian topology dimension.

Since

Version 1.3

E.9 otf2/OTF2_DefWriter.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9.2.7 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <i>CartTopology</i> definition.
<i>name</i>	The name of the topology. References a <i>String</i> definition.
<i>communicator</i>	Communicator object used to create the topology. References a <i>Comm</i> definition.
<i>numberOfDimensions</i>	Number of dimensions.
<i>cartDimensions</i>	The dimensions of this topology. References a <i>CartDimension</i> definition.

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9.2.8 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>writer</i>	Writer object.
<i>time</i>	Time when this offset was determined.
<i>offset</i>	The offset to the global clock which was determined at <i>time</i> .
<i>standard-Deviation</i>	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.9.2.9 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <i>Comm</i> definition.
<i>name</i>	The name given by calling <code>MPI_Comm_set_name</code> on this communicator. Or the empty name to indicate that no name was given. References a <i>String</i> definition.
<i>group</i>	The describing MPI group of this MPI communicator The group needs to be of type <i>OTF2_GROUP_TYPE_COMM_GROUP</i> or <i>OTF2_GROUP_TYPE_COMM_SELF</i> . References a <i>Group</i> definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <i>OTF2_UNDEFINED_COMM</i> to indicate no parent. References a <i>Comm</i> definition.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9 otf2/OTF2_DefWriter.h File Reference

E.9.2.10 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this Group definition.
<i>name</i>	Name of this group References a String definition.
<i>groupType</i>	The type of this group. Since version 1.2.
<i>paradigm</i>	The paradigm of this communication group. Since version 1.2.
<i>groupFlags</i>	Flags for this group. Since version 1.2.
<i>numberOfMembers</i>	The number of members in this group.
<i>members</i>	The identifiers of the group members.

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.9.2.11 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this Location definition.
<i>name</i>	Name of the location References a String definition.
<i>locationType</i>	Location type.

APPENDIX E. FILE DOCUMENTATION

<i>numberOfEvents</i>	Number of events this location has recorded.
<i>location-Group</i>	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.9.2.12 **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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this LocationGroup definition.
<i>name</i>	Name of the group. References a String definition.
<i>location-GroupType</i>	Type of this group.
<i>systemTreeParent</i>	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.9 otf2/OTF2_DefWriter.h File Reference

E.9.2.13 **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

<i>writer</i>	Writer object.
<i>location-Group</i>	Parent <i>LocationGroup</i> definition to which this one is a supplementary definition. References a <i>LocationGroup</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.3

Returns

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

E.9.2.14 **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

<i>writer</i>	Writer object.
<i>location</i>	Parent <i>Location</i> definition to which this one is a supplementary definition. References a <i>Location</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.3

Returns

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

APPENDIX E. FILE DOCUMENTATION

E.9.2.15 `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

<i>writer</i>	Writer object.
<i>mapping-Type</i>	Says to what type of ID the mapping table has to be applied.
<i>idMap</i>	Mapping table.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9.2.16 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <i>MetricClass</i> definition.
<i>numberOfMetrics</i>	Number of metrics within the set.
<i>metricMembers</i>	List of metric members. References a <i>MetricMember</i> definition.
<i>metricOccurrence</i>	Defines occurrence of a metric set.
<i>recorderKind</i>	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

E.9 otf2/OTF2_DefWriter.h File Reference

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9.2.17 `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

<i>writer</i>	Writer object.
<i>metricClass</i>	Parent <i>MetricClass</i> definition to which this one is a supplementary definition. References a <i>MetricClass</i> definition.
<i>recorder</i>	The location which recorded the referenced metric class. References a <i>Location</i> definition.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9.2.18 `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

APPENDIX E. FILE DOCUMENTATION

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <i>MetricClass</i> definition.
<i>metricClass</i>	The instanced <i>MetricClass</i> . This metric class must be of kind <i>OTF2_RECORDER_KIND_ABSTRACT</i> . References a <i>MetricClass</i> definition.
<i>recorder</i>	Recorder of the metric: location ID. References a <i>Location</i> definition.
<i>metric-Scope</i>	Defines type of scope: location, location group, system tree node, or a generic group of locations.
<i>scope</i>	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.9.2.19 *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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <i>MetricMember</i> definition.
<i>name</i>	Name of the metric. References a <i>String</i> definition.
<i>description</i>	Description of the metric. References a <i>String</i> definition.
<i>metricType</i>	Metric type: PAPI, etc.
<i>metricMode</i>	Metric mode: accumulative, fix, relative, etc.
<i>valueType</i>	Type of the value. Only <i>OTF2_TYPE_INT64</i> , <i>OTF2_TYPE_UINT64</i> , and <i>OTF2_TYPE_DOUBLE</i> are valid types. If this metric member is recorded in an <i>Metric</i> event, than this type and the type in the event must match.

E.9 otf2/OTF2_DefWriter.h File Reference

<i>metricBase</i>	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.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$, 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.
<i>unit</i>	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.9.2.20 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this Parameter definition.
<i>name</i>	Name of the parameter (variable name etc.) References a String definition.
<i>parameter-Type</i>	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.

APPENDIX E. FILE DOCUMENTATION

E.9.2.21 `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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this Region definition.
<i>name</i>	Name of the region (demangled name if available). References a String definition.
<i>canonical-Name</i>	Alternative name of the region (e.g. mangled name). References a String definition. Since version 1.1.
<i>description</i>	A more detailed description of this region. References a String definition.
<i>regionRole</i>	Region role. Since version 1.1.
<i>paradigm</i>	Paradigm. Since version 1.1.
<i>regionFlags</i>	Region flags. Since version 1.1.
<i>sourceFile</i>	The source file where this region was declared. References a String definition.
<i>beginLineNumber</i>	Starting line number of this region in the source file.
<i>endLineNumber</i>	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.9.2.22 `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.

E.9 oftf2/OTF2_DefWriter.h File Reference

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

Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this RmaWin definition.
<i>name</i>	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a String definition.
<i>comm</i>	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.9.2.23 `OTF2_ErrorCode OTF2_DefWriter_WriteString (OTF2_DefWriter * writer,
OTF2_StringRef self, const char * string)`

Writes a String definition record into the DefWriter.

The string definitions.

Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this String definition.
<i>string</i>	The string, null terminated.

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.9.2.24 **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

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <i>SystemTreeNode</i> definition.
<i>name</i>	Free form instance name of this node. References a <i>String</i> definition.
<i>className</i>	Free form class name of this node References a <i>String</i> definition.
<i>parent</i>	Parent id of this node. May be <i>OTF2_UNDEFINED_SYSTEM_TREE_NODE</i> to indicate that there is no parent. References a <i>SystemTreeNode</i> definition.

Since

Version 1.0

Returns

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

E.9.2.25 **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

<i>writer</i>	Writer object.
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>systemTreeDomain</i>	The domain in which the referenced <i>SystemTreeNode</i> operates in.

E.10 otf2/OTF2_Events.h File Reference

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.9.2.26 `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

<i>writer</i>	Writer object.
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.10 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,
}

E.10 otf2/OTF2_Events.h File Reference

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.10.1 Detailed Description

Enums and types used in event records.

Source Template:

templates/OTF2_Events.tmpl.h

E.10.2 Enumeration Type Documentation**E.10.2.1 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.

E.10 otf2/OTF2_Events.h File Reference

- 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.10.2.2 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.10.2.3 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.10.2.4 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.

E.11 otf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.4.

E.10.2.5 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.10.2.6 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.11 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_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)

E.11 otf2/OTF2_EventSizeEstimator.h File Reference

Calculates the size estimate for the `MpiRecvRequest` 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.

APPENDIX E. FILE DOCUMENTATION

- `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.

E.11 oftf2/OTF2_EventSizeEstimator.h File Reference

- `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.

APPENDIX E. FILE DOCUMENTATION

- `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_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)`

E.11 otf2/OTF2_EventSizeEstimator.h File Reference

Sets the number of Group 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_SetNumberOfStringDefinitions](#)
([OTF2_EventSizeEstimator](#) *estimator, uint32_t numberOfStringDefinitions)

Sets the number of String definitions used.

E.11.1 Detailed Description

Provides a interface to estimate the size of an resulting trace file.

Source Template:

templates/OTF2_EventSizeEstimator.tmpl.h

E.11.2 Function Documentation

E.11.2.1 OTF2_ErrorCode OTF2_EventSizeEstimator_Delete (OTF2_EventSizeEstimator * estimator)

Deletes an OTF2_EventSizeEstimator object.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.11.2.2 `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

<i>estimator</i>	Estimator object.
<i>attributeList</i>	Attribute List.

Returns

The estimated size.

E.11.2.3 `size_t OTF2_EventSizeEstimator_GetSizeOfBufferFlushEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the BufferFlush event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11 of2/OTF2_EventSizeEstimator.h File Reference

E.11.2.4 `size_t OTF2_EventSizeEstimator_GetSizeOfEnterEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the Enter event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.5 `size_t OTF2_EventSizeEstimator_GetSizeOfLeaveEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the Leave event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.6 `size_t OTF2_EventSizeEstimator_GetSizeOfMeasurementOnOffEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MeasurementOnOff event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.7 `size_t OTF2_EventSizeEstimator_GetSizeOfMetricEvent (`
`OTF2_EventSizeEstimator * estimator, uint8_t numberOfMetrics)`

Calculates the size estimate for the Metric event.

Parameters

<i>estimator</i>	Estimator object.
<i>numberOfMetrics</i>	Number of metrics with in the set.

Since

Version 1.0

Returns

The estimated size.

E.11.2.8 `size_t OTF2_EventSizeEstimator_GetSizeOfMpiCollectiveBeginEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpiCollectiveBegin event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11 of2/OTF2_EventSizeEstimator.h File Reference

E.11.2.9 `size_t OTF2_EventSizeEstimator_GetSizeOfMpiCollectiveEndEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpiCollectiveEnd event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.10 `size_t OTF2_EventSizeEstimator_GetSizeOfMpiIrecvEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpiIrecv event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.11 `size_t OTF2_EventSizeEstimator_GetSizeOfMpiIrecvRequestEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpiIrecvRequest event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.12 `size_t OTF2_EventSizeEstimator_GetSizeOfMpilsendCompleteEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpilsendComplete event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.13 `size_t OTF2_EventSizeEstimator_GetSizeOfMpilsendEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the Mpilsend event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11 of2/OTF2_EventSizeEstimator.h File Reference

E.11.2.14 `size_t OTF2_EventSizeEstimator_GetSizeOfMpiRecvEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpiRecv event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.15 `size_t OTF2_EventSizeEstimator_GetSizeOfMpiRequestCancelledEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpiRequestCancelled event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.16 `size_t OTF2_EventSizeEstimator_GetSizeOfMpiRequestTestEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpiRequestTest event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.17 `size_t OTF2_EventSizeEstimator_GetSizeOfMpiSendEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the MpiSend event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.18 `size_t OTF2_EventSizeEstimator_GetSizeOfOmpAcquireLockEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the OmpAcquireLock event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.11 of2/OTF2_EventSizeEstimator.h File Reference

E.11.2.19 `size_t OTF2_EventSizeEstimator_GetSizeOfOmpForkEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the OmpFork event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.11.2.20 `size_t OTF2_EventSizeEstimator_GetSizeOfOmpJoinEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the OmpJoin event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.11.2.21 `size_t OTF2_EventSizeEstimator_GetSizeOfOmpReleaseLockEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the OmpReleaseLock event.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.11.2.22 `size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskCompleteEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the OmpTaskComplete event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.11.2.23 `size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskCreateEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the OmpTaskCreate event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

E.11 of2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.11.2.24 `size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskSwitchEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the OmpTaskSwitch event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Deprecated

In version 1.2

Returns

The estimated size.

E.11.2.25 `size_t OTF2_EventSizeEstimator_GetSizeOfParameterIntEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ParameterInt event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.26 `size_t OTF2_EventSizeEstimator_GetSizeOfParameterStringEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ParameterString event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.27 `size_t OTF2_EventSizeEstimator_GetSizeOfParameterUnsignedIntEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ParameterUnsignedInt event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.0

Returns

The estimated size.

E.11.2.28 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaAcquireLockEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaAcquireLock event.

E.11 of2/OTF2_EventSizeEstimator.h File Reference

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.29 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaAtomicEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaAtomic event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.30 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveBeginEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaCollectiveBegin event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

APPENDIX E. FILE DOCUMENTATION

E.11.2.31 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveEndEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaCollectiveEnd event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.32 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaGetEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaGet event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.33 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaGroupSyncEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaGroupSync event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

E.11 oftf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.2

Returns

The estimated size.

E.11.2.34 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteBlockingEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaOpCompleteBlocking event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.35 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteNonBlockingEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaOpCompleteNonBlocking event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

APPENDIX E. FILE DOCUMENTATION

E.11.2.36 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteRemoteEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaOpCompleteRemote event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.37 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpTestEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaOpTest event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.38 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaPutEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaPut event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

E.11 oftf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.2

Returns

The estimated size.

E.11.2.39 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaReleaseLockEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaReleaseLock event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.40 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaRequestLockEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaRequestLock event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

APPENDIX E. FILE DOCUMENTATION

E.11.2.41 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaSyncEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaSync event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.42 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaTryLockEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaTryLock event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.43 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWaitChangeEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaWaitChangeEvent event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

E.11 oftf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.2

Returns

The estimated size.

E.11.2.44 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinCreateEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaWinCreate event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.45 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinDestroyEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the RmaWinDestroy event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

APPENDIX E. FILE DOCUMENTATION

E.11.2.46 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadAcquireLockEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadAcquireLock event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.47 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadBeginEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadBegin event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.3

Returns

The estimated size.

E.11.2.48 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadCreateEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadCreate event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

E.11 otf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.3

Returns

The estimated size.

E.11.2.49 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadEnd event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.3

Returns

The estimated size.

E.11.2.50 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadForkEvent (`
 `OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadFork event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

APPENDIX E. FILE DOCUMENTATION

E.11.2.51 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadJoinEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadJoin event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.52 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadReleaseLockEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadReleaseLock event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.53 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCompleteEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadTaskComplete event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

E.11 oftf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.2

Returns

The estimated size.

E.11.2.54 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCreateEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadTaskCreate event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.55 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskSwitchEvent (OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadTaskSwitch event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

APPENDIX E. FILE DOCUMENTATION

E.11.2.56 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamBeginEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadTeamBegin event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.57 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamEndEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadTeamEnd event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

Since

Version 1.2

Returns

The estimated size.

E.11.2.58 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadWaitEvent (`
`OTF2_EventSizeEstimator * estimator)`

Calculates the size estimate for the ThreadWait event.

Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

E.11 otf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.3

Returns

The estimated size.

E.11.2.59 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

<i>estimator</i>	Estimator object.
------------------	-------------------

Returns

The estimated size.

E.11.2.60 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.11.2.61 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

<i>estimator</i>	Estimator object.
<i>numberOfAttributeDefinitions</i>	The number of definitions.

Since

Version 1.0

Returns

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

E.11.2.62 **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

<i>estimator</i>	Estimator object.
<i>numberOfCommDefinitions</i>	The number of definitions.

Since

Version 1.0

Returns

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

E.11.2.63 **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

<i>estimator</i>	Estimator object.
<i>numberOfGroupDefinitions</i>	The number of definitions.

E.11 otf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

**E.11.2.64 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

<i>estimator</i>	Estimator object.
<i>num- berOfLoca- tionDefini- tions</i>	The number of definitions.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.11.2.65 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

<i>estimator</i>	Estimator object.
<i>numberOf- MetricDefi- nitions</i>	The number of definitions.

APPENDIX E. FILE DOCUMENTATION

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.11.2.66 **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

<i>estimator</i>	Estimator object.
<i>numberOfParameterDefinitions</i>	The number of definitions.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.11.2.67 **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

<i>estimator</i>	Estimator object.
<i>numberOfRegionDefinitions</i>	The number of definitions.

E.11 oftf2/OTF2_EventSizeEstimator.h File Reference

Since

Version 1.0

Returns

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

E.11.2.68 **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

<i>estimator</i>	Estimator object.
<i>num- berOfR- maWinDefi- nitions</i>	The number of definitions.

Since

Version 1.2

Returns

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

E.11.2.69 **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

<i>estimator</i>	Estimator object.
<i>numberOf- StringDefi- nitions</i>	The number of definitions.

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.12 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 tables 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.

E.12 otf2/OTF2_EvtReader.h File Reference

- [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.12.1 Detailed Description

This is the local event reader, which reads events from one location.

E.12.2 Function Documentation

E.12.2.1 [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

<i>reader</i>	Reader object.
<i>action</i>	Truth value whether the clock offsets should be applied or not.

Returns

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

E.12.2.2 **OTF2_ErrorCode** **OTF2_EvtReader.ApplyMappingTables** (
 OTF2_EvtReader * *reader*, **bool** *action*)

Enable or disable applying of the mapping tables to events read from this event reader.

This setting has no effect if the events are read by an global event reader.

Parameters

<i>reader</i>	Reader object.
<i>action</i>	Truth value whether the mappings should be applied or not.

Returns

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

E.12.2.3 **OTF2_ErrorCode** **OTF2_EvtReader.GetLocationID** (**const**
 OTF2_EvtReader * *reader*, **OTF2_LocationRef** * *location*)

Return the location ID of the reading related location.

Parameters

	<i>reader</i>	Reader object which reads the events from its buffer.
<i>out</i>	<i>location</i>	ID of the location.

Returns

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

E.12.2.4 **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

E.12 otf2/OTF2_EvtReader.h File Reference

	<i>reader</i>	Reader object which reads the events from its buffer.
out	<i>position</i>	Number of the event in the stream.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.12.2.5 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

	<i>reader</i>	Reader object which reads the events from its buffer.
	<i>recordsToRead</i>	How many records can be read next.
out	<i>recordsRead</i>	Return how many records where really read.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.12.2.6 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

	<i>reader</i>	Reader object which reads the events from its buffer.
	<i>recordsToRead</i>	How many records can be read next.
out	<i>recordsRead</i>	Return how many records where really read.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

**E.12.2.7 OTF2_ErrorCode OTF2_EvtReader.Seek (OTF2_EvtReader * *reader*,
uint64_t *position*)**

Seek jumps to an event position.

Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
<i>position</i>	Number of the event, where the reader has to jump.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

**E.12.2.8 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

<i>reader</i>	Reader object which reads the events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <i>OTF2_EvtReaderCallbacks_New</i> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

**E.12.2.9 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

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

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

<i>reader</i>	Reader object which reads the events from its buffer.
<i>time</i>	New timestamp

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.13 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_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.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

- 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

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

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.

E.13 otF2/OTF2_EvtReaderCallbacks.h File Reference

- 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.

APPENDIX E. FILE DOCUMENTATION

- 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 holds 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_SetEnterCallback](#) ([OTF2_EvtReaderCallbacks](#) *evtReaderCallbacks, [OTF2_EvtReaderCallback_Enter](#) enterCallback)
Registers the callback for the Enter event.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

- [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.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- [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.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- [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.13.1 Detailed Description

This defines the callbacks for the event reader.

Source Template:

templates/OTF2_EvtReaderCallbacks.tmpl.h

E.13.2 Typedef Documentation

E.13.2.1 `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

<i>location</i>	The location where this event happened.
-----------------	---

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>stopTime</i>	The time the buffer flush finished.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.2 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	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](#).

APPENDIX E. FILE DOCUMENTATION

E.13.2.3 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	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.13.2.4 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>measurementMode</i>	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.13.2.5 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>metric</i>	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.
<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding MetricMember definitions.
<i>metricValues</i>	List of metric values.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.6 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.7 `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:

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a Comm definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_COMM is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.8 `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

APPENDIX E. FILE DOCUMENTATION

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communicator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.9 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the requested receive

E.13 oftf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.10 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.11 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_- EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.12 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.

E.13 oftf2/OTF2_EvtReaderCallbacks.h File Reference

<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communicator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.13 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.14 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.15 `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).

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communicator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.16 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

APPENDIX E. FILE DOCUMENTATION

<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.13.2.17 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>numberOfRequestedThreads</i>	Requested size of the team.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.2.18 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.19 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.

APPENDIX E. FILE DOCUMENTATION

<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.13.2.20 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the completed task instance.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.21 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the newly created task instance.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.22 `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.

APPENDIX E. FILE DOCUMENTATION

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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the now active task instance.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.23 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.24 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	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.
<i>string</i>	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.13.2.25 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	Parameter ID. References a <i>Parameter</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_PARAMETER</i> is available.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.26 `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.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

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

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.27 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

APPENDIX E. FILE DOCUMENTATION

<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>type</i>	Type of atomic operation.
<i>bytesSent</i>	Bytes sent to target.
<i>bytesReceived</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.28 `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 beginning of a collective RMA operation.

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.2.29 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	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.
<i>root</i>	Root process for this operation.
<i>bytesSent</i>	Bytes sent in operation.
<i>bytesReceived</i>	Bytes receives in operation.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.30 `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.

APPENDIX E. FILE DOCUMENTATION

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

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.31 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .

E.13 oftf2/OTF2_EvtReaderCallbacks.h File Reference

<i>attributeList</i>	Additional attributes for this event.
<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	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.
<i>group</i>	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.13.2.32 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.33 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.2.34 `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 local completion of an RMA operation.

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.35 `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

APPENDIX E. FILE DOCUMENTATION

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.36 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes sent to target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.37 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	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.13.2.38 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.39 `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.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the locked remote process.
<i>syncType</i>	Type of synchronization.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.40 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.

APPENDIX E. FILE DOCUMENTATION

<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.41 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.2.42 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.13.2.43 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.

APPENDIX E. FILE DOCUMENTATION

<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_-EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.13.2.44 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_-EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.45 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>sequenceCount</i>	A threadContingent unique number. The corresponding <i>ThreadCreate</i> event does have the same number.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.46 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	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.
<i>sequence-Count</i>	A <code>threadContingent</code> unique number. The corresponding Thread-Begin event does have the same number.

Since

Version 1.3

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.47 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	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.
<i>sequence-Count</i>	A <code>threadContingent</code> unique number. The corresponding Thread-Wait event does have the same number. OTF2_UNDEFINED_UINT64 in case no corresponding ThreadWait event exists.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.48 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>numberOfRequestedThreads</i>	Requested size of the team.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

APPENDIX E. FILE DOCUMENTATION

E.13.2.49 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.50 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.13.2.51 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	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.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.52 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.2.53 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	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.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.2.54 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	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.13.2.55 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	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.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.56 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>sequenceCount</i>	A threadContingent unique number. The corresponding <i>Thread-End</i> event does have the same number.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.13.2.57 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.13.3 Function Documentation

E.13.3.1 `void OTF2_EvtReaderCallbacks_Clear (OTF2_EvtReaderCallbacks *
 evtReaderCallbacks)`

Clears a struct for the event callbacks.

Parameters

<i>evtReaderCallbacks</i>	Handle to a struct previously allocated with OTF2_EvtReaderCallbacks_New .
---------------------------	--

E.13.3.2 `void OTF2_EvtReaderCallbacks_Delete (OTF2_EvtReaderCallbacks *
 evtReaderCallbacks)`

Deallocates a struct for the event callbacks.

Parameters

<i>evtReaderCallbacks</i>	Handle to a struct previously allocated with OTF2_EvtReaderCallbacks_New .
---------------------------	--

E.13.3.3 `OTF2_EvtReaderCallbacks* OTF2_EvtReaderCallbacks_New (void)`

Allocates a new struct for the event callbacks.

Returns

A newly allocated struct of type [OTF2_EvtReaderCallbacks](#).

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.3.4 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetBufferFlushCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_BufferFlush *bufferFlushCallback*)

Registers the callback for the BufferFlush event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>buffer-FlushCall-back</i>	Function which should be called for all BufferFlush events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.5 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetEnterCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>enterCall-back</i>	Function which should be called for all Enter events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

E.13.3.6 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks_SetLeaveCallback**
 (**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_Leave *leaveCallback*)

Registers the callback for the Leave event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>leaveCall-back</i>	Function which should be called for all Leave events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful
OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks*
 argument

E.13.3.7 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks_SetMeasurementOnOffCallback**
 (**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*, **OTF2_-**
EvtReaderCallback_MeasurementOnOff *measurementOnOffCallback*
)

Registers the callback for the MeasurementOnOff event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>measure-mentOnOff-Callback</i>	Function which should be called for all MeasurementOnOff events.

Since

Version 1.0

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.8 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetMetricCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_Metric *metricCallback*)

Registers the callback for the Metric event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.9 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*, **OTF2_EvtReaderCallback_MpiCollectiveBegin** *mpiCollectiveBeginCallback*)

Registers the callback for the MpiCollectiveBegin event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.10 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*, **OTF2_-**
EvtReaderCallback_MpiCollectiveEnd *mpiCollectiveEndCallback*
)

Registers the callback for the `MpiCollectiveEnd` event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEnd-Callback</i>	Function which should be called for all <code>MpiCollectiveEnd</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.11 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetMpiIrecvCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiIrecv *mpilrecvCallback*)

Registers the callback for the `MpiIrecv` event.

Parameters

E.13 oftf2/OTF2_EvtReaderCallbacks.h File Reference

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all MpiIrecv events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.12 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetMpiIrecvRequestCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiIrecvRequest *mpiIrecvRequestCallback*
)

Registers the callback for the MpiIrecvRequest event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all MpiIrecvRequest events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX E. FILE DOCUMENTATION

E.13.3.13 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiIsendCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiIsend *mpiIsendCallback*)

Registers the callback for the MpiIsend event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiIsend-Callback</i>	Function which should be called for all MpiIsend events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.13.3.14 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiIsendCompleteCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiIsendComplete
mpiIsendCompleteCallback)

Registers the callback for the MpiIsendComplete event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiIsend-Complete-Callback</i>	Function which should be called for all MpiIsendComplete events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.15 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetMpiRecvCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiRecv *mpiRecvCallback*)

Registers the callback for the `MpiRecv` event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiRecv-Callback</i>	Function which should be called for all <code>MpiRecv</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.16 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks -**
SetMpiRequestCancelledCallback (**OTF2_EvtReaderCallbacks**
* *evtReaderCallbacks*, **OTF2_EvtReaderCallback -**
MpiRequestCancelled *mpiRequestCancelledCallback*
)

Registers the callback for the `MpiRequestCancelled` event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiRequestCancelledCallback</i>	Function which should be called for all <code>MpiRequestCancelled</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.17 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiRequestTestCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiRequestTest *mpiRequestTestCallback*)

Registers the callback for the `MpiRequestTest` event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRequestTestCallback</i>	Function which should be called for all <code>MpiRequestTest</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.18 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiSendCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiSend *mpiSendCallback*)

Registers the callback for the `MpiSend` event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
---------------------------	---------------------------

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

<i>mpiSend-Callback</i>	Function which should be called for all MpiSend events.
-------------------------	---

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.19 OTF2_ErrorCode OTF2_EvtReaderCallbacks.SetOmpAcquireLockCallback
(OTF2_EvtReaderCallbacks * *evtReaderCallbacks*, OTF2_-
EvtReaderCallback_OmpAcquireLock *ompAcquireLockCallback*
)

Registers the callback for the OmpAcquireLock event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>ompAc-quireLock-Callback</i>	Function which should be called for all OmpAcquireLock events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.20 OTF2_ErrorCode OTF2_EvtReaderCallbacks.SetOmpForkCallback
(OTF2_EvtReaderCallbacks * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork event.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>ompForkCallback</i>	Function which should be called for all OmpFork events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.21 OTF2_ErrorCode OTF2_EvtReaderCallbacks.SetOmpJoinCallback
(**OTF2_EvtReaderCallbacks * *evtReaderCallbacks*,**
OTF2_EvtReaderCallback_OmpJoin *ompJoinCallback*)

Registers the callback for the OmpJoin event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>ompJoinCallback</i>	Function which should be called for all OmpJoin events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.3.22 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetOmpReleaseLockCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpReleaseLock *ompReleaseLockCallback*
)

Registers the callback for the OmpReleaseLock event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>ompReleaseLock-Callback</i>	Function which should be called for all OmpReleaseLock events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.13.3.23 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetOmpTaskCompleteCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpTaskComplete
ompTaskCompleteCallback)

Registers the callback for the OmpTaskComplete event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>ompTaskComplete-Callback</i>	Function which should be called for all OmpTaskComplete events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.24 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetOmpTaskCreateCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpTaskCreate *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCreateCallback</i>	Function which should be called for all OmpTaskCreate events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.25 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetOmpTaskSwitchCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpTaskSwitch *ompTaskSwitchCallback*)

Registers the callback for the OmpTaskSwitch event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskSwitchCallback</i>	Function which should be called for all OmpTaskSwitch events.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.26 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetParameterIntCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ParameterInt *parameterIntCallback*)

Registers the callback for the ParameterInt event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>parameter-IntCallback</i>	Function which should be called for all ParameterInt events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.27 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetParameterStringCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ParameterString *parameterStringCallback*)

Registers the callback for the ParameterString event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
----------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>parameter-StringCallback</i>	Function which should be called for all ParameterString events.
---------------------------------	---

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

**E.13.3.28 OTF2_ErrorCode OTF2_EvtReaderCallbacks_-
SetParameterUnsignedIntCallback (OTF2_EvtReaderCallbacks
* *evtReaderCallbacks*, OTF2_EvtReaderCallback_-
ParameterUnsignedInt *parameterUnsignedIntCallback*
)**

Registers the callback for the ParameterUnsignedInt event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.3.29 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaAcquireLockCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaAcquireLock *rmaAcquireLockCallback*
)

Registers the callback for the RmaAcquireLock event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaAcquireLock-Callback</i>	Function which should be called for all RmaAcquireLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.13.3.30 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaAtomicCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaAtomic *rmaAtomicCallback*)

Registers the callback for the RmaAtomic event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaAtomic-Callback</i>	Function which should be called for all RmaAtomic events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

APPENDIX E. FILE DOCUMENTATION

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.31 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaCollectiveBeginCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaCollectiveBegin
rmaCollectiveBeginCallback)

Registers the callback for the `RmaCollectiveBegin` event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaCollectiveBeginCallback</i>	Function which should be called for all <code>RmaCollectiveBegin</code> events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.32 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaCollectiveEndCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaCollectiveEnd
rmaCollectiveEndCallback)

Registers the callback for the `RmaCollectiveEnd` event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaCollectiveEndCallback</i>	Function which should be called for all <code>RmaCollectiveEnd</code> events.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.33 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaGetCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaGet *rmaGetCallback*)

Registers the callback for the RmaGet event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaGetCallback</i>	Function which should be called for all RmaGet events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.34 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaGroupSyncCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaGroupSync *rmaGroupSyncCallback*)

Registers the callback for the RmaGroupSync event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
---------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>rmaGroup-SyncCallback</i>	Function which should be called for all RmaGroupSync events.
------------------------------	--

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

```
E.13.335  OTF2_ErrorCode OTF2_EvtReaderCallbacks_-  
          SetRmaOpCompleteBlockingCallback ( OTF2_EvtReaderCallbacks  
          * evtReaderCallbacks, OTF2_EvtReaderCallback_-  
          RmaOpCompleteBlocking rmaOpCompleteBlockingCallback  
          )
```

Registers the callback for the RmaOpCompleteBlocking event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpCompleteBlockingCallback</i>	Function which should be called for all RmaOpCompleteBlocking events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.3.36 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_-**
SetRmaOpCompleteNonBlockingCallback (**OTF2_EvtReaderCallbacks**
*** *evtReaderCallbacks***, **OTF2_EvtReaderCallback_-**
RmaOpCompleteNonBlocking ***rmaOpCompleteNonBlockingCallback***
)

Registers the callback for the RmaOpCompleteNonBlocking event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaOp-CompleteNon-Blocking-Callback</i>	Function which should be called for all RmaOpCompleteNonBlocking events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.37 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_-**
SetRmaOpCompleteRemoteCallback (**OTF2_EvtReaderCallbacks**
*** *evtReaderCallbacks***, **OTF2_EvtReaderCallback_-**
RmaOpCompleteRemote ***rmaOpCompleteRemoteCallback***
)

Registers the callback for the RmaOpCompleteRemote event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaOp-CompleteR-remoteCall-back</i>	Function which should be called for all RmaOpCompleteRemote events.

APPENDIX E. FILE DOCUMENTATION

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.38 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetRmaOpTestCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaOpTest *rmaOpTestCallback*)

Registers the callback for the RmaOpTest event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpTestCallback</i>	Function which should be called for all RmaOpTest events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.39 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetRmaPutCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaPut *rmaPutCallback*)

Registers the callback for the RmaPut event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaPutCallback</i>	Function which should be called for all RmaPut events.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

```
E.13.3.40  OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaReleaseLockCallback
            ( OTF2_EvtReaderCallbacks * evtReaderCallbacks,
              OTF2_EvtReaderCallback_RmaReleaseLock rmaReleaseLockCallback
            )
```

Registers the callback for the RmaReleaseLock event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaReleaseLockCallback</i>	Function which should be called for all RmaReleaseLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

```
E.13.3.41  OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaRequestLockCallback
            ( OTF2_EvtReaderCallbacks * evtReaderCallbacks,
              OTF2_EvtReaderCallback_RmaRequestLock rmaRequestLockCallback
            )
```

Registers the callback for the RmaRequestLock event.

Parameters

APPENDIX E. FILE DOCUMENTATION

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaRequestLockCallback</i>	Function which should be called for all RmaRequestLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.42 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetRmaSyncCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaSync *rmaSyncCallback*)

Registers the callback for the RmaSync event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaSyncCallback</i>	Function which should be called for all RmaSync events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.3.43 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaTryLockCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaTryLock *rmaTryLockCallback*)

Registers the callback for the RmaTryLock event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaTry-LockCall-back</i>	Function which should be called for all RmaTryLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.13.3.44 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaWaitChangeCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaWaitChange *rmaWaitChangeCallback*)

Registers the callback for the RmaWaitChange event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaWait-Change-Callback</i>	Function which should be called for all RmaWaitChange events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

APPENDIX E. FILE DOCUMENTATION

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.45 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaWinCreateCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaWinCreate *rmaWinCreateCallback*)

Registers the callback for the RmaWinCreate event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaWin-CreateCall-back</i>	Function which should be called for all RmaWinCreate events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.46 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaWinDestroyCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaWinDestroy *rmaWinDestroyCallback*)

Registers the callback for the RmaWinDestroy event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaWinDe-destroyCall-back</i>	Function which should be called for all RmaWinDestroy events.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.47 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadAcquireLockCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadAcquireLock
threadAcquireLockCallback)

Registers the callback for the ThreadAcquireLock event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadAcquireLockCallback</i>	Function which should be called for all ThreadAcquireLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.48 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadBeginCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadBegin *threadBeginCallback*)

Registers the callback for the ThreadBegin event.

Parameters

APPENDIX E. FILE DOCUMENTATION

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadBeginCallback</i>	Function which should be called for all ThreadBegin events.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.49 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetThreadCreateCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadCreate *threadCreateCallback*)

Registers the callback for the ThreadCreate event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadCreateCallback</i>	Function which should be called for all ThreadCreate events.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.50 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetThreadEndCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadEnd *threadEndCallback*)

Registers the callback for the ThreadEnd event.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadEndCallback</i>	Function which should be called for all ThreadEnd events.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.51 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetThreadForkCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadFork *threadForkCallback*)

Registers the callback for the ThreadFork event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadForkCallback</i>	Function which should be called for all ThreadFork events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.52 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetThreadJoinCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadJoin *threadJoinCallback*)

Registers the callback for the ThreadJoin event.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadJoinCallback</i>	Function which should be called for all ThreadJoin events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.53 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetThreadReleaseLockCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadReleaseLock
threadReleaseLockCallback)

Registers the callback for the ThreadReleaseLock event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadReleaseLockCallback</i>	Function which should be called for all ThreadReleaseLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

E.13.3.54 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.**
SetThreadTaskCompleteCallback (**OTF2_EvtReaderCallbacks** *
evtReaderCallbacks, **OTF2_EvtReaderCallback_ThreadTaskComplete**
threadTaskCompleteCallback)

Registers the callback for the ThreadTaskComplete event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TaskCompleteCallback</i>	Function which should be called for all ThreadTaskComplete events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.13.3.55 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.**
SetThreadTaskCreateCallback (**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadTaskCreate
threadTaskCreateCallback)

Registers the callback for the ThreadTaskCreate event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TaskCreate-Callback</i>	Function which should be called for all ThreadTaskCreate events.

Since

Version 1.2

APPENDIX E. FILE DOCUMENTATION

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.56 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetThreadTaskSwitchCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadTaskSwitch
threadTaskSwitchCallback)

Registers the callback for the ThreadTaskSwitch event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TaskSwitch-Callback</i>	Function which should be called for all ThreadTaskSwitch events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.57 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetThreadTeamBeginCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadTeamBegin
threadTeamBeginCallback)

Registers the callback for the ThreadTeamBegin event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TeamBegin-Callback</i>	Function which should be called for all ThreadTeamBegin events.

E.13 otf2/OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.58 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadTeamEndCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadTeamEnd *threadTeamEndCallback*)

Registers the callback for the ThreadTeamEnd event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTeamEndCallback</i>	Function which should be called for all ThreadTeamEnd events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.59 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadWaitCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadWait *threadWaitCallback*)

Registers the callback for the ThreadWait event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
---------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>threadWait-Callback</i>	Function which should be called for all ThreadWait events.
----------------------------	--

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.13.3.60 **OTF2_StatusCode** **OTF2_EvtReaderCallbacks.SetUnknownCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_Unknown *unknownCallback*)

Registers the callback for the Unknown event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	Function which should be called for all unknown events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.14 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>
```

E.14 otf2/OTF2_EvtWriter.h File Reference

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_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.

APPENDIX E. FILE DOCUMENTATION

- [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)

E.14 oftf2/OTF2_EvtWriter.h File Reference

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 numberOfRequestedThreads)

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.

APPENDIX E. FILE DOCUMENTATION

- [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.

E.14 otf2/OTF2_EvtWriter.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- [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.

E.14 otf2/OTF2_EvtWriter.h File Reference

- [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.14.1 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.14.2 Function Documentation

- E.14.2.1** [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

<i>writer</i>	Writer object.
---------------	----------------

APPENDIX E. FILE DOCUMENTATION

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>stopTime</i>	The time the buffer flush finished.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.2 **OTF2_ErrorCode** **OTF2_EvtWriter_ClearRewindPoint** (**OTF2_EvtWriter *** *writer*, **uint32_t** *rewindId*)

Please give me a documantation.

Parameters

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

Since

Version 1.1

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.3 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

E.14 otf2/OTF2_EvtWriter.h File Reference

<i>region</i>	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.14.2.4 `OTF2_ErrorCode OTF2_EvtWriter_GetLocationID (const OTF2_EvtWriter * writer, OTF2_LocationRef * locationID)`

Function to get the location ID of a writer object.

Parameters

<i>writer</i>	Writer object which has to be deleted
<i>locationID</i>	Pointer to a variable where the ID is returned in

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.5 `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

	<i>writer</i>	Writer object.
out	<i>numberOfEvents</i>	Return pointer to the number of events.

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.6 **OTF2_ErrorCode** **OTF2_EvtWriter_GetUserData** (**const** **OTF2_EvtWriter** * *writer*, **void** ** *userData*)

Function to get the location of a writer object.

Parameters

	<i>writer</i>	Writer object.
out	<i>userData</i>	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.14.2.7 **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

	<i>writer</i>	Writer object.
	<i>attributeList</i>	Generic attributes for the event.
	<i>time</i>	The time for this event.
	<i>region</i>	Needs to be defined in a definition record References a <i>Region</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_REGION</i> is available.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.8 **OTF2_ErrorCode** **OTF2_EvtWriter_MeasurementOnOff** (**OTF2_EvtWriter** * *writer*, **OTF2_AttributeList** * *attributeList*, **OTF2_TimeStamp** *time*, **OTF2_MeasurementMode** *measurementMode*)

Records an MeasurementOnOff event.

E.14 oftf2/OTF2_EvtWriter.h File Reference

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

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>measurementMode</i>	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.14.2.9 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>metric</i>	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.
<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding MetricMember definitions.
<i>metricValues</i>	List of metric values.

Since

Version 1.0

Returns

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

E.14.2.10 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

Since

Version 1.0

Returns

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

E.14.2.11 **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

<i>writer</i>	Writer object.
---------------	----------------

E.14 oftf2/OTF2_EvtWriter.h File Reference

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a Comm definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_COMM is available.
<i>root</i>	MPI rank of root in <code>communicator</code> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.12 `OTF2_ErrorCode OTF2_EvtWriter_Mpilrecv (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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communicator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

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

E.14.2.13 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the requested receive

Since

Version 1.0

Returns

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

E.14.2.14 **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

E.14 oftf2/OTF2_EvtWriter.h File Reference

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communicator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.15 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.14.2.16 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>sender</i>	MPI rank of sender in <i>communicator</i> .
<i>communicator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.17 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

E.14 oftf2/OTF2_EvtWriter.h File Reference

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.18 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.19 **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

APPENDIX E. FILE DOCUMENTATION

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi- cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.20 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>lockID</i>	ID of the lock.
<i>acqui- sitionOrder</i>	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

E.14 otf2/OTF2_EvtWriter.h File Reference

Deprecated

In version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.21 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>numberOfRequestedThreads</i>	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.14.2.22 `OTF2_ErrorCode OTF2_EvtWriter_OmpJoin (OTF2_EvtWriter * writer,
OTF2_AttributeList * attributeList, OTF2_TimeStamp time)`

Records an OmpJoin event.

APPENDIX E. FILE DOCUMENTATION

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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	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.14.2.23 `OTF2_StatusCode 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.

E.14 oftf2/OTF2_EvtWriter.h File Reference

Since

Version 1.0

Deprecated

In version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.24 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	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.14.2.25 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	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.14.2.26 `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

<i>writer</i>	Writer object.
---------------	----------------

E.14 otf2/OTF2_EvtWriter.h File Reference

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	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.14.2.27 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>parameter</i>	Parameter ID. References a <i>Parameter</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_PARAMETER</i> is available.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.14.2.28 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>parameter</i>	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.
<i>string</i>	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.14.2.29 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

E.14 oftf2/OTF2_EvtWriter.h File Reference

<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.30 `OTF2_ErrorCode OTF2_EvtWriter.Rewind (OTF2_EvtWriter * writer,
uint32_t rewindId)`

Please give me a documantation.

Parameters

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

Since

Version 1.1

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.31 `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 aquired by the process.

Parameters

<i>writer</i>	Writer object.
---------------	----------------

APPENDIX E. FILE DOCUMENTATION

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.32 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>type</i>	Type of atomic operation.
<i>bytesSent</i>	Bytes sent to target.
<i>bytesReceived</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

E.14 otf2/OTF2_EvtWriter.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.33 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.34 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>syncLevel</i>	Synchronization level of this collective operation.

APPENDIX E. FILE DOCUMENTATION

<i>win</i>	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.
<i>root</i>	Root process for this operation.
<i>bytesSent</i>	Bytes sent in operation.
<i>bytesReceived</i>	Bytes receives in operation.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.35 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14 otf2/OTF2_EvtWriter.h File Reference

E.14.2.36 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	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.
<i>group</i>	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.14.2.37 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

APPENDIX E. FILE DOCUMENTATION

<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.38 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14 otf2/OTF2_EvtWriter.h File Reference

E.14.2.39 **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 local completion of an RMA operation.

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.40 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.41 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes sent to target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.42 `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.

E.14 of2/OTF2_EvtWriter.h File Reference

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	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.14.2.43 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.44 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
<i>remote</i>	Rank of the locked remote process.
<i>syncType</i>	Type of synchronization.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.45 `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

E.14 oftf2/OTF2_EvtWriter.h File Reference

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.46 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.

APPENDIX E. FILE DOCUMENTATION

E.14.2.47 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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.14.2.48 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	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

E.14 otf2/OTF2_EvtWriter.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.49 **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

<i>writer</i>	Writer object.
<i>location</i>	Location ID.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.50 **OTF2_ErrorCode** OTF2_EvtWriter.SetUserData (OTF2_EvtWriter *
writer, void * *userData*)

Function to set user defined data to a writer object.

Parameters

<i>writer</i>	Writer object.
<i>userData</i>	User provided data. Can be queried with <i>OTF2_EvtWriter_GetUserData</i> .

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.51 **OTF2_ErrorCode** OTF2_EvtWriter.StoreRewindPoint (OTF2_EvtWriter
* *writer*, uint32_t *rewindId*)

Please give me a documantation.

Parameters

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

Since

Version 1.1

Returns

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

E.14.2.52 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.14.2.53 **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

E.14 oftf2/OTF2_EvtWriter.h File Reference

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadContingent</i>	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.
<i>sequence-Count</i>	A threadContingent unique number. The corresponding Thread-Create event does have the same number.

Since

Version 1.3

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.14.2.54 `OTF2_StatusCode 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadContingent</i>	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.
<i>sequence-Count</i>	A threadContingent unique number. The corresponding Thread-Begin event does have the same number.

Since

Version 1.3

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.14.2.55 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadContingent</i>	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.
<i>sequence-Count</i>	A <code>threadContingent</code> 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.14.2.56 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>model</i>	The threading paradigm this event took place.
<i>numberOfRequestedThreads</i>	Requested size of the team.

E.14 oftf2/OTF2_EvtWriter.h File Reference

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.57 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>model</i>	The threading paradigm this event took place.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.14.2.58 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

APPENDIX E. FILE DOCUMENTATION

<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.14.2.59 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	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.14 otf2/OTF2_EvtWriter.h File Reference

E.14.2.60 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	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.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	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.14.2.61 **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

<i>writer</i>	Writer object.
---------------	----------------

APPENDIX E. FILE DOCUMENTATION

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	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.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	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.14.2.62 **OTF2_StatusCode** **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	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.14 otf2/OTF2_EvtWriter.h File Reference

E.14.2.63 `OTF2_StatusCode 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	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.14.2.64 `OTF2_StatusCode 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadContingent</i>	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.
<i>sequence-Count</i>	A threadContingent unique number. The corresponding Thread-End event does have the same number.

Since

Version 1.3

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.15 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_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.

E.15 otf2/OTF2_GeneralDefinitions.h File Reference

- #define `OTF2_UNDEFINED_GROUP` ((`OTF2_GroupRef`)OTF2_UNDEFINED_-
UINT32)

The invalid value for a reference to a `Group` 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_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](#) 4
Minor version number of this OTF2 version.
- #define [OTF2_VERSION_BUGFIX](#) 0
Bugfix version number of this OTF2 version.
- #define [OTF2_VERSION_SUFFIX](#) ""
Any string suffix of this OTF2 version.
- #define [OTF2_VERSION](#) "1.4"
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 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.

E.15 otf2/OTF2_GeneralDefinitions.h File Reference

- 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 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 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_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_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.

E.15 otf2/OTF2_GeneralDefinitions.h File Reference

- 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_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_MAX` = 9 }
 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 }`

List of known paradigms.

- `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,`

E.15 otf2/OTF2_GeneralDefinitions.h File Reference

```
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 basic data types.

E.15.1 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.15.2 Enumeration Type Documentation

E.15.2.1 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 callbacks (see *Operating OTF2 in an collective context*). Any value different than *OTF2_CALLBACK_SUCCESS* is treated as an error and the calling function will return *OTF2_ERROR_COLLECTIVE_CALLBACK* to its caller. As the name *OTF2_CALLBACK_INTERRUPT* does not really fit, a second alias is named *OTF2_CALLBACK_ERROR* provided.

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* Signalling an error in the callback.

E.15.2.2 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.15.2.3 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.15 otf2/OTF2_GeneralDefinitions.h File Reference

E.15.2.4 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.15.2.5 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.15.2.6 enum OTF2_FlushType_enum

Defines whether the recorded data is flushed to a file or not.

Enumerator:

OTF2_NO_FLUSH Flushing will be suppressed when running out of memory.

OTF2_FLUSH Recorded data is flushed when running out of memory.

E.15.2.7 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_MAX Max entry.

E.15.2.8 enum OTF2_Paradigm_enum

List of known paradigms.

Since

Version 1.1

E.15 otf2/OTF2_GeneralDefinitions.h File Reference

Enumerator:

OTF2_PARADIGM_UNKNOWN An unknown paradigm.

OTF2_PARADIGM_USER User instrumentation.

OTF2_PARADIGM_COMPILER Compiler instrumentation.

OTF2_PARADIGM_OPENMP OpenMP.

OTF2_PARADIGM_MPI MPI.

OTF2_PARADIGM_CUDA CUDA.

OTF2_PARADIGM_MEASUREMENT_SYSTEM The measurement software.

Since

Version 1.2.

OTF2_PARADIGM_PTHREAD POSIX threads.

Since

Version 1.3.

OTF2_PARADIGM_HMPP HMPP.

Since

Version 1.3.

OTF2_PARADIGM_OMPSS OmpSs.

Since

Version 1.3.

OTF2_PARADIGM_HARDWARE Hardware.

Since

Version 1.3.

OTF2_PARADIGM_GASPI GASPI.

Since

Version 1.4.

OTF2_PARADIGM_UPC Unified Parallel C (UPC).

Since

Version 1.4.

OTF2_PARADIGM_SHMEM SGI SHMEM, Cray SHMEM, OpenShmem.

Since

Version 1.4.

E.15.2.9 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.15.2.10 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

E.16 otf2/OTF2_GlobalDefReader.h File Reference

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

E.16 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.

APPENDIX E. FILE DOCUMENTATION

- [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.16.1 Detailed Description

This is the definition reader.

E.16.2 Function Documentation

E.16.2.1 [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

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.
out	<i>recordsRead</i>	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given recordsToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finished his job by checking recordsRead < recordsToRead.

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.16.2.2 [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.

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

<i>reader</i>	This given reader object will be setted up with new callback functions.
<i>callbacks</i>	Struct which holds a function pointer for each record type. OTF2_GlobalDefReaderCallbacks_New .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.17 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_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_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.

APPENDIX E. FILE DOCUMENTATION

- 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_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)

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

Function pointer definition for the callback which is triggered by a [Location-Group](#) 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 [Location-GroupProperty](#) 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 [Location-Property](#) 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_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](#)

APPENDIX E. FILE DOCUMENTATION

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_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 [SystemTreeNode](#) 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 [SystemTreeNodeDomain](#) 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 [SystemTreeNodeProperty](#) 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 holds all global definition record callbacks.

Functions

- void [OTF2_GlobalDefReaderCallbacks_Clear](#)([OTF2_GlobalDefReaderCallbacks](#) *globalDefReaderCallbacks)

Clears a struct for the global definition callbacks.

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

- 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_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)

APPENDIX E. FILE DOCUMENTATION

Registers the callback for the [Group](#) 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_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.

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

- [OTF2_ErrorCode](#) [OTF2_GlobalDefReaderCallbacks_SetRmaWinCallback](#) ([OTF2_GlobalDefReaderCallbacks](#) *globalDefReaderCallbacks, [OTF2_GlobalDefReaderCallback_RmaWin](#) rmaWinCallback)
Registers the callback for the [RmaWin](#) 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.17.1 Detailed Description

This defines the callbacks for the global definition reader.

Source Template:

templates/OTF2_GlobalDefReaderCallbacks.tmpl.h

E.17.2 Typedef Documentation

- E.17.2.1** `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Attribute definition.
<i>name</i>	Name of the attribute. References a String definition.
<i>description</i>	Description of the attribute. References a String definition. Since version 1.4.
<i>type</i>	Type of the attribute value.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17.2.2 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Callpath definition.
<i>parent</i>	The parent of this callpath. References a Callpath definition.
<i>region</i>	The region of this callpath. References a Region definition.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

E.17.2.3 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Callsite definition.
<i>sourceFile</i>	The source file where this call was made. References a String definition.
<i>lineNumber</i>	Line number in the source file where this call was made.
<i>enteredRegion</i>	The region which was called. References a Region definition.
<i>leftRegion</i>	The region which made the call. References a Region definition.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17.2.4 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>cartTopology</i>	Parent CartTopology definition to which this one is a supplementary definition. References a CartTopology definition.

APPENDIX E. FILE DOCUMENTATION

<i>rank</i>	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
<i>numberOfDimensions</i>	Number of dimensions.
<i>coordinates</i>	Coordinates, indexed by dimension.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.5 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>self</i>	The unique identifier for this <i>CartDimension</i> definition.
<i>name</i>	The name of the cartesian topology dimension. References a <i>String</i> definition.
<i>size</i>	The size of the cartesian topology dimension.
<i>cartPeriodicity</i>	Periodicity of the cartesian topology dimension.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

E.17.2.6 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this CartTopology definition.
<i>name</i>	The name of the topology. References a String definition.
<i>communi- cator</i>	Communicator object used to create the topology. References a Comm definition.
<i>num- berOfDi- mensions</i>	Number of dimensions.
<i>cartDimen- sions</i>	The dimensions of this topology. References a CartDimension definition.

Since

Version 1.3

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17.2.7 `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)`.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>timerResolution</i>	Ticks per seconds.
<i>globalOffset</i>	A timestamp smaller than all event timestamps.
<i>traceLength</i>	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.17.2.8 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Comm definition.
<i>name</i>	The name given by calling <code>MPI_Comm_set_name</code> on this communicator. Or the empty name to indicate that no name was given. References a String definition.
<i>group</i>	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.
<i>parent</i>	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

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.9 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>self</i>	The unique identifier for this <i>Group</i> definition.
<i>name</i>	Name of this group References a <i>String</i> definition.
<i>groupType</i>	The type of this group. Since version 1.2.
<i>paradigm</i>	The paradigm of this communication group. Since version 1.2.
<i>groupFlags</i>	Flags for this group. Since version 1.2.
<i>numberOfMembers</i>	The number of members in this group.
<i>members</i>	The identifiers of the group members.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.10 `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.

APPENDIX E. FILE DOCUMENTATION

The location definition.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Location definition.
<i>name</i>	Name of the location References a String definition.
<i>location-Type</i>	Location type.
<i>numberOfEvents</i>	Number of events this location has recorded.
<i>location-Group</i>	Location group which includes this location. References a Location-Group definition.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17.2.11 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this LocationGroup definition.
<i>name</i>	Name of the group. References a String definition.
<i>location-GroupType</i>	Type of this group.
<i>systemTreeParent</i>	Parent of this location group in the system tree. References a SystemTreeNode definition.

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.12 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>location-Group</i>	Parent <i>LocationGroup</i> definition to which this one is a supplementary definition. References a <i>LocationGroup</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.13 `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

APPENDIX E. FILE DOCUMENTATION

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>location</i>	Parent Location definition to which this one is a supplementary definition. References a Location definition.
<i>name</i>	Name of the property. References a String definition.
<i>value</i>	Property value. References a String definition.

Since

Version 1.3

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17.2.14 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this MetricClass definition.
<i>numberOfMetrics</i>	Number of metrics within the set.
<i>metricMembers</i>	List of metric members. References a MetricMember definition.
<i>metricOccurrence</i>	Defines occurrence of a metric set.
<i>recorderKind</i>	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

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.15 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>metricClass</i>	Parent <i>MetricClass</i> definition to which this one is a supplementary definition. References a <i>MetricClass</i> definition.
<i>recorder</i>	The location which recorded the referenced metric class. References a <i>Location</i> definition.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.16 `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

APPENDIX E. FILE DOCUMENTATION

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this MetricClass definition.
<i>metricClass</i>	The instanced MetricClass . This metric class must be of kind OTF2_RECORDER_KIND_ABSTRACT . References a MetricClass definition.
<i>recorder</i>	Recorder of the metric: location ID. References a Location definition.
<i>metric-Scope</i>	Defines type of scope: location, location group, system tree node, or a generic group of locations.
<i>scope</i>	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.17.2.17 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this MetricMember definition.
<i>name</i>	Name of the metric. References a String definition.
<i>description</i>	Description of the metric. References a String definition.
<i>metricType</i>	Metric type: PAPI, etc.
<i>metricMode</i>	Metric mode: accumulative, fix, relative, etc.

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

<i>valueType</i>	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.
<i>metricBase</i>	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.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$, 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.
<i>unit</i>	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.17.2.18 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this Parameter definition.
<i>name</i>	Name of the parameter (variable name etc.) References a String definition.
<i>parameter-Type</i>	Type of the parameter, OTF2_ParameterType for possible types.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.19 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>self</i>	The unique identifier for this <i>Region</i> definition.
<i>name</i>	Name of the region (demangled name if available). References a <i>String</i> definition.
<i>canonical-Name</i>	Alternative name of the region (e.g. mangled name). References a <i>String</i> definition. Since version 1.1.
<i>description</i>	A more detailed description of this region. References a <i>String</i> definition.
<i>regionRole</i>	Region role. Since version 1.1.
<i>paradigm</i>	Paradigm. Since version 1.1.
<i>regionFlags</i>	Region flags. Since version 1.1.
<i>sourceFile</i>	The source file where this region was declared. References a <i>String</i> definition.
<i>beginLineNumber</i>	Starting line number of this region in the source file.
<i>endLineNumber</i>	Ending line number of this region in the source file.

Since

Version 1.0

E.17 oftf2/OTF2_GlobalDefReaderCallbacks.h File Reference

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17.2.20 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this RmaWin definition.
<i>name</i>	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a String definition.
<i>comm</i>	Communicator object used to create the window. References a Comm definition.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17.2.21 `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 definitions.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
-----------------	---

APPENDIX E. FILE DOCUMENTATION

<i>self</i>	The unique identifier for this String definition.
<i>string</i>	The string, null terminated.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.17.2.22 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>self</i>	The unique identifier for this SystemTreeNode definition.
<i>name</i>	Free form instance name of this node. References a String definition.
<i>className</i>	Free form class name of this node References a String definition.
<i>parent</i>	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.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

E.17.2.23 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>systemTreeDomain</i>	The domain in which the referenced <i>SystemTreeNode</i> operates in.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.17.2.24 `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

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.2.25 `typedef OTF2_CallbackCode(* OTF2_GlobalDefReaderCallback_ -
Unknown)(void *userData)`

Function pointer definition for the callback which is triggered by an unknown definition record.

Parameters

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
-----------------	---

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.17.3 Function Documentation

E.17.3.1 `void OTF2_GlobalDefReaderCallbacks_Clear (OTF2_ -
GlobalDefReaderCallbacks * globalDefReaderCallbacks
)`

Clears a struct for the global definition callbacks.

Parameters

<i>globalDef-Reader-Callbacks</i>	Handle to a struct previously allocated with <i>OTF2_GlobalDefReaderCallbacks_New</i> .
-----------------------------------	---

E.17.3.2 `void OTF2_GlobalDefReaderCallbacks_Delete (OTF2_ -
GlobalDefReaderCallbacks * globalDefReaderCallbacks
)`

Deallocates a struct for the global definition callbacks.

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

Parameters

<i>globalDef-Reader-Callbacks</i>	Handle to a struct previously allocated with OTF2_GlobalDefReaderCallbacks_New .
-----------------------------------	--

E.17.3.3 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.17.3.4 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetAttributeCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Attribute *attributeCallback*)

Registers the callback for the [Attribute](#) definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>attribute-Callback</i>	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.17.3.5 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_SetCallpathCallback**
 (**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*,
OTF2_GlobalDefReaderCallback_Callpath *callpathCallback*)

Registers the callback for the *Callpath* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>callpath-Callback</i>	Function which should be called for all <i>Callpath</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.17.3.6 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_SetCallsiteCallback**
 (**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*,
OTF2_GlobalDefReaderCallback_Callsite *callsiteCallback*)

Registers the callback for the *Callsite* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>callsite-Callback</i>	Function which should be called for all <i>Callsite</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

**E.17.3.7 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_-
SetCartCoordinateCallback (OTF2_GlobalDefReaderCallbacks *
globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-
CartCoordinate cartCoordinateCallback)**

Registers the callback for the *CartCoordinate* definition.

Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>cartCoordi- nateCall- back</i>	Function which should be called for all <i>CartCoordinate</i> definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

**E.17.3.8 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_-
SetCartDimensionCallback (OTF2_GlobalDefReaderCallbacks *
globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_-
CartDimension cartDimensionCallback)**

Registers the callback for the *CartDimension* definition.

Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>cartDimen- sionCall- back</i>	Function which should be called for all <i>CartDimension</i> definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.9 `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetCartTopologyCallback (OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_CartTopology cartTopologyCallback)`

Registers the callback for the *CartTopology* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>cartTopologyCallback</i>	Function which should be called for all <i>CartTopology</i> definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.10 `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetClockPropertiesCallback (OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_ClockProperties clockPropertiesCallback)`

Registers the callback for the *ClockProperties* definition.

Parameters

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>clockPropertiesCallback</i>	Function which should be called for all <i>ClockProperties</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.11 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCommCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Comm *commCallback*)

Registers the callback for the *Comm* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>commCallback</i>	Function which should be called for all <i>Comm</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.12 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_SetGroupCallback**
(**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*,
OTF2_GlobalDefReaderCallback_Group *groupCallback*)

Registers the callback for the *Group* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>groupCallback</i>	Function which should be called for all <i>Group</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.17.3.13 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_SetLocationCallback**
(**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*,
OTF2_GlobalDefReaderCallback_Location *locationCallback*)

Registers the callback for the *Location* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>location-Callback</i>	Function which should be called for all <i>Location</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.14 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_**
SetLocationGroupCallback (OTF2_GlobalDefReaderCallbacks *
***globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_**
LocationGroup *locationGroupCallback*)

Registers the callback for the *LocationGroup* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>location-GroupCallback</i>	Function which should be called for all <i>LocationGroup</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.15 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_**
SetLocationGroupPropertyCallback (OTF2_GlobalDefReaderCallbacks
*** *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_**
LocationGroupProperty *locationGroupPropertyCallback*
)

Registers the callback for the *LocationGroupProperty* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
-----------------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>location-GroupPropertyCallback</i>	Function which should be called for all <i>LocationGroupProperty</i> definitions.
---------------------------------------	---

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.16 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_-SetLocationPropertyCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_ *LocationProperty locationPropertyCallback*)

Registers the callback for the *LocationProperty* definition.

Parameters

<i>globalDefReaderCallbacks</i>	Struct for all callbacks.
<i>locationPropertyCallback</i>	Function which should be called for all <i>LocationProperty</i> definitions.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

E.17.3.17 **OTF2_**`ErrorCode` **OTF2_GlobalDefReaderCallbacks_SetMetricClassCallback**
(**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*,
OTF2_GlobalDefReaderCallback_MetricClass *metricClassCallback*)

Registers the callback for the *MetricClass* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metric-ClassCallback</i>	Function which should be called for all <i>MetricClass</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.18 **OTF2_**`ErrorCode` **OTF2_GlobalDefReaderCallbacks_-SetMetricClassRecorderCallback** (**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*, **OTF2_GlobalDefReaderCallback_-MetricClassRecorder** *metricClassRecorderCallback*)

Registers the callback for the *MetricClassRecorder* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metric-ClassRecorderCallback</i>	Function which should be called for all <i>MetricClassRecorder</i> definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.19 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_-SetMetricInstanceCallback** (**OTF2_GlobalDefReaderCallbacks *** *globalDefReaderCallbacks*, **OTF2_GlobalDefReaderCallback_-MetricInstance** *metricInstanceCallback*)

Registers the callback for the *MetricInstance* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metricInstanceCallback</i>	Function which should be called for all <i>MetricInstance</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.20 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_-SetMetricMemberCallback** (**OTF2_GlobalDefReaderCallbacks *** *globalDefReaderCallbacks*, **OTF2_GlobalDefReaderCallback_-MetricMember** *metricMemberCallback*)

Registers the callback for the *MetricMember* definition.

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metricMemberCallback</i>	Function which should be called for all <i>MetricMember</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.21 `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetParameterCallback (OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Parameter parameterCallback)`

Registers the callback for the *Parameter* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>parameter-Callback</i>	Function which should be called for all <i>Parameter</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.22 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetRegionCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_Region *regionCallback*)

Registers the callback for the *Region* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>regionCallback</i>	Function which should be called for all <i>Region</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.17.3.23 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetRmaWinCallback (OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*, OTF2_GlobalDefReaderCallback_RmaWin *rmaWinCallback*)

Registers the callback for the *RmaWin* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>rmaWin-Callback</i>	Function which should be called for all <i>RmaWin</i> definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.24 **OTF2_StatusCode** **OTF2_GlobalDefReaderCallbacks_SetStringCallback**
(**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*,
OTF2_GlobalDefReaderCallback_String *stringCallback*)

Registers the callback for the *String* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>stringCallback</i>	Function which should be called for all <i>String</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.25 **OTF2_StatusCode** **OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeCallback** (**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*, **OTF2_GlobalDefReaderCallback_SystemTreeNode** *systemTreeNodeCallback*)

Registers the callback for the *SystemTreeNode* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>systemTreeNodeCallback</i>	Function which should be called for all <i>SystemTreeNode</i> definitions.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17.3.26 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_-SetSystemTreeNodeDomainCallback** (**OTF2_GlobalDefReaderCallbacks** * **globalDefReaderCallbacks**, **OTF2_GlobalDefReaderCallback_-SystemTreeNodeDomain** **systemTreeNodeDomainCallback**)

Registers the callback for the *SystemTreeNodeDomain* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>systemTreeNodeDomainCallback</i>	Function which should be called for all <i>SystemTreeNodeDomain</i> definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.17 otf2/OTF2_GlobalDefReaderCallbacks.h File Reference

E.17.3.27 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_**
SetSystemTreeNodePropertyCallback (**OTF2_GlobalDefReaderCallbacks**
***** *globalDefReaderCallbacks*, **OTF2_GlobalDefReaderCallback_**
SystemTreeNodeProperty *systemTreeNodePropertyCallback*
)

Registers the callback for the *SystemTreeNodeProperty* definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>systemTreeNodePropertyCallback</i>	Function which should be called for all <i>SystemTreeNodeProperty</i> definitions.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.17.3.28 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_SetUnknownCallback**
(**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*,
OTF2_GlobalDefReaderCallback_Unknown *unknownCallback*)

Registers the callback for an unknown definition.

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	Function which should be called for all Unknown definitions.

Returns

[*OTF2_SUCCESS*](#) if successful

[*OTF2_ERROR_INVALID_ARGUMENT*](#) for an invalid defReaderCallbacks argument

E.18 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_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_WriteCallpath \(OTF2_GlobalDefWriter *writerHandle, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region\)*](#)

E.18 otf2/OTF2_GlobalDefWriter.h File Reference

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_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.

- [OTF2_ErrorCode](#) [OTF2_GlobalDefWriter_WriteLocationGroup](#) ([OTF2_GlobalDefWriter](#) *writerHandle, [OTF2_LocationGroupRef](#) self, [OTF2_StringRef](#) name, [OTF2_LocationGroupType](#) locationGroupType, [OTF2_SystemTreeNodeRef](#) systemTreeParent)

APPENDIX E. FILE DOCUMENTATION

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_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)`

E.18 otf2/OTF2_GlobalDefWriter.h File Reference

Writes a [RmaWin](#) 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](#) systemTreeNode, [OTF2_SystemTreeDomain](#) systemTreeDomain)

Writes a [SystemTreeNodeDomain](#) definition record into the GlobalDefWriter.

- [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.

E.18.1 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.templ.h

E.18.2 Function Documentation

E.18.2.1 [OTF2_ErrorCode](#) [OTF2_GlobalDefWriter_GetNumberOfDefinitions](#) ([OTF2_GlobalDefWriter](#) * *writerHandle*, [uint64_t](#) * *numberOfDefinitions*)

Returns the current number of written definitions of a global definition writer.

Parameters

	<i>writerHandle</i>	Handle to the global definition writer.
--	---------------------	---

APPENDIX E. FILE DOCUMENTATION

out	<i>numberOfDefinitions</i>	Storage for the number of definitions.
-----	----------------------------	--

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.18.2.2 `OTF2_StatusCode OTF2_GlobalDefWriter_GetNumberOfLocations (OTF2_GlobalDefWriter * writerHandle, uint64_t * numberOfLocations)`

Returns the current number of written location definitions of a global definition writer.

Parameters

	<i>writerHandle</i>	Handle to the global definition writer.
out	<i>numberOfLocations</i>	Storage for the number of locations.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.18.2.3 `OTF2_StatusCode 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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <i>Attribute</i> definition.
<i>name</i>	Name of the attribute. References a <i>String</i> definition.
<i>description</i>	Description of the attribute. References a <i>String</i> definition. Since version 1.4.
<i>type</i>	Type of the attribute value.

E.18 otf2/OTF2_GlobalDefWriter.h File Reference

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.18.2.4 `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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <i>Callpath</i> definition.
<i>parent</i>	The parent of this callpath. References a <i>Callpath</i> definition.
<i>region</i>	The region of this callpath. References a <i>Region</i> definition.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.18.2.5 `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

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

APPENDIX E. FILE DOCUMENTATION

<i>self</i>	The unique identifier for this Callsite definition.
<i>sourceFile</i>	The source file where this call was made. References a String definition.
<i>lineNumber</i>	Line number in the source file where this call was made.
<i>enteredRegion</i>	The region which was called. References a Region definition.
<i>leftRegion</i>	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.18.2.6 `OTF2_StatusCode 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

<i>writerHandle</i>	The writer handle.
<i>cartTopology</i>	Parent CartTopology definition to which this one is a supplementary definition. References a CartTopology definition.
<i>rank</i>	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
<i>numberOfDimensions</i>	Number of dimensions.
<i>coordinates</i>	Coordinates, indexed by dimension.

Since

Version 1.3

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.18 otf2/OTF2_GlobalDefWriter.h File Reference

E.18.2.7 `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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this CartDimension definition.
<i>name</i>	The name of the cartesian topology dimension. References a String definition.
<i>size</i>	The size of the cartesian topology dimension.
<i>cartPeriodicity</i>	Periodicity of the cartesian topology dimension.

Since

Version 1.3

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.18.2.8 `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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this CartTopology definition.

APPENDIX E. FILE DOCUMENTATION

<i>name</i>	The name of the topology. References a String definition.
<i>communicator</i>	Communicator object used to create the topology. References a Comm definition.
<i>numberOfDimensions</i>	Number of dimensions.
<i>cartDimensions</i>	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.18.2.9 OTF2_StatusCode 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

<i>writerHandle</i>	The writer handle.
<i>timerResolution</i>	Ticks per seconds.
<i>globalOffset</i>	A timestamp smaller than all event timestamps.
<i>traceLength</i>	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.18 otf2/OTF2_GlobalDefWriter.h File Reference

E.18.2.10 `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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this Comm definition.
<i>name</i>	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.
<i>group</i>	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.
<i>parent</i>	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.18.2.11 `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

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

APPENDIX E. FILE DOCUMENTATION

<i>self</i>	The unique identifier for this Group definition.
<i>name</i>	Name of this group References a String definition.
<i>groupType</i>	The type of this group. Since version 1.2.
<i>paradigm</i>	The paradigm of this communication group. Since version 1.2.
<i>groupFlags</i>	Flags for this group. Since version 1.2.
<i>numberOfMembers</i>	The number of members in this group.
<i>members</i>	The identifiers of the group members.

Since

Version 1.0

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.18.2.12 `OTF2_StatusCode 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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this Location definition.
<i>name</i>	Name of the location References a String definition.
<i>locationType</i>	Location type.
<i>numberOfEvents</i>	Number of events this location has recorded.
<i>locationGroup</i>	Location group which includes this location. References a Location-Group definition.

Since

Version 1.0

E.18 otf2/OTF2_GlobalDefWriter.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.18.2.13 **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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <i>LocationGroup</i> definition.
<i>name</i>	Name of the group. References a <i>String</i> definition.
<i>locationGroupType</i>	Type of this group.
<i>systemTreeParent</i>	Parent of this location group in the system tree. References a <i>SystemTreeNode</i> definition.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.18.2.14 **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

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

APPENDIX E. FILE DOCUMENTATION

<i>location-Group</i>	Parent <i>LocationGroup</i> definition to which this one is a supplementary definition. References a <i>LocationGroup</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.3

Returns

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

E.18.2.15 **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

<i>writerHandle</i>	The writer handle.
<i>location</i>	Parent <i>Location</i> definition to which this one is a supplementary definition. References a <i>Location</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.3

Returns

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

E.18 otf2/OTF2_GlobalDefWriter.h File Reference

E.18.2.16 **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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this MetricClass definition.
<i>numberOfMetrics</i>	Number of metrics within the set.
<i>metricMembers</i>	List of metric members. References a MetricMember definition.
<i>metricOccurrence</i>	Defines occurrence of a metric set.
<i>recorderKind</i>	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.18.2.17 **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

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

APPENDIX E. FILE DOCUMENTATION

<i>metricClass</i>	Parent MetricClass definition to which this one is a supplementary definition. References a MetricClass definition.
<i>recorder</i>	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.18.2.18 **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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this MetricClass definition.
<i>metricClass</i>	The instanced MetricClass . This metric class must be of kind OTF2_RECORDER_KIND_ABSTRACT . References a MetricClass definition.
<i>recorder</i>	Recorder of the metric: location ID. References a Location definition.
<i>metricScope</i>	Defines type of scope: location, location group, system tree node, or a generic group of locations.
<i>scope</i>	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.18 otf2/OTF2_GlobalDefWriter.h File Reference

E.18.2.19 `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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this MetricMember definition.
<i>name</i>	Name of the metric. References a String definition.
<i>description</i>	Description of the metric. References a String definition.
<i>metricType</i>	Metric type: PAPI, etc.
<i>metricMode</i>	Metric mode: accumulative, fix, relative, etc.
<i>valueType</i>	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.
<i>metricBase</i>	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.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$, 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.
<i>unit</i>	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.18.2.20 **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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <i>Parameter</i> definition.
<i>name</i>	Name of the parameter (variable name etc.) References a <i>String</i> definition.
<i>parameterType</i>	Type of the parameter, <i>OTF2_ParameterType</i> for possible types.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.18.2.21 **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

E.18 of2/OTF2_GlobalDefWriter.h File Reference

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this Region definition.
<i>name</i>	Name of the region (demangled name if available). References a String definition.
<i>canonicalName</i>	Alternative name of the region (e.g. mangled name). References a String definition. Since version 1.1.
<i>description</i>	A more detailed description of this region. References a String definition.
<i>regionRole</i>	Region role. Since version 1.1.
<i>paradigm</i>	Paradigm. Since version 1.1.
<i>regionFlags</i>	Region flags. Since version 1.1.
<i>sourceFile</i>	The source file where this region was declared. References a String definition.
<i>beginLineNumber</i>	Starting line number of this region in the source file.
<i>endLineNumber</i>	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.18.2.22 **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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this RmaWin definition.
<i>name</i>	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a String definition.
<i>comm</i>	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.18.2.23 **OTF2_ErrorCode** **OTF2_GlobalDefWriter_WriteString** (
 OTF2_GlobalDefWriter * *writerHandle*, **OTF2_StringRef** *self*, **const**
 char * *string*)

Writes a [*String*](#) definition record into the GlobalDefWriter.

The string definitions.

Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <i>String</i> definition.
<i>string</i>	The string, null terminated.

Since

Version 1.0

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.18.2.24 **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

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <i>SystemTreeNode</i> definition.

E.18 otf2/OTF2_GlobalDefWriter.h File Reference

<i>name</i>	Free form instance name of this node. References a String definition.
<i>className</i>	Free form class name of this node References a String definition.
<i>parent</i>	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.18.2.25 `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

<i>writerHandle</i>	The writer handle.
<i>systemTreeNode</i>	Parent SystemTreeNode definition to which this one is a supplementary definition. References a SystemTreeNode definition.
<i>systemTreeDomain</i>	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.18.2.26 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

<i>writerHandle</i>	The writer handle.
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

Since

Version 1.2

Returns

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

E.19 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.

E.19 otf2/OTF2_GlobalEvtReader.h File Reference

- [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.19.1 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.19.2 Function Documentation

E.19.2.1 [OTF2_ErrorCode OTF2_GlobalEvtReader_HasEvent](#) ([OTF2_GlobalEvtReader](#) * reader, int * flag)

Has more events.

Parameters

	<i>reader</i>	Global event reader handle.
out	<i>flag</i>	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.19.2.2 [OTF2_ErrorCode OTF2_GlobalEvtReader_ReadEvent](#) ([OTF2_GlobalEvtReader](#) * reader)

Triggers the callback for the next event record.

Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
---------------	---

APPENDIX E. FILE DOCUMENTATION

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.19.2.3 **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

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.
out	<i>recordsRead</i>	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given <i>recordsToRead</i> if there are no more records left in the trace. In this case the programmer can easily check that the reader has finished his job by checking <i>recordsRead</i> < <i>recordsToRead</i> .

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.19.2.4 **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

<i>reader</i>	Reader object which reads the events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <i>OTF2_GlobalEvtReaderCallbacks_New</i> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.20 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_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](#) measurementMode)
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 numberOfMetrics, 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 sizeReceived)
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 communicator, 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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

- 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.

APPENDIX E. FILE DOCUMENTATION

- 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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

- 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)

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

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 holds 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_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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- [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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- [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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_GlobalEvtReaderCallbacks_SetUnknownCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_Unknown](#) unknownCallback)

Registers the callback for unknown events.

E.20.1 Detailed Description

This defines the callbacks for the global event reader.

Source Template:

templates/OTF2_GlobalEvtReaderCallbacks.tmpl.h

E.20.2 Typedef Documentation

E.20.2.1 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>stopTime</i>	The time the buffer flush finished.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

E.20.2.2 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	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.20.2.3 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	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.20.2.4 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>measure- mentMode</i>	Is the measurement turned on (<i>OTF2_MEASUREMENT_ON</i>) or off (<i>OTF2_MEASUREMENT_OFF</i>)?

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.5 `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

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>metric</i>	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.
<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding MetricMember definitions.
<i>metricValues</i>	List of metric values.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.6 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.7 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20 oftf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

E.20.2.8 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi- cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.9 `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

APPENDIX E. FILE DOCUMENTATION

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the requested receive

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.10 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi- cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

E.20 oftf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.11 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.12 `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).

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communicator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.13 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

Since

Version 1.0

E.20 oftf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.14 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.15 `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

<i>locationID</i>	The location where this event happened.
-------------------	---

APPENDIX E. FILE DOCUMENTATION

<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communicator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.16 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.

E.20 oftf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.17 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>num- berOfRe- quest- edThreads</i>	Requested size of the team.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.18 `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ -
OmpJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void
*userData, OTF2_AttributeList *attributeList)`

Callback for the OmpJoin event record.

APPENDIX E. FILE DOCUMENTATION

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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.19 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.20 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the completed task instance.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.21 `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.

APPENDIX E. FILE DOCUMENTATION

This event record is superseded by the [ThreadTaskCreate](#) event record and should not be used when the [ThreadTaskCreate](#) event record is in use.

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the newly created task instance.

Since

Version 1.0

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.22 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the now active task instance.

Since

Version 1.0

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.23 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	Parameter ID. References a <i>Parameter</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_PARAMETER</i> is available.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.24 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	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.
<i>string</i>	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.20.2.25 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.26 `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 aquired by the process.

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.27 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>type</i>	Type of atomic operation.
<i>bytesSent</i>	Bytes sent to target.
<i>bytesReceived</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.28 `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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.29 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	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.
<i>root</i>	Root process for this operation.
<i>bytesSent</i>	Bytes sent in operation.
<i>bytesReceived</i>	Bytes receives in operation.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.30 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20 oftf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

E.20.2.31 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	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.
<i>group</i>	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.20.2.32 `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

APPENDIX E. FILE DOCUMENTATION

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.33 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

E.20 oftf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.34 `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 local completion of an RMA operation.

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.35 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.36 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes sent to target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.37 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	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.20.2.38 `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.

APPENDIX E. FILE DOCUMENTATION

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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.39 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>remote</i>	Rank of the locked remote process.
<i>syncType</i>	Type of synchronization.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.40 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_RMA_WIN</i> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

APPENDIX E. FILE DOCUMENTATION

E.20.2.41 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.20.2.42 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	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.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.43 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window destructed. References a <i>RmaWin</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_-MAPPING_RMA_WIN</i> is available.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.44 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.20.2.45 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	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.
<i>sequenceCount</i>	A threadContingent unique number. The corresponding ThreadCreate event does have the same number.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.46 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>sequenceCount</i>	A <code>threadContingent</code> unique number. The corresponding <i>Thread-Begin</i> event does have the same number.

Since

Version 1.3

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.47 `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.

APPENDIX E. FILE DOCUMENTATION

Marks the end of a thread.

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	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.
<i>sequence-Count</i>	A <code>threadContingent</code> 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.20.2.48 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>numberOfRequestedThreads</i>	Requested size of the team.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.49 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.50 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.20.2.51 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	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.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.52 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.2.53 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	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.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.54 `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

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	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.20.2.55 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	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](#).

APPENDIX E. FILE DOCUMENTATION

E.20.2.56 `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

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	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.
<i>sequenceCount</i>	A threadContingent unique number. The corresponding Thread-End event does have the same number.

Since

Version 1.3

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.20.2.57 `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ - Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown event record.

Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.20.3 Function Documentation

E.20.3.1 void `OTF2_GlobalEvtReaderCallbacks_Clear` (`OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks`
)

Clears a struct for the global event callbacks.

Parameters

<code>globalEvtReaderCallbacks</code>	Handle to a struct previously allocated with <i>OTF2_GlobalEvtReaderCallbacks_New</i> .
---------------------------------------	---

E.20.3.2 void `OTF2_GlobalEvtReaderCallbacks_Delete` (`OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks`
)

Deallocates a struct for the global event callbacks.

Parameters

<code>globalEvtReaderCallbacks</code>	Handle to a struct previously allocated with <i>OTF2_GlobalEvtReaderCallbacks_New</i> .
---------------------------------------	---

E.20.3.3 `OTF2_GlobalEvtReaderCallbacks*` `OTF2_GlobalEvtReaderCallbacks_New`
(void)

Allocates a new struct for the event callbacks.

Returns

A newly allocated struct of type [*OTF2_GlobalEvtReaderCallbacks*](#).

APPENDIX E. FILE DOCUMENTATION

E.20.3.4 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetBufferFlushCallback
(OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_BufferFlush *bufferFlushCallback*)

Registers the callback for the BufferFlush event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>bufferFlushCallback</i>	Function which should be called for all BufferFlush events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.20.3.5 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetEnterCallback
(OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter events.

Since

Version 1.0

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.6 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_Leave *leaveCallback*)

Registers the callback for the Leave event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>leaveCallback</i>	Function which should be called for all Leave events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.7 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetMeasurementOnOffCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_MeasurementOnOff** *measurementOnOffCallback*)

Registers the callback for the MeasurementOnOff event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
---------------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>measurementOnOff-Callback</i>	Function which should be called for all MeasurementOnOff events.
----------------------------------	--

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.8 OTF2_StatusCode OTF2_GlobalEvtReaderCallbacks_SetMetricCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_Metric *metricCallback*)

Registers the callback for the Metric event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20 oftf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

**E.20.3.9 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-
SetMpiCollectiveBeginCallback (OTF2_GlobalEvtReaderCallbacks
* *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-
MpiCollectiveBegin *mpiCollectiveBeginCallback*
)**

Registers the callback for the MpiCollectiveBegin event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

**E.20.3.10 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-
SetMpiCollectiveEndCallback (OTF2_GlobalEvtReaderCallbacks
* *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-
MpiCollectiveEnd *mpiCollectiveEndCallback*)**

Registers the callback for the MpiCollectiveEnd event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEndCallback</i>	Function which should be called for all MpiCollectiveEnd events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.11 `OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvCallback (OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_MpiIrecv mpiIrecvCallback)`

Registers the callback for the `MpiIrecv` event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all <code>MpiIrecv</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.12 `OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_MpiIrecvRequest mpiIrecvRequestCallback)`

Registers the callback for the `MpiIrecvRequest` event.

Parameters

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all <code>MpiIrecvRequest</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.13 OTF2_StatusCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvCallback
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_MpiIrecv *mpiIrecvCallback*)

Registers the callback for the `MpiIrecv` event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all <code>MpiIrecv</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.14 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetMpiIsendCompleteCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-MpiIsendComplete** *mpiIsendCompleteCallback*)

Registers the callback for the MpiIsendComplete event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIsendCompleteCallback</i>	Function which should be called for all MpiIsendComplete events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.20.3.15 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetMpiRecvCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_MpiRecv** *mpiRecvCallback*)

Registers the callback for the MpiRecv event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecvCallback</i>	Function which should be called for all MpiRecv events.

Since

Version 1.0

E.20 oftf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.16 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetMpiRequestCancelledCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-MpiRequestCancelled** *mpiRequestCancelledCallback*)

Registers the callback for the `MpiRequestCancelled` event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRequestCancelledCallback</i>	Function which should be called for all <code>MpiRequestCancelled</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.17 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetMpiRequestTestCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-MpiRequestTest** *mpiRequestTestCallback*)

Registers the callback for the `MpiRequestTest` event.

Parameters

APPENDIX E. FILE DOCUMENTATION

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRequestTestCallback</i>	Function which should be called for all <code>MpiRequestTest</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.18 **`OTF2_StatusCode OTF2_GlobalEvtReaderCallbacks_SetMpiSendCallback (OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_MpiSend mpiSendCallback)`**

Registers the callback for the `MpiSend` event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSendCallback</i>	Function which should be called for all <code>MpiSend</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

E.20.3.19 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_**
SetOmpAcquireLockCallback (**OTF2_GlobalEvtReaderCallbacks**
***** *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_**
OmpAcquireLock *ompAcquireLockCallback*)

Registers the callback for the OmpAcquireLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompAcquireLockCallback</i>	Function which should be called for all OmpAcquireLock events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.20.3.20 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_**
SetOmpForkCallback (**OTF2_GlobalEvtReaderCallbacks** ***** *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompForkCallback</i>	Function which should be called for all OmpFork events.

Since

Version 1.0

APPENDIX E. FILE DOCUMENTATION

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.21 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetOmpJoinCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_OmpJoin *ompJoinCallback*)

Registers the callback for the OmpJoin event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompJoinCallback</i>	Function which should be called for all OmpJoin events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.22 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetOmpReleaseLockCallback** (**OTF2_GlobalEvtReaderCallbacks**
* *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_OmpReleaseLock** *ompReleaseLockCallback*)

Registers the callback for the OmpReleaseLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompReleaseLockCallback</i>	Function which should be called for all OmpReleaseLock events.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.23 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetOmpTaskCompleteCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-OmpTaskComplete** *ompTaskCompleteCallback*)

Registers the callback for the OmpTaskComplete event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCompleteCallback</i>	Function which should be called for all OmpTaskComplete events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.24 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetOmpTaskCreateCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-OmpTaskCreate** *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate event.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCreateCallback</i>	Function which should be called for all OmpTaskCreate events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.25 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetOmpTaskSwitchCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-OmpTaskSwitch** *ompTaskSwitchCallback*)

Registers the callback for the OmpTaskSwitch event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskSwitchCallback</i>	Function which should be called for all OmpTaskSwitch events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

E.20.3.26 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetParameterIntCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_ParameterInt *parameterIntCallback*
)

Registers the callback for the ParameterInt event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterIntCallback</i>	Function which should be called for all ParameterInt events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.27 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetParameterStringCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_ParameterString** *parameterStringCallback*)

Registers the callback for the ParameterString event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterStringCallback</i>	Function which should be called for all ParameterString events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.28 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_**
SetParameterUnsignedIntCallback (**OTF2_GlobalEvtReaderCallbacks**
*** *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_**
ParameterUnsignedInt *parameterUnsignedIntCallback*
)

Registers the callback for the `ParameterUnsignedInt` event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all <code>ParameterUnsignedInt</code> events.

Since

Version 1.0

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.29 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_**
SetRmaAcquireLockCallback (**OTF2_GlobalEvtReaderCallbacks**
*** *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_**
RmaAcquireLock *rmaAcquireLockCallback*)

Registers the callback for the `RmaAcquireLock` event.

Parameters

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaAcquireLockCallback</i>	Function which should be called for all RmaAcquireLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.30 **OTF2_**`ErrorCode` **OTF2_GlobalEvtReaderCallbacks_SetRmaAtomicCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_RmaAtomic *rmaAtomicCallback*)

Registers the callback for the RmaAtomic event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaAtomicCallback</i>	Function which should be called for all RmaAtomic events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.31 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-**
SetRmaCollectiveBeginCallback (**OTF2_GlobalEvtReaderCallbacks**
*** *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-**
RmaCollectiveBegin *rmaCollectiveBeginCallback*
)

Registers the callback for the RmaCollectiveBegin event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaCollectiveBeginCallback</i>	Function which should be called for all RmaCollectiveBegin events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.32 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-**
SetRmaCollectiveEndCallback (**OTF2_GlobalEvtReaderCallbacks**
*** *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-**
RmaCollectiveEnd *rmaCollectiveEndCallback*)

Registers the callback for the RmaCollectiveEnd event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaCollectiveEndCallback</i>	Function which should be called for all RmaCollectiveEnd events.

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

OTF2_SUCCESS if successful
OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.33 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetRmaGetCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_RmaGet *rmaGetCallback*)

Registers the callback for the RmaGet event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaGetCallback</i>	Function which should be called for all RmaGet events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful
OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.34 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetRmaGroupSyncCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_RmaGroupSync** *rmaGroupSyncCallback*)

Registers the callback for the RmaGroupSync event.

Parameters

APPENDIX E. FILE DOCUMENTATION

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaGroupSyncCallback</i>	Function which should be called for all RmaGroupSync events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.35 OTF2_StatusCode OTF2_GlobalEvtReaderCallbacks_-SetRmaOpCompleteBlockingCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_ RmaOpCompleteBlocking *rmaOpCompleteBlockingCallback*)

Registers the callback for the RmaOpCompleteBlocking event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpCompleteBlockingCallback</i>	Function which should be called for all RmaOpCompleteBlocking events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

**E.20.3.36 OTF2_StatusCode OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpCompleteNonBlockingCallback (OTF2_
GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_RmaOpCompleteNonBlocking
rmaOpCompleteNonBlockingCallback)**

Registers the callback for the RmaOpCompleteNonBlocking event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpCompleteNonBlockingCallback</i>	Function which should be called for all RmaOpCompleteNonBlocking events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful
OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

**E.20.3.37 OTF2_StatusCode OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpCompleteRemoteCallback (OTF2_GlobalEvtReaderCallbacks
* *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_
RmaOpCompleteRemote *rmaOpCompleteRemoteCallback*
)**

Registers the callback for the RmaOpCompleteRemote event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpCompleteRemoteCallback</i>	Function which should be called for all RmaOpCompleteRemote events.

APPENDIX E. FILE DOCUMENTATION

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.38 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetRmaOpTestCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_RmaOpTest *rmaOpTestCallback*)

Registers the callback for the RmaOpTest event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpTestCallback</i>	Function which should be called for all RmaOpTest events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.39 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetRmaPutCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_RmaPut *rmaPutCallback*)

Registers the callback for the RmaPut event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
---------------------------------	---------------------------

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>rmaPut-Callback</i>	Function which should be called for all RmaPut events.
------------------------	--

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.40 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetRmaReleaseLockCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-RmaReleaseLock *rmaReleaseLockCallback*)

Registers the callback for the RmaReleaseLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaReleaseLockCallback</i>	Function which should be called for all RmaReleaseLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX E. FILE DOCUMENTATION

E.20.3.41 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_**
SetRmaRequestLockCallback (**OTF2_GlobalEvtReaderCallbacks**
***** *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_**
RmaRequestLock *rmaRequestLockCallback*)

Registers the callback for the RmaRequestLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaRequestLockCallback</i>	Function which should be called for all RmaRequestLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks*
argument

E.20.3.42 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_**
SetRmaSyncCallback (**OTF2_GlobalEvtReaderCallbacks** ***** *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_RmaSync *rmaSyncCallback*)

Registers the callback for the RmaSync event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaSyncCallback</i>	Function which should be called for all RmaSync events.

Since

Version 1.2

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.43 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetRmaTryLockCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_RmaTryLock *rmaTryLockCallback*)

Registers the callback for the RmaTryLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaTryLockCallback</i>	Function which should be called for all RmaTryLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.44 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetRmaWaitChangeCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_RmaWaitChange** *rmaWaitChangeCallback*)

Registers the callback for the RmaWaitChange event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
---------------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>rmaWait-Change-Callback</i>	Function which should be called for all RmaWaitChange events.
--------------------------------	---

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.45 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetRmaWinCreateCallback (OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-RmaWinCreate *rmaWinCreateCallback*)

Registers the callback for the RmaWinCreate event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWinCreateCallback</i>	Function which should be called for all RmaWinCreate events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

E.20.3.46 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetRmaWinDestroyCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-RmaWinDestroy** *rmaWinDestroyCallback*)

Registers the callback for the RmaWinDestroy event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWinDestroyCallback</i>	Function which should be called for all RmaWinDestroy events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.20.3.47 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetThreadAcquireLockCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-ThreadAcquireLock** *threadAcquireLockCallback*)

Registers the callback for the ThreadAcquireLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadAcquireLockCallback</i>	Function which should be called for all ThreadAcquireLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.48 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetThreadBeginCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_ThreadBegin *threadBeginCallback*)

Registers the callback for the ThreadBegin event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadBeginCallback</i>	Function which should be called for all ThreadBegin events.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.49 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetThreadCreateCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_ThreadCreate *threadCreateCallback*
)

Registers the callback for the ThreadCreate event.

Parameters

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadCreateCallback</i>	Function which should be called for all ThreadCreate events.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.50 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetThreadEndCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_ThreadEnd *threadEndCallback*)

Registers the callback for the ThreadEnd event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadEndCallback</i>	Function which should be called for all ThreadEnd events.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX E. FILE DOCUMENTATION

E.20.3.51 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetThreadForkCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_ThreadFork *threadForkCallback*)

Registers the callback for the ThreadFork event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadForkCallback</i>	Function which should be called for all ThreadFork events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.20.3.52 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetThreadJoinCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_ThreadJoin *threadJoinCallback*)

Registers the callback for the ThreadJoin event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadJoinCallback</i>	Function which should be called for all ThreadJoin events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.53 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetThreadReleaseLockCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-ThreadReleaseLock *threadReleaseLockCallback*)

Registers the callback for the ThreadReleaseLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadReleaseLockCallback</i>	Function which should be called for all ThreadReleaseLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful
OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.54 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetThreadTaskCompleteCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-ThreadTaskComplete *threadTaskCompleteCallback*)

Registers the callback for the ThreadTaskComplete event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
---------------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>thread-TaskCompleteCallback</i>	Function which should be called for all ThreadTaskComplete events.
------------------------------------	--

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.55 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetThreadTaskCreateCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-ThreadTaskCreate *threadTaskCreateCallback*)

Registers the callback for the ThreadTaskCreate event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTaskCreateCallback</i>	Function which should be called for all ThreadTaskCreate events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20 otf2/OTF2_GlobalEvtReaderCallbacks.h File Reference

E.20.3.56 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-**
SetThreadTaskSwitchCallback (**OTF2_GlobalEvtReaderCallbacks**
*** globalEvtReaderCallbacks**, **OTF2_GlobalEvtReaderCallback_-**
ThreadTaskSwitch threadTaskSwitchCallback)

Registers the callback for the ThreadTaskSwitch event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTaskSwitchCallback</i>	Function which should be called for all ThreadTaskSwitch events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid defReaderCallbacks
argument

E.20.3.57 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-**
SetThreadTeamBeginCallback (**OTF2_GlobalEvtReaderCallbacks**
*** globalEvtReaderCallbacks**, **OTF2_GlobalEvtReaderCallback_-**
ThreadTeamBegin threadTeamBeginCallback)

Registers the callback for the ThreadTeamBegin event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTeamBeginCallback</i>	Function which should be called for all ThreadTeamBegin events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.58 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_**
SetThreadTeamEndCallback (**OTF2_GlobalEvtReaderCallbacks**
*** *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_**
ThreadTeamEnd *threadTeamEndCallback*)

Registers the callback for the ThreadTeamEnd event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTeamEndCallback</i>	Function which should be called for all ThreadTeamEnd events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.59 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_**
SetThreadWaitCallback
(**OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*,**
OTF2_GlobalEvtReaderCallback_
ThreadWait *threadWaitCallback*)

Registers the callback for the ThreadWait event.

Parameters

E.21 otf2/OTF2_GlobalSnapReader.h File Reference

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadWait-Callback</i>	Function which should be called for all ThreadWait events.

Since

Version 1.3

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.20.3.60 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetUnknownCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*,
OTF2_GlobalEvtReaderCallback_Unknown *unknownCallback*)

Registers the callback for unknown events.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	Function which should be called for all unknown events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.21 otf2/OTF2_GlobalSnapReader.h File Reference

This is the global snapshot event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
```

APPENDIX E. FILE DOCUMENTATION

```
#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.21.1 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.21.2 Function Documentation

E.21.2.1 [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

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

out	<i>recordsRead</i>	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given recordsToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finished his job by checking <code>recordsRead < recordsToRead</code> .
-----	--------------------	---

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.21.2.2 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

<i>reader</i>	Reader object which reads the snap events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <i>OTF2_GlobalSnapReaderCallbacks_New</i> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

This defines the callbacks for the global snap reader.

APPENDIX E. FILE DOCUMENTATION

```
#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 msgLength, uint64_t requestID)
Callback for the MpiIrecv snap record.

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

- 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 msgLength, 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 msgLength)
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 msgLength)
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)

APPENDIX E. FILE DOCUMENTATION

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 holds all snap record callbacks.

Functions

- void OTF2_GlobalSnapReaderCallbacks_Clear (OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks)

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

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)

APPENDIX E. FILE DOCUMENTATION

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](#) parameterUnsignedIntCallback)

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

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.22.1 Detailed Description

This defines the callbacks for the global snap reader.

Source Template:

templates/OTF2_GlobalSnapReaderCallbacks.tmpl.h

E.22.2 Typedef Documentation

- E.22.2.1** `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.

APPENDIX E. FILE DOCUMENTATION

<i>region</i>	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.22.2.2 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent- Time</i>	The original time this event happended.
<i>measure- mentMode</i>	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.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

E.22.2.3 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>metric</i>	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.
<i>numberOf-Metrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding MetricMember definitions.
<i>metricVal-ues</i>	List of metric values.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.4 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.5 `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 locally but not all of the participating locations completed the operation yet. The corresponding [Mpi-CollectiveBeginSnap](#) record is still in the snapshot though.

Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a Comm definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_COMM is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.6 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .

APPENDIX E. FILE DOCUMENTATION

<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.7 `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 [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.

Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the requested receive

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.8 `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 [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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.22.2.9 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.22.2.10 `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

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.11 `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

APPENDIX E. FILE DOCUMENTATION

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>receiver</i>	MPI rank of receiver in communicator.
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.12 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>lockID</i>	ID of the lock.
<i>acqui-sitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible).
628	Corresponding acquire-release events have same number.

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.22.2.13 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent- Time</i>	The original time this event happended.
<i>num- berOfRe- quest- edThreads</i>	Requested size of the team.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

APPENDIX E. FILE DOCUMENTATION

E.22.2.14 `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 *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the newly created task instance.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.22.2.15 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the now active task instance.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.16 `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 greatest timestamp less or equal the timestamp of this record.

Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.22.2.17 `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 greatest timestamp less or equal the timestamp of this record.

Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>parameter</i>	Parameter ID. References a <i>Parameter</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_-PARAMETER</i> is available.
<i>string</i>	Value: Handle of a string definition References a <i>String</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_STRING</i> is available.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

E.22.2.18 `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 greatest timestamp less or equal the timestamp of this record.

Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.19 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>contRead-Pos</i>	Position to continue reading in the event trace.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.22.2.20 `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

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>numberOfRecord</i>	Number of snapshot event records in this snapshot. Excluding the SnapshotEnd record.

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.22.2.21 `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown snap record.

Parameters

<i>locationID</i>	The location where this snap happened.
<i>snapTime</i>	The time of this snapshot.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.22.2.22 `typedef struct OTF2_GlobalSnapReaderCallbacks_struct OTF2_GlobalSnapReaderCallbacks`

Opaque struct which holds all snap record callbacks.

Since

Version 1.2

E.22.3 Function Documentation

E.22.3.1 void OTF2_GlobalSnapReaderCallbacks_Clear (OTF2_-
GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*
)

Clears a struct for the global snap callbacks.

Parameters

<i>global-SnapReaderCallbacks</i>	Handle to a struct previously allocated with OTF2_-GlobalSnapReaderCallbacks_New .
-----------------------------------	--

Since

Version 1.2

E.22.3.2 void OTF2_GlobalSnapReaderCallbacks_Delete (OTF2_-
GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*
)

Deallocates a struct for the global snap callbacks.

Parameters

<i>global-SnapReaderCallbacks</i>	Handle to a struct previously allocated with OTF2_-GlobalSnapReaderCallbacks_New .
-----------------------------------	--

Since

Version 1.2

E.22.3.3 OTF2_GlobalSnapReaderCallbacks* OTF2_GlobalSnapReaderCallbacks_-
New (void)

Allocates a new struct for the snap callbacks.

Since

Version 1.2

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

Returns

A newly allocated struct of type *OTF2_GlobalSnapReaderCallbacks*.

E.22.3.4 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_SetEnterCallback** (
 OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*,
 OTF2_GlobalSnapReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.22.3.5 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_-SetMeasurementOnOffCallback** (
 OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*,
 OTF2_GlobalSnapReaderCallback_-MeasurementOnOff *measurementOnOffCallback*
)

Registers the callback for the MeasurementOnOff snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>measurementOnOffCallback</i>	Function which should be called for all MeasurementOnOff snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.6 `OTF2_StatusCode OTF2_GlobalSnapReaderCallbacks_SetMetricCallback (OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_Metric metricCallback)`

Registers the callback for the Metric snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.7 `OTF2_StatusCode OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin mpiCollectiveBeginCallback)`

Registers the callback for the MpiCollectiveBegin snap.

Parameters

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid defReaderCallbacks argument

E.22.3.8 OTF2_StatusCode OTF2_GlobalSnapReaderCallbacks_-SetMpiCollectiveEndCallback (OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_ - MpiCollectiveEnd mpiCollectiveEndCallback)

Registers the callback for the MpiCollectiveEnd snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEndCallback</i>	Function which should be called for all MpiCollectiveEnd snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid defReaderCallbacks argument

E.22.3.9 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvCallback**
(OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*,
OTF2_GlobalSnapReaderCallback_MpiIrecv *mpiIrecvCallback*)

Registers the callback for the MpiIrecv snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all MpiIrecv snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.10 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_-SetMpiIrecvRequestCallback**
(OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*,
OTF2_GlobalSnapReaderCallback_-MpiIrecvRequest *mpiIrecvRequestCallback*)

Registers the callback for the MpiIrecvRequest snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all MpiIrecvRequest snaps.

Since

Version 1.2

E.22 oftf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.11 **OTF2_StatusCode** **OTF2_GlobalSnapReaderCallbacks_SetMpiSendCallback**
(**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*,
OTF2_GlobalSnapReaderCallback_MpiSend *mpisendCallback*)

Registers the callback for the MpiSend snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpisend-Callback</i>	Function which should be called for all MpiSend snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.12 **OTF2_StatusCode** **OTF2_GlobalSnapReaderCallbacks_SetMpiSendCompleteCallback** (**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*, **OTF2_GlobalSnapReaderCallback_MpiSendComplete** *mpisendCompleteCallback*)

Registers the callback for the MpiSendComplete snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
-----------------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>mpiIsend-Complete-Callback</i>	Function which should be called for all MpiIsendComplete snaps.
-----------------------------------	---

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.13 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_SetMpiRecvCallback**
(**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*,
OTF2_GlobalSnapReaderCallback_MpiRecv *mpiRecvCallback*)

Registers the callback for the MpiRecv snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecv-Callback</i>	Function which should be called for all MpiRecv snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.14 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_SetMpiSendCallback**
(**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*,
OTF2_GlobalSnapReaderCallback_MpiSend *mpiSendCallback*)

Registers the callback for the MpiSend snap.

E.22 oftf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSend-Callback</i>	Function which should be called for all MpiSend snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid defReaderCallbacks argument

E.22.3.15 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_-SetOmpAcquireLockCallback (OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_ - OmpAcquireLock ompAcquireLockCallback)

Registers the callback for the OmpAcquireLock snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>ompAcquireLock-Callback</i>	Function which should be called for all OmpAcquireLock snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid defReaderCallbacks argument

APPENDIX E. FILE DOCUMENTATION

E.22.3.16 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_SetOmpForkCallback**
(**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*,
OTF2_GlobalSnapReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>ompFork-Callback</i>	Function which should be called for all OmpFork snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.17 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_-SetOmpTaskCreateCallback** (**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*, **OTF2_GlobalSnapReaderCallback_-OmpTaskCreate** *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskCreate-Callback</i>	Function which should be called for all OmpTaskCreate snaps.

Since

Version 1.2

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.18 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_-SetOmpTaskSwitchCallback** (**OTF2_GlobalSnapReaderCallbacks** * **globalSnapReaderCallbacks**, **OTF2_GlobalSnapReaderCallback_-OmpTaskSwitch** **ompTaskSwitchCallback**)

Registers the callback for the OmpTaskSwitch snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskSwitch-Callback</i>	Function which should be called for all OmpTaskSwitch snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.19 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_-SetParameterIntCallback** (**OTF2_GlobalSnapReaderCallbacks** * **globalSnapReaderCallbacks**, **OTF2_GlobalSnapReaderCallback_-ParameterInt** **parameterIntCallback**)

Registers the callback for the ParameterInt snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
-----------------------------------	---------------------------

APPENDIX E. FILE DOCUMENTATION

<i>parameter-IntCallback</i>	Function which should be called for all ParameterInt snaps.
------------------------------	---

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.20 ***OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_-SetParameterStringCallback (OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_ -ParameterString parameterStringCallback)***

Registers the callback for the ParameterString snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>parameter-StringCallback</i>	Function which should be called for all ParameterString snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22 otf2/OTF2_GlobalSnapReaderCallbacks.h File Reference

E.22.3.21 **OTF2_***ErrorCode* **OTF2_GlobalSnapReaderCallbacks_**
SetParameterUnsignedIntCallback (**OTF2_GlobalSnapReaderCallbacks**
***** *globalSnapReaderCallbacks*, **OTF2_GlobalSnapReaderCallback_**
ParameterUnsignedInt *parameterUnsignedIntCallback*
)

Registers the callback for the ParameterUnsignedInt snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid defReaderCallbacks
argument

E.22.3.22 **OTF2_***ErrorCode* **OTF2_GlobalSnapReaderCallbacks_**
SetSnapshotEndCallback (**OTF2_GlobalSnapReaderCallbacks** *****
globalSnapReaderCallbacks, **OTF2_GlobalSnapReaderCallback_**
SnapshotEnd *snapshotEndCallback*)

Registers the callback for the SnapshotEnd snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>snapshotEndCallback</i>	Function which should be called for all SnapshotEnd snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.23 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_**
SetSnapshotStartCallback (**OTF2_GlobalSnapReaderCallbacks**
*** *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_**
SnapshotStart *snapshotStartCallback*)

Registers the callback for the SnapshotStart snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>snapshot-StartCallback</i>	Function which should be called for all SnapshotStart snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.22.3.24 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_SetUnknownCallback**
(**OTF2_GlobalSnapReaderCallbacks * *globalSnapReaderCallbacks*,**
OTF2_GlobalSnapReaderCallback_Unknown *unknownCallback*)

Registers the callback for unknown snaps.

Parameters

E.23 otf2/OTF2_IdMap.h File Reference

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	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.23 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.23.1 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.23.2 Typedef Documentation

E.23.2.1 typedef struct OTF2_IdMap_struct OTF2_IdMap

Opaque data structure representing an ID mapping table.

E.23 otf2/OTF2_IdMap.h File Reference

E.23.2.2 typedef uint8_t OTF2_IdMapMode

Wrapper around enum OTF2_IdMapMode_enum, so that it is guaranteed that it is a uint8_t

E.23.3 Enumeration Type Documentation

E.23.3.1 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.23.4 Function Documentation

E.23.4.1 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

<i>instance</i>	Object to add the mapping to.
<i>localId</i>	Local identifier.
<i>globalId</i>	Global identifier.

Returns

OTF2_SUCCESS, or error code.

E.23.4.2 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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>instance</i>	Object to remove entries from.
-----------------	--------------------------------

Returns

OTF2_SUCCESS, or error code.

E.23.4.3 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

<i>mode</i>	Mapping mode.
<i>capacity</i>	Initial capacity.

Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.

E.23.4.4 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

<i>length</i>	Number of elements in the <i>mappings</i> array.
<i>mappings</i>	Array with a dense mapping.
<i>optimize-Size</i>	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.23 otf2/OTF2_IdMap.h File Reference

E.23.4.5 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-identity 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-identity entries equals zero, ie. the given map is the identity map.

Parameters

<i>length</i>	Number of elements in the <i>mappings</i> array.
<i>mappings</i>	Array with a dense mapping.
<i>optimize-Size</i>	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.23.4.6 void OTF2_IdMap.Free (OTF2_IdMap * *instance*)

Destroys the given *instance* of OTF2_IdMap and releases the allocated memory.

Parameters

<i>instance</i>	Object to be freed
-----------------	--------------------

E.23.4.7 OTF2_ErrorCode OTF2_IdMap.GetGlobalId (const OTF2_IdMap * *instance*, uint64_t *localId*, uint64_t * *globalId*)

Maps the given *localId* to the global id and store it in the starge provide by *globalId*.

If the given *localId* is not in the mapping, sets *globalId* to the *localId*.

Parameters

	<i>instance</i>	Object to add the mapping to.
	<i>localId</i>	Local identifier.
out	<i>globalId</i>	Global identifier.

Returns

OTF2_SUCCESS, or error code.

E.23.4.8 **OTF2_ErrorCode** **OTF2_IdMap_GetGlobalIdSave** (**const** **OTF2_IdMap** * *instance*, **uint64_t** *localId*, **uint64_t** * *globalId*)

Maps the given *localId* to the global id and store it in the storage provided by *globalId*.

If the given *localId* is not in the mapping, returns [*OTF2_ERROR_INDEX_OUT_OF_BOUNDS*](#).

Parameters

	<i>instance</i>	Object to add the mapping to.
	<i>localId</i>	Local identifier.
out	<i>globalId</i>	Global identifier.

Returns

OTF2_SUCCESS, or error code.

E.23.4.9 **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

	<i>instance</i>	Queried object.
out	<i>mode</i>	Identifier mapping mode.

Returns

OTF2_SUCCESS, or error code.

E.23.4.10 **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*.

E.24 otf2/OTF2_Marker.h File Reference

Parameters

	<i>instance</i>	Queried object.
out	<i>size</i>	Number of entries.

Returns

OTF2_SUCCESS, or error code.

E.23.4.11 **OTF2_ErrorCode** OTF2_IdMap_Traverse (const OTF2_IdMap * *instance*, OTF2_IdMap_TraverseCallback *callback*, void * *userData*)

Calls for each mapping pair the callback *callback*.

Parameters

<i>instance</i>	Object to add the mapping to.
<i>callback</i>	Callback function which is called for eaach mapping pair.
<i>userData</i>	Data which is passed to the <i>callback</i> function.

Returns

OTF2_SUCCESS, or error code.

E.24 otf2/OTF2_Marker.h File Reference

This file provides types and enums for markers.

```
#include <stdint.h>
#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.24.1 Detailed Description

This file provides types and enums for markers.

E.24.2 Enumeration Type Documentation

E.24.2.1 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.

E.25 otf2/OTF2_MarkerReader.h File Reference

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.24.2.2 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.25 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)

APPENDIX E. FILE DOCUMENTATION

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.25.1 Detailed Description

This file provides all routines that read marker records.

E.25.2 Function Documentation

E.25.2.1 `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

<i>reader</i>	Reader Object.
<i>recordsToRead</i>	How many records have to be read next.
<i>recordsRead</i>	How many records where read?

Since

Version 1.2

Returns

OTF2_ErrorCode with !=OTF2_SUCCESS if there was an error.

E.25.2.2 `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.

E.26 otf2/OTF2_MarkerReaderCallbacks.h File Reference

Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

<i>reader</i>	This given reader object will be setted up with new callback functions.
<i>callbacks</i>	Struct which holds a function pointer for each record type. OTF2_MarkerReaderCallbacks_New .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.26 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>
```

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.

APPENDIX E. FILE DOCUMENTATION

- 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 holds 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.26.1 Detailed Description

This defines the callbacks for the marker reader.

E.26 otf2/OTF2_MarkerReaderCallbacks.h File Reference

E.26.2 Typedef Documentation

E.26.2.1 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterMarkerCallbacks or OTF2_MarkerReader_SetCallbacks .
<i>self</i>	Reference to this marker definition.
<i>marker-Group</i>	Group name, e.g., "MUST", ...
<i>markerCategory</i>	Marker category, e.g., "Argument type error", ... The tuple (marker-Group, markerCategory) must be unique over all marker definitions.
<i>severity</i>	The severity for this marker category.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.26.2.2 `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

<i>userData</i>	User data as set by OTF2_Reader_RegisterMarkerCallbacks or OTF2_MarkerReader_SetCallbacks .
<i>timestamp</i>	Timestamp of the marker.
<i>duration</i>	Duration the marker applies.
<i>marker</i>	Reference to the marker definition.
<i>scope</i>	The type of scope of this marker instance.
<i>scopeRef</i>	The reference to an element of the scope of this marker. Depends on <i>scope</i> .
<i>text</i>	A textual description for this marker.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.26.2.3 `typedef OTF2_CallbackCode(* OTF2_MarkerReaderCallback_Unknown)(void *userData)`

Function pointer definition for the callback which is triggered for an unknown marker.

Parameters

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterMarkerCallbacks</i> or <i>OTF2_MarkerReader_SetCallbacks</i> .
-----------------	---

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.26.3 Function Documentation

E.26.3.1 `void OTF2_MarkerReaderCallbacks_Clear (OTF2_MarkerReaderCallbacks * markerReaderCallbacks)`

Clears a struct for the marker callbacks.

Since

Version 1.2

Parameters

<i>marker-Reader-Callbacks</i>	Handle to a struct previously allocated with <i>OTF2_MarkerReaderCallbacks_New</i> .
--------------------------------	--

E.26 otf2/OTF2_MarkerReaderCallbacks.h File Reference

E.26.3.2 void OTF2_MarkerReaderCallbacks_Delete (OTF2_MarkerReaderCallbacks
* *markerReaderCallbacks*)

Deallocates a struct for the marker callbacks.

Since

Version 1.2

Parameters

<i>marker-Reader-Callbacks</i>	Handle to a struct previously allocated with OTF2_MarkerReaderCallbacks_New .
--------------------------------	---

E.26.3.3 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.26.3.4 OTF2_ErrorCode OTF2_MarkerReaderCallbacks_SetDefMarkerCallback
(OTF2_MarkerReaderCallbacks * *markerReaderCallbacks*,
OTF2_MarkerReaderCallback_DefMarker *defMarkerCallback*)

Registers the callback for the [DefMarker](#) definition.

Parameters

<i>marker-Reader-Callbacks</i>	Struct for all callbacks.
<i>defMarker-Callback</i>	Function which should be called for all DefMarker definitions.

APPENDIX E. FILE DOCUMENTATION

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.26.3.5 **OTF2_StatusCode** **OTF2_MarkerReaderCallbacks_SetMarkerCallback**
(**OTF2_MarkerReaderCallbacks** * *markerReaderCallbacks*,
OTF2_MarkerReaderCallback_Marker *markerCallback*)

Registers the callback for the *Marker* record.

Parameters

<i>marker-Reader-Callbacks</i>	Struct for all callbacks.
<i>marker-Callback</i>	Function which should be called for all <i>Marker</i> records.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.26.3.6 **OTF2_StatusCode** **OTF2_MarkerReaderCallbacks_SetUnknownCallback**
(**OTF2_MarkerReaderCallbacks** * *markerReaderCallbacks*,
OTF2_MarkerReaderCallback_Unknown *unknownCallback*)

Registers the callback for an unknown marker.

Parameters

<i>marker-Reader-Callbacks</i>	Struct for all callbacks.

E.27 otf2/OTF2_MarkerWriter.h File Reference

<i>unknown-Callback</i>	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.27 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.27.1 Detailed Description

This file provides all routines that write marker records.

E.27.2 Function Documentation

E.27.2.1 **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

<i>writerHandle</i>	Marker writer handle.
<i>self</i>	Reference to this marker definition.
<i>markerGroup</i>	Group name, e.g., "MUST", ...
<i>markerCategory</i>	Marker category, e.g., "Argument type error", ... The tuple (markerGroup, markerCategory) must be unique over all marker definitions.
<i>severity</i>	The severity for this marker category.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.27.2.2 **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

<i>writerHandle</i>	Marker writer handle.
<i>timestamp</i>	Time of the marker.

E.28 otf2/OTF2_MPI_Collectives.h File Reference

<i>duration</i>	A possible duration of this marker. May be 0.
<i>marker</i>	Reference to a marker definition.
<i>scope</i>	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 .
<i>scopeRef</i>	The reference to an element of the scope of this marker. Depends on <i>scope</i> .
<i>text</i>	A textual description for this marker.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.28 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.

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.

E.28 otf2/OTF2_MPI_Collectives.h File Reference

- 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.28.1 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.28.2 Function Documentation

E.28.2.1 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

<i>archive</i>	The archive handle.
<i>global-Comm</i>	The global communicator to use. Will be duplicated.
<i>localComm</i>	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.28.2.2 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

<i>archive</i>	The archive handle.
----------------	---------------------

APPENDIX E. FILE DOCUMENTATION

<i>global-Comm</i>	The global communicator to use. Will be duplicated.
<i>numberOfFiles</i>	Splits the <code>globalComm</code> into <code>numberOfFiles</code> disjoint sub-communicators and evenly distribute the ranks among them.

Returns

Success or error code.

E.28.2.3 `static OTF2_ErrorCode OTF2_MPI_Reader_SetCollectiveCallbacks (`
`OTF2_Reader * reader, MPI_Comm globalComm) [static]`

Register an MPI collective context to an OTF2 reader.

Parameters

<i>reader</i>	The reader handle.
<i>global-Comm</i>	The global communicator to use. Will be duplicated.

Returns

Success or error code.

E.29 otf2/OTF2_Reader.h File Reference

Reading interface for OTF2 archives.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Archive.h>
```

Typedefs

- `typedef struct OTF2_Reader_struct OTF2_Reader`
Keeps all necessary information for the reader.

Functions

- `OTF2_ErrorCode OTF2_Reader_Close (OTF2_Reader *reader)`

E.29 otf2/OTF2_Reader.h File Reference

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_StatusCode](#) [OTF2_Reader_GetCompression](#) ([OTF2_Reader](#) *reader, [OTF2_Compression](#) *compression)

Get compression mode.

- [OTF2_StatusCode](#) [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_StatusCode](#) [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_StatusCode](#) [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_StatusCode](#) [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_StatusCode](#) [OTF2_Reader_GetNumberOfGlobalDefinitions](#) ([OTF2_Reader](#) *reader, [uint64_t](#) *numberOfDefinitions)

Get number of global definitions.

E.29 otf2/OTF2_Reader.h File Reference

- [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)

APPENDIX E. FILE DOCUMENTATION

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)

E.29 otf2/OTF2_Reader.h File Reference

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.

APPENDIX E. FILE DOCUMENTATION

- [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_SetSerialCollectiveCallbacks](#) ([OTF2_Reader](#) *reader)
Convenient function to set the collective callbacks to an serial implementation.

E.29.1 Detailed Description

Reading interface for OTF2 archives.

E.29.2 Function Documentation

E.29.2.1 [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

<i>reader</i>	Reader handle.
---------------	----------------

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.29.2.2 [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

<i>reader</i>	Reader handle.
---------------	----------------

E.29 otf2/OTF2_Reader.h File Reference

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.3 **OTF2_ErrorCode** OTF2_Reader_CloseDefReader (OTF2_Reader * *reader*, OTF2_DefReader * *defReader*)

Close a local definition reader.

Parameters

<i>reader</i>	Valid reader handle.
<i>defReader</i>	Definition reader to be closed.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.4 **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

<i>reader</i>	Reader handle.
---------------	----------------

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.5 OTF2_ErrorCode OTF2_Reader_CloseEvtReader (OTF2_Reader * *reader*, OTF2_EvtReader * *evtReader*)

Close a local event reader.

Parameters

<i>reader</i>	Valid reader handle.
<i>evtReader</i>	Event reader to be closed.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.6 OTF2_ErrorCode OTF2_Reader_CloseGlobalDefReader (OTF2_Reader * *reader*, OTF2_GlobalDefReader * *globalDefReader*)

Closes the global definition reader.

Parameters

<i>reader</i>	Valid reader handle.
<i>globalDef-Reader</i>	The global definition reader.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.7 OTF2_ErrorCode OTF2_Reader_CloseGlobalEvtReader (OTF2_Reader * *reader*, OTF2_GlobalEvtReader * *globalEvtReader*)

Closes the global event reader.

This closes also all local event readers.

Parameters

<i>reader</i>	Valid reader handle.
<i>globalEvtReader</i>	The global event reader.

E.29 otf2/OTF2_Reader.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.8 `OTF2_StatusCode OTF2_Reader_CloseGlobalSnapReader (OTF2_Reader *
reader, OTF2_GlobalSnapReader * globalSnapReader)`

Closes the global snapshot reader.

Parameters

<i>reader</i>	Valid reader handle.
<i>global-SnapReader</i>	The global snapshot reader.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

Since

Version 1.2

E.29.2.9 `OTF2_StatusCode OTF2_Reader_CloseMarkerReader (OTF2_Reader *
reader, OTF2_MarkerReader * markerReader)`

Closes the marker reader.

Parameters

<i>reader</i>	Valid reader handle.
<i>marker-Reader</i>	The marker reader.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.29.2.10 **OTF2_ErrorCode** **OTF2_Reader_CloseMarkerWriter** (**OTF2_Reader** * *reader*, **OTF2_MarkerWriter** * *markerWriter*)

Closes the marker writer.

Parameters

<i>reader</i>	Valid reader handle.
<i>marker-Writer</i>	The marker writer.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.11 **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

<i>reader</i>	Reader handle.
---------------	----------------

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.12 **OTF2_ErrorCode** **OTF2_Reader_CloseSnapReader** (**OTF2_Reader** * *reader*, **OTF2_SnapReader** * *snapReader*)

Close a local snapshot reader.

E.29 otf2/OTF2_Reader.h File Reference

Parameters

<i>reader</i>	Valid reader handle.
<i>snapReader</i>	snapshot reader to be closed.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

Since

Version 1.2

E.29.2.13 **OTF2_ErrorCode** OTF2_Reader_CloseThumbReader (OTF2_Reader * *reader*, OTF2_ThumbReader * *thumbReader*)

Close an opened thumbnail reader.

Parameters

<i>reader</i>	Reader handle.
<i>thumbReader</i>	Thumbn reader handle to be closed.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.14 **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

	<i>reader</i>	Reader handle.
	<i>name</i>	Name of the property.
out	<i>value</i>	Returned boolean value of the property.

APPENDIX E. FILE DOCUMENTATION

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.29.2.15 **OTF2_ErrorCode** **OTF2_Reader_GetChunkSize** (**OTF2_Reader** * *reader*,
uint64_t * *chunkSizeEvents*, uint64_t * *chunkSizeDefinitions*)

Get event and definition chunk sizes.

Parameters

	<i>reader</i>	Reader handle.
out	<i>chunk-SizeEvents</i>	Returned size of event chunks
out	<i>chunk-SizeDefinitions</i>	Returned size of definition chunks.

Returns

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

E.29.2.16 **OTF2_ErrorCode** **OTF2_Reader_GetCompression** (**OTF2_Reader** *
reader, **OTF2_Compression** * *compression*)

Get copression mode.

Parameters

	<i>reader</i>	Reader handle.
out	<i>compression</i>	Returned compression mode.

Returns

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

E.29 otf2/OTF2_Reader.h File Reference

E.29.2.17 `OTF2_ErrorCode OTF2_Reader_GetCreator (OTF2_Reader * reader,
char ** creator)`

Get creator name.

Parameters

	<i>reader</i>	Reader handle.
out	<i>creator</i>	Returned creator. Allocated with <i>malloc</i> .

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.18 `OTF2_DefReader* OTF2_Reader_GetDefReader (OTF2_Reader * reader,
OTF2_LocationRef location)`

Get a local definition reader.

Parameters

	<i>reader</i>	Valid reader handle.
	<i>location</i>	Location ID for the requested local reader.

Returns

Returns a handle to the local definition reader if successful, NULL otherwise.

E.29.2.19 `OTF2_ErrorCode OTF2_Reader_GetDescription (OTF2_Reader * reader,
char ** description)`

Get description.

Parameters

	<i>reader</i>	Reader handle.
out	<i>description</i>	Returned description. Allocated with <i>malloc</i> .

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.29.2.20 **OTF2_EvtReader*** **OTF2_Reader_GetEvtReader** (**OTF2_Reader *** *reader*, **OTF2_LocationRef** *location*)

Get a local event reader.

Parameters

<i>reader</i>	Valid reader handle.
<i>location</i>	Location ID for the requested local reader.

Returns

Returns a handle to the local event reader if successful, NULL otherwise.

E.29.2.21 **OTF2_ErrorCode** **OTF2_Reader_GetFileSubstrate** (**OTF2_Reader *** *reader*, **OTF2_FileSubstrate *** *substrate*)

Get file substrate information.

Parameters

	<i>reader</i>	Reader handle.
out	<i>substrate</i>	Returned file substrate.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.22 **OTF2_GlobalDefReader*** **OTF2_Reader_GetGlobalDefReader** (**OTF2_Reader *** *reader*)

Get a global definition reader.

Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

Returns

Returns a handle to the global definition reader if successful, NULL otherwise.

E.29 otf2/OTF2_Reader.h File Reference

E.29.2.23 OTF2_GlobalEvtReader* OTF2_Reader_GetGlobalEvtReader (OTF2_Reader * *reader*)

Get a global event reader.

Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

Returns

Returns a handle to the global event reader if successful, NULL otherwise.

E.29.2.24 OTF2_GlobalSnapReader* OTF2_Reader_GetGlobalSnapReader (OTF2_Reader * *reader*)

Get a global snap reader.

Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

Returns

Returns a handle to the global snap reader if successful, NULL otherwise.

Since

Version 1.2

E.29.2.25 OTF2_ErrorCode OTF2_Reader_GetMachineName (OTF2_Reader * *reader*, char ** *machineName*)

Get machine name.

Parameters

	<i>reader</i>	Reader handle.
out	<i>machine-Name</i>	Returned machine name. Allocated with <i>malloc</i> .

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

APPENDIX E. FILE DOCUMENTATION

E.29.2.26 **OTF2_MarkerReader*** **OTF2_Reader_GetMarkerReader** (**OTF2_Reader** * *reader*)

Get a marker reader.

Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

Since

Version 1.2

Returns

Returns a handle to the marker reader if successful, NULL otherwise.

E.29.2.27 **OTF2_MarkerWriter*** **OTF2_Reader_GetMarkerWriter** (**OTF2_Reader** * *reader*)

Get a marker writer.

Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

Since

Version 1.2

Returns

Returns a handle to the marker writer if successful, NULL otherwise.

E.29.2.28 **OTF2_ErrorCode** **OTF2_Reader_GetNumberOfGlobalDefinitions** (**OTF2_Reader** * *reader*, **uint64_t** * *numberOfDefinitions*)

Get number of global definitions.

Parameters

	<i>reader</i>	Reader handle.
out	<i>numberOfDefinitions</i>	Returned number of global definitions.

E.29 otf2/OTF2_Reader.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.29 **OTF2_ErrorCode** **OTF2_Reader_GetNumberOfLocations** (**OTF2_Reader**
* *reader*, **uint64_t** * *numberOfLocations*)

Get number of locations.

Parameters

	<i>reader</i>	Reader handle.
out	<i>num- berOfLoca- tions</i>	Returned number of locations.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.30 **OTF2_ErrorCode** **OTF2_Reader_GetNumberOfSnapshots** (**OTF2_Reader**
* *reader*, **uint32_t** * *number*)

Get number of snapshots.

Parameters

	<i>reader</i>	Reader handle.
out	<i>number</i>	Returned number of snapshots.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

Since

Version 1.2

E.29.2.31 **OTF2_ErrorCode** **OTF2_Reader_GetNumberOfThumbnails** (**OTF2_Reader**
* *reader*, **uint32_t** * *number*)

Get number of thumbs.

APPENDIX E. FILE DOCUMENTATION

Parameters

	<i>reader</i>	Reader handle.
out	<i>number</i>	Returned number of thumbs.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

Since

Version 1.2

E.29.2.32 **OTF2_StatusCode** **OTF2_Reader_GetProperty** (**OTF2_Reader** * *reader*,
const char * *name*, char ** *value*)

Get the value of the named trace file property.

Parameters

	<i>reader</i>	Reader handle.
	<i>name</i>	Name of the property.
out	<i>value</i>	Returned value of the property. Allocated with <i>malloc</i> .

Returns

[*OTF2_SUCCESS*](#) if successful

[*OTF2_ERROR_PROPERTY_NOT_FOUND*](#) if the named property was not found

E.29.2.33 **OTF2_StatusCode** **OTF2_Reader_GetPropertyNames** (**OTF2_Reader** *
reader, uint32_t * *numberOfProperties*, char *** *names*)

Get the names of all trace file properties.

Parameters

	<i>reader</i>	Reader handle.
out	<i>numberOfProperties</i>	Returned number of trace file properties.
out	<i>names</i>	Returned list of property names. Allocated with <i>malloc</i> . To release memory, just pass * <i>names</i> to <i>free</i> .

E.29 otf2/OTF2_Reader.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.34 `OTF2_SnapReader* OTF2_Reader_GetSnapReader (OTF2_Reader *
reader, OTF2_LocationRef location)`

Get a local snapshot reader.

Parameters

<i>reader</i>	Valid reader handle.
<i>location</i>	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.29.2.35 `OTF2_ThumbReader* OTF2_Reader_GetThumbReader (OTF2_Reader
* reader, uint32_t number)`

Get a thumb reader.

Parameters

<i>reader</i>	Reader handle.
<i>number</i>	Thumbnail number.

Since

Version 1.2

Returns

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

APPENDIX E. FILE DOCUMENTATION

E.29.2.36 **OTF2_ErrorCode** **OTF2_Reader_GetTraceId** (**OTF2_Reader** * *reader*,
uint64_t * *id*)

Get the identifier of the trace file.

Parameters

	<i>reader</i>	Reader handle.
out	<i>id</i>	Trace identifier.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.37 **OTF2_ErrorCode** **OTF2_Reader_GetVersion** (**OTF2_Reader** * *reader*,
uint8_t * *major*, uint8_t * *minor*, uint8_t * *bugfix*)

Get OTF2 version.

Parameters

	<i>reader</i>	Valid reader handle.
out	<i>major</i>	Major version.
out	<i>minor</i>	Minor version.
out	<i>bugfix</i>	Bugfix revision.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.38 **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

	<i>reader</i>	Global event reader handle.
	<i>evtReader</i>	Global event reader handle.
out	<i>flag</i>	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.

E.29 otf2/OTF2_Reader.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.39 **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

<i>anchor-FilePath</i>	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.29.2.40 **OTF2_ErrorCode** OTF2_Reader_OpenDefFiles (**OTF2_Reader *** *reader*)

Open the local definitions file container.

This function is an collective operation.

Parameters

<i>reader</i>	Reader handle.
---------------	----------------

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.41 **OTF2_ErrorCode** OTF2_Reader_OpenEvtFiles (**OTF2_Reader *** *reader*)

Open the events file container.

This function is an collective operation.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>reader</i>	Reader handle.
---------------	----------------

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.42 **OTF2_StatusCode** OTF2_Reader_OpenSnapFiles (OTF2_Reader * *reader*
)

Open the snapshots file container.

This function is an collective operation.

Parameters

<i>reader</i>	Reader handle.
---------------	----------------

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.43 **OTF2_StatusCode** OTF2_Reader_ReadAllGlobalDefinitions (OTF2_Reader
* *reader*, OTF2_GlobalDefReader * *defReader*, uint64_t * *definitionsRead*)

Read all definitions via a global definition reader.

Parameters

	<i>reader</i>	Reader handle.
	<i>defReader</i>	Global definition reader handle.
out	<i>definitionsRead</i>	Return pointer to the number of definitions actually read.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29 otf2/OTF2_Reader.h File Reference

E.29.2.44 `OTF2_StatusCode OTF2_Reader_ReadAllGlobalEvents (OTF2_Reader *
reader, OTF2_GlobalEvtReader * evtReader, uint64_t * eventsRead)`

Read all events via a global event reader.

Parameters

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Global event reader handle.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.45 `OTF2_StatusCode OTF2_Reader_ReadAllGlobalSnapshots (OTF2_Reader
* reader, OTF2_GlobalSnapReader * snapReader, uint64_t * recordsRead
)`

Read all records via a global snapshot reader.

Parameters

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Global snapshot reader handle.
out	<i>recordsRead</i>	Return pointer to the number of records

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

Since

Version 1.2

E.29.2.46 `OTF2_StatusCode OTF2_Reader_ReadAllLocalDefinitions (OTF2_Reader
* reader, OTF2_DefReader * defReader, uint64_t * definitionsRead)`

Read all definitions via a local definition reader.

Parameters

	<i>reader</i>	Reader handle.
--	---------------	----------------

APPENDIX E. FILE DOCUMENTATION

	<i>defReader</i>	Local definition reader handle.
out	<i>definition- sRead</i>	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.29.2.47 **OTF2_StatusCode** OTF2_Reader_ReadAllLocalEvents (OTF2_Reader * *reader*, OTF2_EvtReader * *evtReader*, uint64_t * *eventsRead*)

Read all events via a local event reader.

Parameters

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Local event reader handle.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

Returns

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

E.29.2.48 **OTF2_StatusCode** OTF2_Reader_ReadAllLocalSnapshots (OTF2_Reader * *reader*, OTF2_SnapReader * *snapReader*, uint64_t * *recordsRead*)

Read all records via a local snapshot reader.

Parameters

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Local snapshot reader handle.
out	<i>recordsRead</i>	Return pointer to the number of records

E.29 otf2/OTF2_Reader.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

Since

Version 1.2

E.29.2.49 **OTF2_ErrorCode** **OTF2_Reader_ReadAllMarkers** (**OTF2_Reader** * *reader*,
OTF2_MarkerReader * *markerReader*, **uint64_t** * *markersRead*)

Read all markers via a marker reader.

Parameters

	<i>reader</i>	Reader handle.
	<i>marker-Reader</i>	Marker reader handle.
out	<i>marker-sRead</i>	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.29.2.50 **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

	<i>reader</i>	Reader handle.
	<i>defReader</i>	Global definition reader handle.
	<i>definition-sToRead</i>	Number definitions to be read.
out	<i>definition-sRead</i>	Return pointer to the number of definitions actually read.

Returns

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

E.29.2.51 `OTF2_StatusCode OTF2_Reader_ReadGlobalEvent (OTF2_Reader *
reader, OTF2_GlobalEvtReader * evtReader)`

Read an event via a global event reader.

Parameters

<i>reader</i>	Reader handle.
<i>evtReader</i>	Global event reader handle.

Returns

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

E.29.2.52 `OTF2_StatusCode 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

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Global event reader handle.
	<i>eventsToRead</i>	Number events to be read.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

Returns

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

E.29.2.53 `OTF2_StatusCode 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.

E.29 otf2/OTF2_Reader.h File Reference

Parameters

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Global snapshot reader handle.
	<i>recordsToRead</i>	Number records to be read.
out	<i>recordsRead</i>	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.29.2.54 **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

	<i>reader</i>	Reader handle.
	<i>defReader</i>	Local definition reader handle.
	<i>definitionsToRead</i>	Number definitions to be read.
out	<i>definitionsRead</i>	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

APPENDIX E. FILE DOCUMENTATION

E.29.2.55 **OTF2_StatusCode** **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

<i>reader</i>	Reader handle.
<i>evtReader</i>	Local event reader handle.
<i>eventsToRead</i>	Number events to be read.
<i>eventsRead</i>	Return pointer to the number of events actually read.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.56 **OTF2_StatusCode** **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

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Local event reader handle.
	<i>eventsToRead</i>	Number events to be read.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.57 **OTF2_StatusCode** **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.

E.29 otf2/OTF2_Reader.h File Reference

Parameters

<i>reader</i>	Reader handle.
<i>snapReader</i>	Local snapshot reader handle.
<i>recordsToRead</i>	Number records to be read.
<i>recordsRead</i>	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.29.2.58 **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

	<i>reader</i>	Reader handle.
	<i>markerReader</i>	Marker reader handle.
	<i>markersToRead</i>	Number markers to be read.
out	<i>markersRead</i>	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.

APPENDIX E. FILE DOCUMENTATION

E.29.2.59 **OTF2_ErrorCode** **OTF2_Reader_RegisterDefCallbacks** (
 OTF2_Reader * *reader*, **OTF2_DefReader** * *defReader*, **const**
 OTF2_DefReaderCallbacks * *callbacks*, **void** * *userData*)

Register local definition reader callbacks.

Parameters

<i>reader</i>	OTF2_Reader handle.
<i>defReader</i>	Local definition reader handle.
<i>callbacks</i>	Callbacks for the local definition readers.
<i>userData</i>	Addition user data.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.60 **OTF2_ErrorCode** **OTF2_Reader_RegisterEvtCallbacks** (
 OTF2_Reader * *reader*, **OTF2_EvtReader** * *evtReader*, **const**
 OTF2_EvtReaderCallbacks * *callbacks*, **void** * *userData*)

Register event reader callbacks.

Parameters

<i>reader</i>	OTF2_Reader handle.
<i>evtReader</i>	Local event reader handle.
<i>callbacks</i>	Callbacks for the event readers.
<i>userData</i>	Addition user data.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.61 **OTF2_ErrorCode** **OTF2_Reader_RegisterGlobalDefCallbacks** (
 OTF2_Reader * *reader*, **OTF2_GlobalDefReader** * *defReader*, **const**
 OTF2_GlobalDefReaderCallbacks * *callbacks*, **void** * *userData*)

Register global definition reader callbacks.

Parameters

<i>reader</i>	OTF2_Reader handle.
---------------	---------------------

E.29 otf2/OTF2_Reader.h File Reference

<i>defReader</i>	Global definition reader handle.
<i>callbacks</i>	Callbacks for the global definition readers.
<i>userData</i>	Addition user data.

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.62 **OTF2_ErrorCode** **OTF2_Reader_RegisterGlobalEvtCallbacks** (
 OTF2_Reader * *reader*, **OTF2_GlobalEvtReader** * *evtReader*, **const**
 OTF2_GlobalEvtReaderCallbacks * *callbacks*, **void** * *userData*)

Register global event reader callbacks.

Parameters

<i>reader</i>	OTF2_Reader handle.
<i>evtReader</i>	Global event reader handle.
<i>callbacks</i>	Callbacks for the global event reader.
<i>userData</i>	Addition user data.

Returns

Returns [*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.63 **OTF2_ErrorCode** **OTF2_Reader_RegisterGlobalSnapCallbacks** (
 OTF2_Reader * *reader*, **OTF2_GlobalSnapReader** * *evtReader*, **const**
 OTF2_GlobalSnapReaderCallbacks * *callbacks*, **void** * *userData*)

Register global event reader callbacks.

Parameters

<i>reader</i>	OTF2_Reader handle.
<i>evtReader</i>	Global event reader handle.
<i>callbacks</i>	Callbacks for the global event reader.
<i>userData</i>	Addition user data.

Since

Version 1.2

Returns

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

E.29.2.64 **OTF2_ErrorCode** **OTF2_Reader_RegisterMarkerCallbacks** (
 OTF2_Reader * *reader*, **OTF2_MarkerReader** * *markerReader*, **const**
 OTF2_MarkerReaderCallbacks * *callbacks*, **void** * *userData*)

Register marker reader callbacks.

Parameters

<i>reader</i>	OTF2_Reader handle.
<i>marker-Reader</i>	Marker reader handle.
<i>callbacks</i>	Callbacks for the marker reader.
<i>userData</i>	Addition user data.

Since

Version 1.2

Returns

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

E.29.2.65 **OTF2_ErrorCode** **OTF2_Reader_RegisterSnapCallbacks** (
 OTF2_Reader * *reader*, **OTF2_SnapReader** * *snapReader*, **const**
 OTF2_SnapReaderCallbacks * *callbacks*, **void** * *userData*)

Register snapshot event reader callbacks.

Parameters

<i>reader</i>	OTF2_Reader handle.
<i>snapReader</i>	Local snap reader handle.
<i>callbacks</i>	Callbacks for the event readers.
<i>userData</i>	Addition user data.

Since

Version 1.2

E.29 otf2/OTF2_Reader.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.66 **OTF2_ErrorCode** **OTF2_Reader_SelectLocation** (**OTF2_Reader** * *reader*,
OTF2_LocationRef *location*)

Select a location to be read.

Parameters

<i>reader</i>	Reader handle.
<i>location</i>	Location ID.

Since

Version 1.3

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.29.2.67 **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

<i>reader</i>	Reader handle.
<i>collective-Callbacks</i>	Struct holding the collective callback functions.
<i>collective-Data</i>	Data passed to the collective callbacks in the <code>userData</code> argument.
<i>global-CommContext</i>	Global communication context.
<i>local-CommContext</i>	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.29.2.68 **OTF2_ErrorCode** **OTF2_Reader_SetSerialCollectiveCallbacks** (**OTF2_Reader * reader**)

Convenient function to set the collective callbacks to an serial implementation.

Parameters

<i>reader</i>	Reader handle.
---------------	----------------

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.30 **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)

E.30 otf2/OTF2_SnapReader.h File Reference

Seek jumps to start of latest snapshot 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.30.1 Detailed Description

This is the local snap reader, which reads snapshot events from one location.

E.30.2 Function Documentation

E.30.2.1 [OTF2_ErrorCode](#) [OTF2_SnapReader_GetLocationID](#) (const [OTF2_SnapReader](#) * reader, [OTF2_LocationRef](#) * location)

Return the location ID of the reading related location.

Parameters

	<i>reader</i>	Reader object which reads the snapshot events from its buffer.
out	<i>location</i>	ID of the location.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.30.2.2 [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

APPENDIX E. FILE DOCUMENTATION

	<i>reader</i>	Reader object which reads the events from its buffer.
	<i>recordsToRead</i>	How many records can be read next.
out	<i>recordsRead</i>	Return how many records were really read.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.30.2.3 **OTF2_ErrorCode OTF2_SnapReader_Seek (OTF2_SnapReader * *reader*,
uint64_t *req_time*, bool * *found*)**

Seek jumps to start of latest snapshot that was made before a given time '*req_time*'.

Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
<i>req_time</i>	Requested time (see above)
<i>found</i>	returns if a matching snapshot was found

Since

Version 1.2

Returns

OTF2_ErrorCode with !=OTF2_SUCCESS if there was an error.

E.30.2.4 **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.

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. OTF2_SnapReaderCallbacks_New .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.31 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)

APPENDIX E. FILE DOCUMENTATION

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 receiver, 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-`

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

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](#) numberOfRequestedThreads)

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)

APPENDIX E. FILE DOCUMENTATION

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 holds all snap event record callbacks.

Functions

- void OTF2_SnapReaderCallbacks_Clear (OTF2_SnapReaderCallbacks *snapReaderCallbacks)

Clears a struct for the snap 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.

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- [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.31.1 Detailed Description

This defines the callbacks for the snap reader.

Source Template:

templates/OTF2_SnapReaderCallbacks.tmpl.h

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

E.31.2 Typedef Documentation

E.31.2.1 `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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_-SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent- Time</i>	The original time this event happened.
<i>region</i>	Needs to be defined in a definition record References a <i>Region</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_REGION</i> is available.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.31.2.2 `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

APPENDIX E. FILE DOCUMENTATION

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>measure-mentMode</i>	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.31.2.3 `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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>metric</i>	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.

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding MetricMember definitions.
<i>metricValues</i>	List of metric values.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.4 `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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEventTime</i>	The original time this event happened.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.5 `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 locally but not all of the participating locations completed the operation yet. The corresponding *Mpi-CollectiveBeginSnap* record is still in the snapshot though.

Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>collec-tiveOp</i>	Determines which collective operation it is.
<i>communi-cator</i>	Communicator References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeRe-ceived</i>	Size of the received message.

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

E.31.2.6 `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 [MpiSendComplete](#) 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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.7 `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 *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.

Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the requested receive

Since

Version 1.2

Returns

OTF2_CALLBACK_SUCCESS or *OTF2_CALLBACK_INTERRUPT*.

E.31.2.8 `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 *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

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

did not occur on the remote location before the snapshot. (This could either be an [MpiRecv](#) or an [MpiIrecv](#) event.)

Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.9 `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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.

APPENDIX E. FILE DOCUMENTATION

<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.10 `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 [MpiSendComplete](#) 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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.31.2.11 `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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>receiver</i>	MPI rank of receiver in <i>communicator</i> .
<i>communi-cator</i>	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_-COMM</i> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

APPENDIX E. FILE DOCUMENTATION

E.31.2.12 `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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEventTime</i>	The original time this event happended.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	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.31.2.13 `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

<i>location</i>	The location where this snap event happened.
-----------------	--

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>num-berOfRe-quest-edThreads</i>	Requested size of the team.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.14 `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 [OmpTaskComplete](#) event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the newly created task instance.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.31.2.15 `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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEventTime</i>	The original time this event happended.
<i>taskID</i>	Identifier of the now active task instance.

Since

Version 1.2

Returns

[*OTF2_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

E.31.2.16 `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.

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.17 `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 greatest timestamp less or equal the timestamp of this record.

Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>parameter</i>	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.

APPENDIX E. FILE DOCUMENTATION

<i>string</i>	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.31.2.18 `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 greatest timestamp less or equal the timestamp of this record.

Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

E.31.2.19 `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

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>contRead-Pos</i>	Position to continue reading in the event trace.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.20 `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.

APPENDIX E. FILE DOCUMENTATION

Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>num-berOfRecord</i>	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.31.2.21 `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

<i>location</i>	The location where this event happened.
<i>time</i>	Snapshot time.
<i>userData</i>	User data as set by OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

Since

Version 1.2

Returns

[OTF2_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

E.31.2.22 `typedef struct OTF2_SnapReaderCallbacks_struct OTF2_SnapReaderCallbacks`

Opaque struct which holds all snap event record callbacks.

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

Since

Version 1.2

E.31.3 Function Documentation

E.31.3.1 void OTF2_SnapReaderCallbacks_Clear (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*)

Clears a struct for the snap event callbacks.

Parameters

<i>snapReaderCallbacks</i>	Handle to a struct previously allocated with OTF2_SnapReaderCallbacks_New .
----------------------------	---

Since

Version 1.2

E.31.3.2 void OTF2_SnapReaderCallbacks_Delete (OTF2_SnapReaderCallbacks * *snapReaderCallbacks*)

Deallocates a struct for the snap event callbacks.

Parameters

<i>snapReaderCallbacks</i>	Handle to a struct previously allocated with OTF2_SnapReaderCallbacks_New .
----------------------------	---

Since

Version 1.2

E.31.3.3 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](#).

APPENDIX E. FILE DOCUMENTATION

E.31.3.4 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetEnterCallback
(OTF2_SnapReaderCallbacks * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.31.3.5 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMeasurementOnOffCallback
(OTF2_SnapReaderCallbacks * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MeasurementOnOff
measurementOnOffCallback)

Registers the callback for the MeasurementOnOff snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>measurementOnOffCallback</i>	Function which should be called for all MeasurementOnOff events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.6 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetMetricCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_Metric *metricCallback*)

Registers the callback for the Metric snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.7 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MpiCollectiveBegin
mpiCollectiveBeginCallback)

Registers the callback for the MpiCollectiveBegin snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.8 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback**
(OTF2_SnapReaderCallbacks * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MpiCollectiveEnd
***mpiCollectiveEndCallback*)**

Registers the callback for the MpiCollectiveEnd snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEndCallback</i>	Function which should be called for all MpiCollectiveEnd events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.9 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetMpiIrecvCallback**
(OTF2_SnapReaderCallbacks * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MpiIrecv *mpilrecvCallback*)

Registers the callback for the MpiIrecv snap event.

Parameters

E.31 oftf2/OTF2_SnapReaderCallbacks.h File Reference

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all MpiIrecv events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.10 OTF2_StatusCode OTF2_SnapReaderCallbacks_SetMpiIrecvRequestCallback
(OTF2_SnapReaderCallbacks * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MpiIrecvRequest
mpiIrecvRequestCallback)

Registers the callback for the MpiIrecvRequest snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all MpiIrecvRequest events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX E. FILE DOCUMENTATION

E.31.3.11 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetMpiIsendCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MpiIsend *mpiIsendCallback*)

Registers the callback for the MpiIsend snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIsendCallback</i>	Function which should be called for all MpiIsend events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

E.31.3.12 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback** (**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*, **OTF2_SnapReaderCallback_MpiIsendComplete** *mpiIsendCompleteCallback*)

Registers the callback for the MpiIsendComplete snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIsendCompleteCallback</i>	Function which should be called for all MpiIsendComplete events.

Since

Version 1.2

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.13 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetMpiRecvCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MpiRecv *mpiRecvCallback*)

Registers the callback for the MpiRecv snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecvCallback</i>	Function which should be called for all MpiRecv events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.14 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetMpiSendCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MpiSend *mpiSendCallback*)

Registers the callback for the MpiSend snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSendCallback</i>	Function which should be called for all MpiSend events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.15 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetOmpAcquireLockCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_OmpAcquireLock
ompAcquireLockCallback)

Registers the callback for the OmpAcquireLock snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompAcquireLockCallback</i>	Function which should be called for all OmpAcquireLock events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.16 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetOmpForkCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork snap event.

Parameters

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompForkCallback</i>	Function which should be called for all OmpFork events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.17 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetOmpTaskCreateCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_OmpTaskCreate *ompTaskCreateCallback*
)

Registers the callback for the OmpTaskCreate snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCreateCallback</i>	Function which should be called for all OmpTaskCreate events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.18 `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpTaskSwitchCallback`
`(OTF2_SnapReaderCallbacks * snapReaderCallbacks,`
`OTF2_SnapReaderCallback_OmpTaskSwitch ompTaskSwitchCallback`
`)`

Registers the callback for the OmpTaskSwitch snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskSwitchCallback</i>	Function which should be called for all OmpTaskSwitch events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.19 `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetParameterIntCallback`
`(OTF2_SnapReaderCallbacks * snapReaderCallbacks,`
`OTF2_SnapReaderCallback_ParameterInt parameterIntCallback)`

Registers the callback for the ParameterInt snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterIntCallback</i>	Function which should be called for all ParameterInt events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

E.31 otf2/OTF2_SnapReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.20 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetParameterStringCallback**
 (**OTF2_SnapReaderCallbacks** * ***snapReaderCallbacks***,
 OTF2_SnapReaderCallback_ParameterString ***parameterStringCallback***
)

Registers the callback for the ParameterString snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterStringCallback</i>	Function which should be called for all ParameterString events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful
OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.21 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetParameterUnsignedIntCallback** (**OTF2_SnapReaderCallbacks**
 * ***snapReaderCallbacks***, **OTF2_SnapReaderCallback_ParameterUnsignedInt** ***parameterUnsignedIntCallback***
)

Registers the callback for the ParameterUnsignedInt snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt events.

APPENDIX E. FILE DOCUMENTATION

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.22 **OTF2_StatusCode** **OTF2_SnapReaderCallbacks_SetSnapshotEndCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_SnapshotEnd *snapshotEndCallback*)

Registers the callback for the SnapshotEnd snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>snapshotEndCallback</i>	Function which should be called for all SnapshotEnd events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.23 **OTF2_StatusCode** **OTF2_SnapReaderCallbacks_SetSnapshotStartCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_SnapshotStart *snapshotStartCallback*)

Registers the callback for the SnapshotStart snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
----------------------------	---------------------------

E.32 otf2/OTF2_SnapWriter.h File Reference

<i>snapshot-StartCallback</i>	Function which should be called for all SnapshotStart events.
-------------------------------	---

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

E.31.3.24 OTF2_ErrorCode OTF2_SnapReaderCallbacks.SetUnknownCallback
(OTF2_SnapReaderCallbacks * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_Unknown *unknownCallback*)

Registers the callback for the Unknown snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>unknownCallback</i>	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.32 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>
```

APPENDIX E. FILE DOCUMENTATION

```
#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.

E.32 otf2/OTF2_SnapWriter.h File Reference

- [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.

APPENDIX E. FILE DOCUMENTATION

- `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.32.1 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.32.2 Typedef Documentation

E.32.2.1 typedef struct OTF2_SnapWriter_struct OTF2_SnapWriter

Keeps all necessary information about the snap writer. See `OTF2_SnapWriter_struct` for detailed information.

E.32 otf2/OTF2_SnapWriter.h File Reference

Since

Version 1.2

E.32.3 Function Documentation

E.32.3.1 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happended.
<i>region</i>	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.32.3.2 OTF2_ErrorCode OTF2_SnapWriter_GetLocationID (const OTF2_SnapWriter * *writer*, OTF2_LocationRef * *locationID*)

Function to get the location ID of a snap writer object.

Parameters

<i>writer</i>	Snap writer object of interest
<i>locationID</i>	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.32.3.3 `OTF2_StatusCode 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happened.
<i>measurementMode</i>	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.32.3.4 `OTF2_StatusCode 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.

E.32 oftf2/OTF2_SnapWriter.h File Reference

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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>metric</i>	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.
<i>numberOf-Metrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding MetricMember definitions.
<i>metricVal-ues</i>	List of metric values.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.5 [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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.6 `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 locally but not all of the participating locations completed the operation yet. The corresponding [Mpi-CollectiveBeginSnap](#) record is still in the snapshot though.

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>collec-tiveOp</i>	Determines which collective operation it is.
<i>communi-cator</i>	Communicator References a Comm definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_COMM is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeRe-ceived</i>	Size of the received message.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32 otf2/OTF2_SnapWriter.h File Reference

E.32.3.7 `OTF2_ErrorCode OTF2_SnapWriter_Mpilrecv (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 [MpiSendComplete](#) 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.8 `OTF2_ErrorCode OTF2_SnapWriter_MpilrecvRequest (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 [MpiIrecv](#) or [MpiRequestCancelled](#) event did not occur on this location before the snapshot. Or the corresponding [MpiIrecv](#) did occurred (the [MpiIrecvSnap](#) record exists

APPENDIX E. FILE DOCUMENTATION

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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>requestID</i>	ID of the requested receive

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.9 `OTF2_StatusCode OTF2_SnapWriter_Mpilsend (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 Mpilsend snapshot record.

This record exists for each [Mpilsend](#) event where an corresponding [MpilsendComplete](#) or [MpiRequestCancelled](#) event did not occur on this location before the snapshot. Or the corresponding [MpilsendComplete](#) did occurred (the [MpilsendCompleteSnap](#) 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi-cator</i>	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.

E.32 otf2/OTF2_SnapWriter.h File Reference

<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.32.3.10 OTF2_ErrorCode OTF2_SnapWriter_MpilsendComplete (
OTF2_SnapWriter * *writer*, OTF2_AttributeList * *attributeList*,
OTF2_TimeStamp *snapTime*, OTF2_TimeStamp *origEventTime*,
uint64_t *requestID*)

Records an MpilsendComplete snapshot record.

This record exists for each [*Mpilsend*](#) event where the corresponding [*MpilsendComplete*](#) 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 [*Mpilrecv*](#) event.) .

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happended.
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.32.3.11 `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 [MpiSendComplete](#) event. Or an [MpiRecvRequest](#) occurred before this event but the corresponding [MpiRecv](#) event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing [MpiRecvRequest](#) is not yet known.

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.12 `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

E.32 otf2/OTF2_SnapWriter.h File Reference

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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>receiver</i>	MPI rank of receiver in communicator.
<i>communi-cator</i>	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.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.13 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>lockID</i>	ID of the lock.

APPENDIX E. FILE DOCUMENTATION

<i>acquisitionOrder</i>	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.32.3.14 `OTF2_StatusCode 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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happended.
<i>numberOfRequestedThreads</i>	Requested size of the team.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.32 otf2/OTF2_SnapWriter.h File Reference

E.32.3.15 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the newly created task instance.

Since

Version 1.2

Returns

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

E.32.3.16 `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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the now active task instance.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.17 **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 greatest timestamp less or equal the timestamp of this record.

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happened.
<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.18 **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.

E.32 otf2/OTF2_SnapWriter.h File Reference

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

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.
<i>parameter</i>	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.
<i>string</i>	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.32.3.19 `OTF2_StatusCode 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 greatest timestamp less or equal the timestamp of this record.

Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.

APPENDIX E. FILE DOCUMENTATION

<i>parameter</i>	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.
<i>value</i>	Value of the recorded parameter.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.32.3.20 **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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>contRead-Pos</i>	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.32.3.21 **OTF2_ErrorCode** **OTF2_SnapWriter_SnapshotStart** (**OTF2_SnapWriter** * **writer**, **OTF2_AttributeList** * **attributeList**, **OTF2_TimeStamp** **snapTime**, **uint64_t** **numberOfRecords**)

Records an SnapshotStart snapshot record.

E.33 otf2/OTF2_Thumbnail.h File Reference

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

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>numberOfRecord</i>	Number of snapshot event records in this snapshot. Excluding the SnapshotEnd record.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful, an error code if an error occurs.

E.33 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.

APPENDIX E. FILE DOCUMENTATION

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.33.1 Detailed Description

This lowest user-visible layer provides write routines to read and write thumbnail data.

E.33.2 Function Documentation

E.33.2.1 [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

	<i>reader</i>	Reader object.
out	<i>name</i>	Name of thumbnail.
out	<i>description</i>	Description of thumbnail.
out	<i>type</i>	Type of thumbnail.
out	<i>numberOfSamples</i>	Number of samples.
out	<i>numberOfMetrics</i>	Number of metrics.
out	<i>refsToDefs</i>	The sorted set of references to definitions used in an thumbnail sample.

E.33 otf2/OTF2_Thumbnail.h File Reference

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.33.2.2 **OTF2_ErrorCode** **OTF2_ThumbReader_ReadSample** (
 OTF2_ThumbReader * *reader*, **uint64_t** * *baseline*, **uint32_t**
 numberOfMetrics, **uint64_t** * *metricSamples*)

Reads a thumbnail sample.

Parameters

	<i>reader</i>	Reader object.
out	<i>baseline</i>	Baseline for this sample. If zero, the baseline is the sum of all metric values in this sample.
	<i>numberOfMetrics</i>	Number of metric sample values.
out	<i>metricSamples</i>	Metric sample values.

Since

Version 1.2

Returns

[*OTF2_SUCCESS*](#) if successful, an error code if an error occurs.

E.33.2.3 **OTF2_ErrorCode** **OTF2_ThumbWriter_WriteSample** (
 OTF2_ThumbWriter * *writer*, **uint64_t** *baseline*, **uint32_t** *numberOfMetrics*,
 const uint64_t * *metricSamples*)

Writes a thumbnail sample.

Parameters

	<i>writer</i>	Writer object.
	<i>baseline</i>	Baseline for this sample. If zero, the baseline is the sum of all metric values in this sample.
	<i>numberOfMetrics</i>	Number of metric sample values.

APPENDIX E. FILE DOCUMENTATION

<i>metricSamples</i>	Metric sample values.
----------------------	-----------------------

Since

Version 1.2

Returns

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

Index

- Controlling OTF2 flush behavior in writing mode, [91](#)
 - [OTF2_PostFlushCallback](#), [92](#)
 - [OTF2_PreFlushCallback](#), [93](#)
- How to use the attribute list for writing additional attributes to event records, [91](#)
- List of all definition records, [22](#)
- List of all event records, [39](#)
- List of all marker records, [72](#)
- List of all snapshot records, [73](#)
- Memory pooling for OTF2, [93](#)
 - [OTF2_MemoryAllocate](#), [94](#)
 - [OTF2_MemoryFreeAll](#), [94](#)
- Operating OTF2 in an collective context, [95](#)
 - [OTF2_Collectives_Barrier](#), [97](#)
 - [OTF2_Collectives_Bcast](#), [98](#)
 - [OTF2_Collectives_CreateLocalComm](#), [98](#)
 - [OTF2_Collectives_FreeLocalComm](#), [98](#)
 - [OTF2_Collectives_Gather](#), [99](#)
 - [OTF2_Collectives_Gatherv](#), [99](#)
 - [OTF2_Collectives_GetRank](#), [99](#)
 - [OTF2_Collectives_GetSize](#), [100](#)
 - [OTF2_Collectives_Release](#), [100](#)
 - [OTF2_Collectives_Scatter](#), [100](#)
 - [OTF2_Collectives_Scatterv](#), [101](#)
- OTF2 callbacks, [91](#)
- OTF2 config tool, [17](#)
- OTF2 estimator tool, [20](#)
- OTF2 marker tool, [19](#)
- OTF2 print tool, [18](#)
- OTF2 records, [21](#)
- OTF2 snapshots tool, [19](#)
- OTF2 usage examples, [86](#)
- [otf2/otf2.h](#), [136](#)
- [otf2/OTF2_Archive.h](#), [136](#)
- [otf2/OTF2_AttributeList.h](#), [167](#)
- [otf2/OTF2_Callbacks.h](#), [192](#)
- [otf2/OTF2_Definitions.h](#), [195](#)
- [otf2/OTF2_DefReader.h](#), [209](#)
- [otf2/OTF2_DefReaderCallbacks.h](#), [212](#)
- [otf2/OTF2_DefWriter.h](#), [250](#)
- [otf2/OTF2_ErrorCodes.h](#), [127](#)
- [otf2/OTF2_Events.h](#), [269](#)
- [otf2/OTF2_EventSizeEstimator.h](#), [275](#)
- [otf2/OTF2_EvtReader.h](#), [310](#)
- [otf2/OTF2_EvtReaderCallbacks.h](#), [315](#)
- [otf2/OTF2_EvtWriter.h](#), [402](#)
- [otf2/OTF2_GeneralDefinitions.h](#), [448](#)
- [otf2/OTF2_GlobalDefReader.h](#), [461](#)
- [otf2/OTF2_GlobalDefReaderCallbacks.h](#), [463](#)
- [otf2/OTF2_GlobalDefWriter.h](#), [502](#)
- [otf2/OTF2_GlobalEvtReader.h](#), [522](#)
- [otf2/OTF2_GlobalEvtReaderCallbacks.h](#), [525](#)
- [otf2/OTF2_GlobalSnapReader.h](#), [611](#)
- [otf2/OTF2_GlobalSnapReaderCallbacks.h](#), [613](#)
- [otf2/OTF2_IdMap.h](#), [649](#)
- [otf2/OTF2_Marker.h](#), [655](#)
- [otf2/OTF2_MarkerReader.h](#), [657](#)
- [otf2/OTF2_MarkerReaderCallbacks.h](#), [659](#)
- [otf2/OTF2_MarkerWriter.h](#), [665](#)
- [otf2/OTF2_MPI_Collectives.h](#), [667](#)
- [otf2/OTF2_Reader.h](#), [670](#)

otf2/OTF2_SnapReader.h, [704](#)
otf2/OTF2_SnapReaderCallbacks.h, [707](#)
otf2/OTF2_SnapWriter.h, [741](#)
otf2/OTF2_Thumbnail.h, [759](#)
OTF2_ABORT
 OTF2_ErrorCodes.h, [132](#)
OTF2_BASE_BINARY
 OTF2_Definitions.h, [203](#)
OTF2_BASE_DECIMAL
 OTF2_Definitions.h, [203](#)
OTF2_CALLBACK_ERROR
 OTF2_GeneralDefinitions.h, [456](#)
OTF2_CALLBACK_INTERRUPT
 OTF2_GeneralDefinitions.h, [456](#)
OTF2_CALLBACK_SUCCESS
 OTF2_GeneralDefinitions.h, [456](#)
OTF2_CART_PERIODIC_FALSE
 OTF2_Definitions.h, [200](#)
OTF2_CART_PERIODIC_TRUE
 OTF2_Definitions.h, [200](#)
OTF2_COLLECTIVE_OP_ALLGATHER
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_ALLGATHERV
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_ALLOCATE
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_ALLREDUCE
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_ALLTOALL
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_ALLTOALLV
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_ALLTOALLW
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_BARRIER
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_BCAST
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_CREATE_HANDLE
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_CREATE_HANDLE_-
 AND_ALLOCATE
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_DEALLOCATE
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_DESTROY_-
 HANDLE
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_DESTROY_-
 HANDLE_AND_DEALLOCATE
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_EXSCAN
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_GATHER
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_GATHERV
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_REDUCE
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_REDUCE_-
 SCATTER
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_REDUCE_-
 SCATTER_BLOCK
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_SCAN
 OTF2_Events.h, [273](#)
OTF2_COLLECTIVE_OP_SCATTER
 OTF2_Events.h, [272](#)
OTF2_COLLECTIVE_OP_SCATTERV
 OTF2_Events.h, [272](#)
OTF2_COMPRESSION_NONE
 OTF2_GeneralDefinitions.h, [456](#)
OTF2_COMPRESSION_UNDEFINED
 OTF2_GeneralDefinitions.h, [456](#)
OTF2_COMPRESSION_ZLIB
 OTF2_GeneralDefinitions.h, [456](#)
OTF2_Definitions.h
 OTF2_BASE_BINARY, [203](#)
 OTF2_BASE_DECIMAL, [203](#)
 OTF2_CART_PERIODIC_FALSE,
 [200](#)
 OTF2_CART_PERIODIC_TRUE, [200](#)
 OTF2_GROUP_FLAG_GLOBAL_-
 MEMBERS, [201](#)
 OTF2_GROUP_FLAG_NONE, [201](#)
 OTF2_GROUP_TYPE_COMM_GROUP,
 [201](#)

INDEX

OTF2_GROUP_TYPE_COMM_LOCATIONS, 201
OTF2_GROUP_TYPE_COMM_SELF, 202
OTF2_GROUP_TYPE_LOCATIONS, 201
OTF2_GROUP_TYPE_METRIC, 201
OTF2_GROUP_TYPE_REGIONS, 201
OTF2_GROUP_TYPE_UNKNOWN, 201
OTF2_LOCATION_GROUP_TYPE_-
PROCESS, 202
OTF2_LOCATION_GROUP_TYPE_-
UNKNOWN, 202
OTF2_LOCATION_TYPE_CPU_THREAD, 202
OTF2_LOCATION_TYPE_GPU, 202
OTF2_LOCATION_TYPE_METRIC, 202
OTF2_LOCATION_TYPE_UNKNOWN, 202
OTF2_METRIC_ABSOLUTE_LAST, 203
OTF2_METRIC_ABSOLUTE_NEXT, 203
OTF2_METRIC_ABSOLUTE_POINT, 203
OTF2_METRIC_ACCUMULATED_-
LAST, 203
OTF2_METRIC_ACCUMULATED_-
NEXT, 203
OTF2_METRIC_ACCUMULATED_-
POINT, 203
OTF2_METRIC_ACCUMULATED_-
START, 203
OTF2_METRIC_ASYNCHRONOUS, 204
OTF2_METRIC_RELATIVE_LAST, 203
OTF2_METRIC_RELATIVE_NEXT, 203
OTF2_METRIC_RELATIVE_POINT, 203
OTF2_METRIC_SYNCHRONOUS, 204
OTF2_METRIC_SYNCHRONOUS_-
STRICT, 204
OTF2_METRIC_TIMING_LAST, 205
OTF2_METRIC_TIMING_MASK, 205
OTF2_METRIC_TIMING_NEXT, 205
OTF2_METRIC_TIMING_POINT, 205
OTF2_METRIC_TIMING_START, 205
OTF2_METRIC_TYPE_OTHER, 205
OTF2_METRIC_TYPE_PAPI, 205
OTF2_METRIC_TYPE_RUSAGE, 205
OTF2_METRIC_TYPE_USER, 205
OTF2_METRIC_VALUE_ABSOLUTE, 206
OTF2_METRIC_VALUE_ACCUMULATED, 206
OTF2_METRIC_VALUE_MASK, 206
OTF2_METRIC_VALUE_RELATIVE, 206
OTF2_PARAMETER_TYPE_INT64, 206
OTF2_PARAMETER_TYPE_STRING, 206
OTF2_PARAMETER_TYPE_UINT64, 206
OTF2_RECORDER_KIND_ABSTRACT, 206
OTF2_RECORDER_KIND_CPU, 206
OTF2_RECORDER_KIND_GPU, 206
OTF2_RECORDER_KIND_UNKNOWN, 206
OTF2_REGION_FLAG_DYNAMIC, 207
OTF2_REGION_FLAG_NONE, 207
OTF2_REGION_FLAG_PHASE, 207
OTF2_REGION_ROLE_ARTIFICIAL, 208
OTF2_REGION_ROLE_ATOMIC, 208

OTF2_REGION_ROLE_BARRIER, 208
 OTF2_REGION_ROLE_CODE, 207
 OTF2_REGION_ROLE_COLL_ALL2ALL, 208
 OTF2_REGION_ROLE_COLL_ALL2ONE, 208
 OTF2_REGION_ROLE_COLL_ONE2ALL, 208
 OTF2_REGION_ROLE_COLL_OTHER, 208
 OTF2_REGION_ROLE_CRITICAL, 208
 OTF2_REGION_ROLE_CRITICAL_-SBLOCK, 208
 OTF2_REGION_ROLE_DATA_TRANSFER, 208
 OTF2_REGION_ROLE_FILE_IO, 208
 OTF2_REGION_ROLE_FLUSH, 208
 OTF2_REGION_ROLE_FUNCTION, 207
 OTF2_REGION_ROLE_IMPLICIT_-BARRIER, 208
 OTF2_REGION_ROLE_LOOP, 207
 OTF2_REGION_ROLE_MASTER, 208
 OTF2_REGION_ROLE_ORDERED, 208
 OTF2_REGION_ROLE_ORDERED_-SBLOCK, 208
 OTF2_REGION_ROLE_PARALLEL, 207
 OTF2_REGION_ROLE_POINT2POINT, 208
 OTF2_REGION_ROLE_RMA, 208
 OTF2_REGION_ROLE_SECTION, 207
 OTF2_REGION_ROLE_SECTIONS, 207
 OTF2_REGION_ROLE_SINGLE, 207
 OTF2_REGION_ROLE_SINGLE_-SBLOCK, 207
 OTF2_REGION_ROLE_TASK, 208
 OTF2_REGION_ROLE_TASK_CREATE, 208
 OTF2_REGION_ROLE_TASK_WAIT, 208
 OTF2_REGION_ROLE_THREAD_-CREATE, 208
 OTF2_REGION_ROLE_THREAD_-WAIT, 209
 OTF2_REGION_ROLE_UNKNOWN, 207
 OTF2_REGION_ROLE_WORKSHARE, 207
 OTF2_REGION_ROLE_WRAPPER, 207
 OTF2_SCOPE_GROUP, 204
 OTF2_SCOPE_LOCATION, 204
 OTF2_SCOPE_LOCATION_GROUP, 204
 OTF2_SCOPE_SYSTEM_TREE_-NODE, 204
 OTF2_SYSTEM_TREE_DOMAIN_-CACHE, 209
 OTF2_SYSTEM_TREE_DOMAIN_-CORE, 209
 OTF2_SYSTEM_TREE_DOMAIN_-MACHINE, 209
 OTF2_SYSTEM_TREE_DOMAIN_-NUMA, 209
 OTF2_SYSTEM_TREE_DOMAIN_-PU, 209
 OTF2_SYSTEM_TREE_DOMAIN_-SHARED_MEMORY, 209
 OTF2_SYSTEM_TREE_DOMAIN_-SOCKET, 209
 OTF2_DEPRECATED
 OTF2_ErrorCodes.h, 131
 OTF2_ERROR_COLLECTIVE_CALLBACK
 OTF2_ErrorCodes.h, 135
 OTF2_ERROR_DUPLICATE_MAPPING_-TABLE
 OTF2_ErrorCodes.h, 135
 OTF2_ERROR_E2BIG
 OTF2_ErrorCodes.h, 132
 OTF2_ERROR_EACCES

INDEX

OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EADDRNOTAVAIL
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EAFNOSUPPORT
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EAGAIN
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EALREADY
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EBADF
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EBADMSG
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EBUSY
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_ECANCELED
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_ECHILD
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_ECONNREFUSED
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_ECONNRESET
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EDEADLK
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EDESTADDRREQ
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EDOM
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EDQUOT
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EEXIST
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EFAULT
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EFBIG
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EINPROGRESS
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EINTR
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EINVAL
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EIO
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EISCONN
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EISDIR
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_ELOOP
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EMFILE
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EMLINK
OTF2_ErrorCodes.h, [132](#)
OTF2_ERROR_EMMSGSIZE
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_EMULTIHOP
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENAMETOOLONG
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_END_OF_BUFFER
OTF2_ErrorCodes.h, [134](#)
OTF2_ERROR_END_OF_FUNCTION
OTF2_ErrorCodes.h, [134](#)
OTF2_ERROR_ENETDOWN
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENETRESET
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENETUNREACH
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENFILE
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENOBUFS
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENODATA
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENODEV
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENOENT
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENOEXEC
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENOLCK
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENOLINK
OTF2_ErrorCodes.h, [133](#)
OTF2_ERROR_ENOMEM

OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOMSG	OTF2_ERROR_ESPIPE
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOPROTOPT	OTF2_ERROR_ESRCH
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOSPC	OTF2_ERROR_ESTALE
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOSR	OTF2_ERROR_ETIME
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOSTR	OTF2_ERROR_ETIMEDOUT
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOSYS	OTF2_ERROR_ETXTBSY
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOTCONN	OTF2_ERROR_EWOULDBLOCK
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOTDIR	OTF2_ERROR_EXDEV
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOTEMPTY	OTF2_ERROR_FILE_CAN_NOT_OPEN
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_ENOTSOK	OTF2_ERROR_FILE_COMPRESSION_-
OTF2_ErrorCodes.h, 133	NOT_SUPPORTED
OTF2_ERROR_ENOTSUP	OTF2_ErrorCodes.h, 135
OTF2_ErrorCodes.h, 133	OTF2_ERROR_FILE_INTERACTION
OTF2_ERROR_ENOTTY	OTF2_ErrorCodes.h, 134
OTF2_ErrorCodes.h, 133	OTF2_ERROR_FILE_SUBSTRATE_NOT_-
OTF2_ERROR_ENXIO	SUPPORTED
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 135
OTF2_ERROR_EOPNOTSUPP	OTF2_ERROR_INDEX_OUT_OF_BOUNDS
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_EOVERFLOW	OTF2_ERROR_INTEGRITY_FAULT
OTF2_ErrorCodes.h, 133	OTF2_ErrorCodes.h, 134
OTF2_ERROR_EPERM	OTF2_ERROR_INTERRUPTED_BY_-
OTF2_ErrorCodes.h, 133	CALLBACK
OTF2_ERROR_EPIPE	OTF2_ErrorCodes.h, 134
OTF2_ErrorCodes.h, 133	OTF2_ERROR_INVALID
OTF2_ERROR_EPROTO	OTF2_ErrorCodes.h, 132
OTF2_ErrorCodes.h, 133	OTF2_ERROR_INVALID_ARGUMENT
OTF2_ERROR_EPROTONOSUPPORT	OTF2_ErrorCodes.h, 134
OTF2_ErrorCodes.h, 133	OTF2_ERROR_INVALID_CALL
OTF2_ERROR_EPROTOTYPE	OTF2_ErrorCodes.h, 134
OTF2_ErrorCodes.h, 134	OTF2_ERROR_INVALID_DATA
OTF2_ERROR_ERANGE	OTF2_ErrorCodes.h, 134
OTF2_ErrorCodes.h, 134	OTF2_ERROR_INVALID_FILE_MODE_-
OTF2_ERROR_EROFS	TRANSITION

INDEX

- OTF2_ErrorCodes.h, [135](#)
- OTF2_ERROR_INVALID_LINENO
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ERROR_INVALID_RECORD
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ERROR_INVALID_SIZE_GIVEN
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ERROR_MEM_ALLOC_FAILED
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ERROR_MEM_FAULT
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ERROR_PROCESSED_WITH_FAULTS
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ERROR_PROPERTY_EXISTS
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ERROR_PROPERTY_NAME_INVALID
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ERROR_PROPERTY_NOT_FOUND
 - OTF2_ErrorCodes.h, [135](#)
- OTF2_ERROR_PROPERTY_VALUE_INVALID
 - OTF2_ErrorCodes.h, [135](#)
- OTF2_ERROR_UNKNOWN_TYPE
 - OTF2_ErrorCodes.h, [134](#)
- OTF2_ErrorCodes.h
 - OTF2_ABORT, [132](#)
 - OTF2_DEPRECATED, [131](#)
 - OTF2_ERROR_COLLECTIVE_CALLBACK, [135](#)
 - OTF2_ERROR_DUPLICATE_MAPPING_TABLE, [135](#)
 - OTF2_ERROR_E2BIG, [132](#)
 - OTF2_ERROR_EACCES, [132](#)
 - OTF2_ERROR_EADDRNOTAVAIL, [132](#)
 - OTF2_ERROR_EAFNOSUPPORT, [132](#)
 - OTF2_ERROR_EAGAIN, [132](#)
 - OTF2_ERROR_EALREADY, [132](#)
 - OTF2_ERROR_EBADF, [132](#)
 - OTF2_ERROR_EBADMSG, [132](#)
 - OTF2_ERROR_EBUSY, [132](#)
 - OTF2_ERROR_ECANCELED, [132](#)
 - OTF2_ERROR_ECHILD, [132](#)
 - OTF2_ERROR_ECONNREFUSED, [132](#)
 - OTF2_ERROR_ECONNRESET, [132](#)
 - OTF2_ERROR_EDEADLK, [132](#)
 - OTF2_ERROR_EDESTADDRREQ, [132](#)
 - OTF2_ERROR_EDOM, [132](#)
 - OTF2_ERROR_EDQUOT, [132](#)
 - OTF2_ERROR_EEXIST, [132](#)
 - OTF2_ERROR_EFAULT, [132](#)
 - OTF2_ERROR_EFBIG, [132](#)
 - OTF2_ERROR_EINPROGRESS, [132](#)
 - OTF2_ERROR_EINTR, [132](#)
 - OTF2_ERROR_EINVAL, [132](#)
 - OTF2_ERROR_EIO, [132](#)
 - OTF2_ERROR_EISCONN, [132](#)
 - OTF2_ERROR_EISDIR, [132](#)
 - OTF2_ERROR_ELOOP, [132](#)
 - OTF2_ERROR_EMFILE, [132](#)
 - OTF2_ERROR_EMLINK, [132](#)
 - OTF2_ERROR EMSGSIZE, [133](#)
 - OTF2_ERROR_EMULTIHOP, [133](#)
 - OTF2_ERROR_ENAMETOOLONG, [133](#)
 - OTF2_ERROR_END_OF_BUFFER, [134](#)
 - OTF2_ERROR_END_OF_FUNCTION, [134](#)
 - OTF2_ERROR_ENETDOWN, [133](#)
 - OTF2_ERROR_ENETRESET, [133](#)
 - OTF2_ERROR_ENETUNREACH, [133](#)
 - OTF2_ERROR_ENFILE, [133](#)
 - OTF2_ERROR_ENOBUFS, [133](#)
 - OTF2_ERROR_ENODATA, [133](#)
 - OTF2_ERROR_ENODEV, [133](#)
 - OTF2_ERROR_ENOENT, [133](#)
 - OTF2_ERROR_ENOEXEC, [133](#)
 - OTF2_ERROR_ENOLCK, [133](#)
 - OTF2_ERROR_ENOLINK, [133](#)
 - OTF2_ERROR_ENOMEM, [133](#)
 - OTF2_ERROR_ENOMSG, [133](#)
 - OTF2_ERROR_ENOPROTOPT, [133](#)

-
- OTF2_ERROR_ENOSPC, [133](#)
 - OTF2_ERROR_ENOSR, [133](#)
 - OTF2_ERROR_ENOSTR, [133](#)
 - OTF2_ERROR_ENOSYS, [133](#)
 - OTF2_ERROR_ENOTCONN, [133](#)
 - OTF2_ERROR_ENOTDIR, [133](#)
 - OTF2_ERROR_ENOTEMPTY, [133](#)
 - OTF2_ERROR_ENOTSOCK, [133](#)
 - OTF2_ERROR_ENOTSUP, [133](#)
 - OTF2_ERROR_ENOTTY, [133](#)
 - OTF2_ERROR_ENXIO, [133](#)
 - OTF2_ERROR_EOPNOTSUPP, [133](#)
 - OTF2_ERROR_EOVERFLOW, [133](#)
 - OTF2_ERROR_EPERM, [133](#)
 - OTF2_ERROR_EPIPE, [133](#)
 - OTF2_ERROR_EPROTO, [133](#)
 - OTF2_ERROR_EPROTONOSUPPORT, [133](#)
 - OTF2_ERROR_EPROTOTYPE, [134](#)
 - OTF2_ERROR_ERANGE, [134](#)
 - OTF2_ERROR_EROFS, [134](#)
 - OTF2_ERROR_ESPIPE, [134](#)
 - OTF2_ERROR_ESRCH, [134](#)
 - OTF2_ERROR_ESTALE, [134](#)
 - OTF2_ERROR_ETIME, [134](#)
 - OTF2_ERROR_ETIMEDOUT, [134](#)
 - OTF2_ERROR_ETXTBSY, [134](#)
 - OTF2_ERROR_EWOULDBLOCK, [134](#)
 - OTF2_ERROR_EXDEV, [134](#)
 - OTF2_ERROR_FILE_CAN_NOT_OPEN, [134](#)
 - OTF2_ERROR_FILE_COMPRESSION_NOT_SUPPORTED, [135](#)
 - OTF2_ERROR_FILE_INTERACTION, [134](#)
 - OTF2_ERROR_FILE_SUBSTRATE_NOT_SUPPORTED, [135](#)
 - OTF2_ERROR_INDEX_OUT_OF_BOUNDS, [134](#)
 - OTF2_ERROR_INTEGRITY_FAULT, [134](#)
 - OTF2_ERROR_INTERRUPTED_BY_CALLBACK, [134](#)
 - OTF2_ERROR_INVALID, [132](#)
 - OTF2_ERROR_INVALID_ARGUMENT, [134](#)
 - OTF2_ERROR_INVALID_CALL, [134](#)
 - OTF2_ERROR_INVALID_DATA, [134](#)
 - OTF2_ERROR_INVALID_FILE_MODE_TRANSITION, [135](#)
 - OTF2_ERROR_INVALID_LINENO, [134](#)
 - OTF2_ERROR_INVALID_RECORD, [134](#)
 - OTF2_ERROR_INVALID_SIZE_GIVEN, [134](#)
 - OTF2_ERROR_MEM_ALLOC_FAILED, [134](#)
 - OTF2_ERROR_MEM_FAULT, [134](#)
 - OTF2_ERROR_PROCESSED_WITH_FAULTS, [134](#)
 - OTF2_ERROR_PROPERTY_EXISTS, [134](#)
 - OTF2_ERROR_PROPERTY_NAME_INVALID, [134](#)
 - OTF2_ERROR_PROPERTY_NOT_FOUND, [135](#)
 - OTF2_ERROR_PROPERTY_VALUE_INVALID, [135](#)
 - OTF2_ERROR_UNKNOWN_TYPE, [134](#)
 - OTF2_SUCCESS, [132](#)
 - OTF2_WARNING, [132](#)
 - OTF2_Events.h
 - OTF2_COLLECTIVE_OP_ALLGATHER, [272](#)
 - OTF2_COLLECTIVE_OP_ALLGATHERV, [272](#)
 - OTF2_COLLECTIVE_OP_ALLOCATE, [273](#)
 - OTF2_COLLECTIVE_OP_ALLREDUCE, [272](#)
 - OTF2_COLLECTIVE_OP_ALLTOALL, [272](#)
 - OTF2_COLLECTIVE_OP_ALLTOALLV, [272](#)
-

INDEX

OTF2_COLLECTIVE_OP_ALLTOALLWOTF2_RMA_ATOMIC_TYPE_FETCH_-
272 AND_ADD, 274

OTF2_COLLECTIVE_OP_BARRIER, OTF2_RMA_ATOMIC_TYPE_FETCH_-
272 AND_INCREMENT, 274

OTF2_COLLECTIVE_OP_BCAST, OTF2_RMA_ATOMIC_TYPE_INCREMENT,
272 274

OTF2_COLLECTIVE_OP_CREATE_- OTF2_RMA_ATOMIC_TYPE_SWAP,
HANDLE, 273 274

OTF2_COLLECTIVE_OP_CREATE_- OTF2_RMA_ATOMIC_TYPE_TEST_-
HANDLE_AND_ALLOCATE, AND_SET, 274
273

OTF2_COLLECTIVE_OP_DEALLOCATE, OTF2_RMA_SYNC_LEVEL_MEMORY,
273 275

OTF2_COLLECTIVE_OP_DESTROY_- OTF2_RMA_SYNC_LEVEL_NONE,
HANDLE, 273 275

OTF2_COLLECTIVE_OP_DESTROY_- OTF2_RMA_SYNC_LEVEL_PROCESS,
HANDLE_AND_DEALLOCATE, 275
273

OTF2_COLLECTIVE_OP_EXSCAN, OTF2_RMA_SYNC_TYPE_MEMORY,
273 275

OTF2_COLLECTIVE_OP_GATHER, OTF2_RMA_SYNC_TYPE_NOTIFY_-
272 OUT, 275

OTF2_COLLECTIVE_OP_GATHEROTF2_FILEMODE_MODIFY
272 OTF2_GeneralDefinitions.h, 456

OTF2_COLLECTIVE_OP_REDUCEOTF2_FILEMODE_READ
273 OTF2_GeneralDefinitions.h, 456

OTF2_COLLECTIVE_OP_REDUCEOTF2_FILEMODE_WRITE
SCATTER, 273 OTF2_GeneralDefinitions.h, 456

OTF2_COLLECTIVE_OP_REDUCEOTF2_FILETYPE_ANCHOR
SCATTER_BLOCK, 273 OTF2_GeneralDefinitions.h, 457

OTF2_COLLECTIVE_OP_SCAN, OTF2_FILETYPE_EVENTS
273 OTF2_GeneralDefinitions.h, 457

OTF2_COLLECTIVE_OP_SCATTEROTF2_FILETYPE_GLOBAL_DEFS
272 OTF2_GeneralDefinitions.h, 457

OTF2_COLLECTIVE_OP_SCATTEROTF2_FILETYPE_LOCAL_DEFS
272 OTF2_GeneralDefinitions.h, 457

OTF2_LOCK_EXCLUSIVE, 273 OTF2_FILETYPE_MARKER
OTF2_LOCK_SHARED, 273 OTF2_GeneralDefinitions.h, 457

OTF2_MEASUREMENT_OFF, 274 OTF2_FILETYPE_SIONRANKMAP
OTF2_MEASUREMENT_ON, 274 OTF2_GeneralDefinitions.h, 457

OTF2_RMA_ATOMIC_TYPE_ACCUMULATEOTF2_FILETYPE_SNAPSHOTS
274 OTF2_GeneralDefinitions.h, 457

OTF2_RMA_ATOMIC_TYPE_COMPAREOTF2_FILETYPE_THUMBNAI
AND_SWAP, 274 OTF2_GeneralDefinitions.h, 457

OTF2_FLUSH
 OTF2_GeneralDefinitions.h, 458
 OTF2_GeneralDefinitions.h
 OTF2_CALLBACK_ERROR, 456
 OTF2_CALLBACK_INTERRUPT, 456
 OTF2_CALLBACK_SUCCESS, 456
 OTF2_COMPRESSION_NONE, 456
 OTF2_COMPRESSION_UNDEFINED, 456
 OTF2_COMPRESSION_ZLIB, 456
 OTF2_FILEMODE_MODIFY, 456
 OTF2_FILEMODE_READ, 456
 OTF2_FILEMODE_WRITE, 456
 OTF2_FILETYPE_ANCHOR, 457
 OTF2_FILETYPE_EVENTS, 457
 OTF2_FILETYPE_GLOBAL_DEFS, 457
 OTF2_FILETYPE_LOCAL_DEFS, 457
 OTF2_FILETYPE_MARKER, 457
 OTF2_FILETYPE_SIONRANKMAP, 457
 OTF2_FILETYPE_SNAPSHOTS, 457
 OTF2_FILETYPE_THUMBNAIL, 457
 OTF2_FLUSH, 458
 OTF2_MAPPING_ATTRIBUTE, 458
 OTF2_MAPPING_COMM, 458
 OTF2_MAPPING_GROUP, 458
 OTF2_MAPPING_LOCATION, 458
 OTF2_MAPPING_MAX, 458
 OTF2_MAPPING_METRIC, 458
 OTF2_MAPPING_PARAMETER, 458
 OTF2_MAPPING_REGION, 458
 OTF2_MAPPING_RMA_WIN, 458
 OTF2_MAPPING_STRING, 458
 OTF2_NO_FLUSH, 458
 OTF2_PARADIGM_COMPILER, 459
 OTF2_PARADIGM_CUDA, 459
 OTF2_PARADIGM_GASPI, 459
 OTF2_PARADIGM_HARDWARE, 459
 OTF2_PARADIGM_HMPP, 459
 OTF2_PARADIGM_MEASUREMENT_SYSTEM, 459
 OTF2_PARADIGM_MPI, 459
 OTF2_PARADIGM_OMPSS, 459
 OTF2_PARADIGM_OPENMP, 459
 OTF2_PARADIGM_PTHREAD, 459
 OTF2_PARADIGM_SHMEM, 459
 OTF2_PARADIGM_UNKNOWN, 459
 OTF2_PARADIGM_UPC, 459
 OTF2_PARADIGM_USER, 459
 OTF2_SUBSTRATE_NONE, 457
 OTF2_SUBSTRATE_POSIX, 457
 OTF2_SUBSTRATE_SION, 457
 OTF2_SUBSTRATE_UNDEFINED, 457
 OTF2_THUMBNAIL_TYPE_ATTRIBUTES, 460
 OTF2_THUMBNAIL_TYPE_METRIC, 460
 OTF2_THUMBNAIL_TYPE_REGION, 460
 OTF2_TYPE_ATTRIBUTE, 461
 OTF2_TYPE_COMM, 461
 OTF2_TYPE_DOUBLE, 460
 OTF2_TYPE_FLOAT, 460
 OTF2_TYPE_GROUP, 461
 OTF2_TYPE_INT16, 460
 OTF2_TYPE_INT32, 460
 OTF2_TYPE_INT64, 460
 OTF2_TYPE_INT8, 460
 OTF2_TYPE_LOCATION, 461
 OTF2_TYPE_METRIC, 461
 OTF2_TYPE_NONE, 460
 OTF2_TYPE_PARAMETER, 461
 OTF2_TYPE_REGION, 461
 OTF2_TYPE_RMA_WIN, 461
 OTF2_TYPE_STRING, 460
 OTF2_TYPE_UINT16, 460
 OTF2_TYPE_UINT32, 460
 OTF2_TYPE_UINT64, 460
 OTF2_TYPE_UINT8, 460
 OTF2_GROUP_FLAG_GLOBAL_MEMBERS
 OTF2_Definitions.h, 201
 OTF2_GROUP_FLAG_NONE

INDEX

OTF2_Definitions.h, [201](#)
OTF2_GROUP_TYPE_COMM_GROUP
OTF2_Definitions.h, [201](#)
OTF2_GROUP_TYPE_COMM_LOCATIONS
OTF2_Definitions.h, [201](#)
OTF2_GROUP_TYPE_COMM_SELF
OTF2_Definitions.h, [202](#)
OTF2_GROUP_TYPE_LOCATIONS
OTF2_Definitions.h, [201](#)
OTF2_GROUP_TYPE_METRIC
OTF2_Definitions.h, [201](#)
OTF2_GROUP_TYPE_REGIONS
OTF2_Definitions.h, [201](#)
OTF2_GROUP_TYPE_UNKNOWN
OTF2_Definitions.h, [201](#)
OTF2_ID_MAP_DENSE
OTF2_IdMap.h, [651](#)
OTF2_ID_MAP_SPARSE
OTF2_IdMap.h, [651](#)
OTF2_IdMap.h
OTF2_ID_MAP_DENSE, [651](#)
OTF2_ID_MAP_SPARSE, [651](#)
OTF2_LOCATION_GROUP_TYPE_PROCESS
OTF2_Definitions.h, [202](#)
OTF2_LOCATION_GROUP_TYPE_UNKNOWN
OTF2_Definitions.h, [202](#)
OTF2_LOCATION_TYPE_CPU_THREAD
OTF2_Definitions.h, [202](#)
OTF2_LOCATION_TYPE_GPU
OTF2_Definitions.h, [202](#)
OTF2_LOCATION_TYPE_METRIC
OTF2_Definitions.h, [202](#)
OTF2_LOCATION_TYPE_UNKNOWN
OTF2_Definitions.h, [202](#)
OTF2_LOCK_EXCLUSIVE
OTF2_Events.h, [273](#)
OTF2_LOCK_SHARED
OTF2_Events.h, [273](#)
OTF2_MAPPING_ATTRIBUTE
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_COMM
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_GROUP
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_LOCATION
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_MAX
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_METRIC
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_PARAMETER
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_REGION
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_RMA_WIN
OTF2_GeneralDefinitions.h, [458](#)
OTF2_MAPPING_STRING
OTF2_GeneralDefinitions.h, [458](#)
OTF2_Marker.h
OTF2_MARKER_SCOPE_COMM,
[657](#)
OTF2_MARKER_SCOPE_GLOBAL,
[656](#)
OTF2_MARKER_SCOPE_GROUP,
[657](#)
OTF2_MARKER_SCOPE_LOCATION,
[656](#)
OTF2_MARKER_SCOPE_LOCATION_-
GROUP, [656](#)
OTF2_MARKER_SCOPE_SYSTEM_-
TREE_NODE, [656](#)
OTF2_SEVERITY_HIGH, [657](#)
OTF2_SEVERITY_LOW, [657](#)
OTF2_SEVERITY_MEDIUM, [657](#)
OTF2_SEVERITY_NONE, [657](#)
OTF2_MARKER_SCOPE_COMM
OTF2_Marker.h, [657](#)
OTF2_MARKER_SCOPE_GLOBAL
OTF2_Marker.h, [656](#)
OTF2_MARKER_SCOPE_GROUP
OTF2_Marker.h, [657](#)
OTF2_MARKER_SCOPE_LOCATION
OTF2_Marker.h, [656](#)
OTF2_MARKER_SCOPE_LOCATION_-
GROUP
OTF2_Marker.h, [656](#)
OTF2_MARKER_SCOPE_SYSTEM_-
TREE_NODE

OTF2_Marker.h, 656	OTF2_Definitions.h, 205
OTF2_MEASUREMENT_OFF	OTF2_METRIC_TYPE_RUSAGE
OTF2_Events.h, 274	OTF2_Definitions.h, 205
OTF2_MEASUREMENT_ON	OTF2_METRIC_TYPE_USER
OTF2_Events.h, 274	OTF2_Definitions.h, 205
OTF2_METRIC_ABSOLUTE_LAST	OTF2_METRIC_VALUE_ABSOLUTE
OTF2_Definitions.h, 203	OTF2_Definitions.h, 206
OTF2_METRIC_ABSOLUTE_NEXT	OTF2_METRIC_VALUE_ACCUMULATED
OTF2_Definitions.h, 203	OTF2_Definitions.h, 206
OTF2_METRIC_ABSOLUTE_POINT	OTF2_METRIC_VALUE_MASK
OTF2_Definitions.h, 203	OTF2_Definitions.h, 206
OTF2_METRIC_ACCUMULATED_LAST	OTF2_METRIC_VALUE_RELATIVE
OTF2_Definitions.h, 203	OTF2_Definitions.h, 206
OTF2_METRIC_ACCUMULATED_NEXT	OTF2_NO_FLUSH
OTF2_Definitions.h, 203	OTF2_GeneralDefinitions.h, 458
OTF2_METRIC_ACCUMULATED_POINT	OTF2_PARADIGM_COMPILER
OTF2_Definitions.h, 203	OTF2_GeneralDefinitions.h, 459
OTF2_METRIC_ACCUMULATED_START	OTF2_PARADIGM_CUDA
OTF2_Definitions.h, 203	OTF2_GeneralDefinitions.h, 459
OTF2_METRIC_ASYNCHRONOUS	OTF2_PARADIGM_GASPI
OTF2_Definitions.h, 204	OTF2_GeneralDefinitions.h, 459
OTF2_METRIC_RELATIVE_LAST	OTF2_PARADIGM_HARDWARE
OTF2_Definitions.h, 203	OTF2_GeneralDefinitions.h, 459
OTF2_METRIC_RELATIVE_NEXT	OTF2_PARADIGM_HMPP
OTF2_Definitions.h, 203	OTF2_GeneralDefinitions.h, 459
OTF2_METRIC_RELATIVE_POINT	OTF2_PARADIGM_MEASUREMENT_ -
OTF2_Definitions.h, 203	SYSTEM
OTF2_METRIC_SYNCHRONOUS	OTF2_GeneralDefinitions.h, 459
OTF2_Definitions.h, 204	OTF2_PARADIGM_MPI
OTF2_METRIC_SYNCHRONOUS_STRICT	OTF2_GeneralDefinitions.h, 459
OTF2_Definitions.h, 204	OTF2_PARADIGM_OMPSS
OTF2_METRIC_TIMING_LAST	OTF2_GeneralDefinitions.h, 459
OTF2_Definitions.h, 205	OTF2_PARADIGM_OPENMP
OTF2_METRIC_TIMING_MASK	OTF2_GeneralDefinitions.h, 459
OTF2_Definitions.h, 205	OTF2_PARADIGM_PTHREAD
OTF2_METRIC_TIMING_NEXT	OTF2_GeneralDefinitions.h, 459
OTF2_Definitions.h, 205	OTF2_PARADIGM_SHMEM
OTF2_METRIC_TIMING_POINT	OTF2_GeneralDefinitions.h, 459
OTF2_Definitions.h, 205	OTF2_PARADIGM_UNKNOWN
OTF2_METRIC_TIMING_START	OTF2_GeneralDefinitions.h, 459
OTF2_Definitions.h, 205	OTF2_PARADIGM_UPC
OTF2_METRIC_TYPE_OTHER	OTF2_GeneralDefinitions.h, 459
OTF2_Definitions.h, 205	OTF2_PARADIGM_USER
OTF2_METRIC_TYPE_PAPI	OTF2_GeneralDefinitions.h, 459

INDEX

OTF2_PARAMETER_TYPE_INT64	OTF2_REGION_ROLE_FLUSH
OTF2_Definitions.h, 206	OTF2_Definitions.h, 208
OTF2_PARAMETER_TYPE_STRING	OTF2_REGION_ROLE_FUNCTION
OTF2_Definitions.h, 206	OTF2_Definitions.h, 207
OTF2_PARAMETER_TYPE_UINT64	OTF2_REGION_ROLE_IMPLICIT_BARRIER
OTF2_Definitions.h, 206	OTF2_Definitions.h, 208
OTF2_RECORDER_KIND_ABSTRACT	OTF2_REGION_ROLE_LOOP
OTF2_Definitions.h, 206	OTF2_Definitions.h, 207
OTF2_RECORDER_KIND_CPU	OTF2_REGION_ROLE_MASTER
OTF2_Definitions.h, 206	OTF2_Definitions.h, 208
OTF2_RECORDER_KIND_GPU	OTF2_REGION_ROLE_ORDERED
OTF2_Definitions.h, 206	OTF2_Definitions.h, 208
OTF2_RECORDER_KIND_UNKNOWN	OTF2_REGION_ROLE_ORDERED_SBLOCK
OTF2_Definitions.h, 206	OTF2_Definitions.h, 208
OTF2_REGION_FLAG_DYNAMIC	OTF2_REGION_ROLE_PARALLEL
OTF2_Definitions.h, 207	OTF2_Definitions.h, 207
OTF2_REGION_FLAG_NONE	OTF2_REGION_ROLE_POINT2POINT
OTF2_Definitions.h, 207	OTF2_Definitions.h, 208
OTF2_REGION_FLAG_PHASE	OTF2_REGION_ROLE_RMA
OTF2_Definitions.h, 207	OTF2_Definitions.h, 208
OTF2_REGION_ROLE_ARTIFICIAL	OTF2_REGION_ROLE_SECTION
OTF2_Definitions.h, 208	OTF2_Definitions.h, 207
OTF2_REGION_ROLE_ATOMIC	OTF2_REGION_ROLE_SECTIONS
OTF2_Definitions.h, 208	OTF2_Definitions.h, 207
OTF2_REGION_ROLE_BARRIER	OTF2_REGION_ROLE_SINGLE
OTF2_Definitions.h, 208	OTF2_Definitions.h, 207
OTF2_REGION_ROLE_CODE	OTF2_REGION_ROLE_SINGLE_SBLOCK
OTF2_Definitions.h, 207	OTF2_Definitions.h, 207
OTF2_REGION_ROLE_COLL_ALL2ALL	OTF2_REGION_ROLE_TASK
OTF2_Definitions.h, 208	OTF2_Definitions.h, 208
OTF2_REGION_ROLE_COLL_ALL2ONE	OTF2_REGION_ROLE_TASK_CREATE
OTF2_Definitions.h, 208	OTF2_Definitions.h, 208
OTF2_REGION_ROLE_COLL_ONE2ALL	OTF2_REGION_ROLE_TASK_WAIT
OTF2_Definitions.h, 208	OTF2_Definitions.h, 208
OTF2_REGION_ROLE_COLL_OTHER	OTF2_REGION_ROLE_THREAD_CREATE
OTF2_Definitions.h, 208	OTF2_Definitions.h, 208
OTF2_REGION_ROLE_CRITICAL	OTF2_REGION_ROLE_THREAD_WAIT
OTF2_Definitions.h, 208	OTF2_Definitions.h, 209
OTF2_REGION_ROLE_CRITICAL_SBLOCK	OTF2_REGION_ROLE_UNKNOWN
OTF2_Definitions.h, 208	OTF2_Definitions.h, 207
OTF2_REGION_ROLE_DATA_TRANSFER	OTF2_REGION_ROLE_WORKSHARE
OTF2_Definitions.h, 208	OTF2_Definitions.h, 207
OTF2_REGION_ROLE_FILE_IO	OTF2_REGION_ROLE_WRAPPER
OTF2_Definitions.h, 208	OTF2_Definitions.h, 207

OTF2_RMA_ATOMIC_TYPE_ACCUMULATE_OTF2_SEVERITY_MEDIUM
 OTF2_Events.h, [274](#) OTF2_Marker.h, [657](#)
 OTF2_RMA_ATOMIC_TYPE_COMPARE_OTF2_SEVERITY_NONE
 AND_SWAP OTF2_Marker.h, [657](#)
 OTF2_Events.h, [274](#) OTF2_SUBSTRATE_NONE
 OTF2_RMA_ATOMIC_TYPE_FETCH_- OTF2_GeneralDefinitions.h, [457](#)
 AND_ADD OTF2_SUBSTRATE_POSIX
 OTF2_Events.h, [274](#) OTF2_GeneralDefinitions.h, [457](#)
 OTF2_RMA_ATOMIC_TYPE_FETCH_-OTF2_SUBSTRATE_SION
 AND_INCREMENT OTF2_GeneralDefinitions.h, [457](#)
 OTF2_Events.h, [274](#) OTF2_SUBSTRATE_UNDEFINED
 OTF2_RMA_ATOMIC_TYPE_INCREMENTOTF2_GeneralDefinitions.h, [457](#)
 OTF2_Events.h, [274](#) OTF2_SUCCESS
 OTF2_RMA_ATOMIC_TYPE_SWAP OTF2_ErrorCodes.h, [132](#)
 OTF2_Events.h, [274](#) OTF2_SYSTEM_TREE_DOMAIN_CACHE
 OTF2_RMA_ATOMIC_TYPE_TEST_- OTF2_Definitions.h, [209](#)
 AND_SET OTF2_SYSTEM_TREE_DOMAIN_CORE
 OTF2_Events.h, [274](#) OTF2_Definitions.h, [209](#)
 OTF2_RMA_SYNC_LEVEL_MEMORYOTF2_SYSTEM_TREE_DOMAIN_MACHINE
 OTF2_Events.h, [275](#) OTF2_Definitions.h, [209](#)
 OTF2_RMA_SYNC_LEVEL_NONE OTF2_SYSTEM_TREE_DOMAIN_NUMA
 OTF2_Events.h, [275](#) OTF2_Definitions.h, [209](#)
 OTF2_RMA_SYNC_LEVEL_PROCESS OTF2_SYSTEM_TREE_DOMAIN_PU
 OTF2_Events.h, [275](#) OTF2_Definitions.h, [209](#)
 OTF2_RMA_SYNC_TYPE_MEMORY OTF2_SYSTEM_TREE_DOMAIN_SHARED_-
 OTF2_Events.h, [275](#) MEMORY
 OTF2_RMA_SYNC_TYPE_NOTIFY_- OTF2_Definitions.h, [209](#)
 IN OTF2_SYSTEM_TREE_DOMAIN_SOCKET
 OTF2_Events.h, [275](#) OTF2_Definitions.h, [209](#)
 OTF2_RMA_SYNC_TYPE_NOTIFY_- OTF2_THUMBNAIL_TYPE_ATTRIBUTES
 OUT OTF2_GeneralDefinitions.h, [460](#)
 OTF2_Events.h, [275](#) OTF2_THUMBNAIL_TYPE_METRIC
 OTF2_SCOPE_GROUP OTF2_GeneralDefinitions.h, [460](#)
 OTF2_Definitions.h, [204](#) OTF2_THUMBNAIL_TYPE_REGION
 OTF2_SCOPE_LOCATION OTF2_GeneralDefinitions.h, [460](#)
 OTF2_Definitions.h, [204](#) OTF2_TYPE_ATTRIBUTE
 OTF2_SCOPE_LOCATION_GROUP OTF2_GeneralDefinitions.h, [461](#)
 OTF2_Definitions.h, [204](#) OTF2_TYPE_COMM
 OTF2_SCOPE_SYSTEM_TREE_NODE OTF2_GeneralDefinitions.h, [461](#)
 OTF2_Definitions.h, [204](#) OTF2_TYPE_DOUBLE
 OTF2_SEVERITY_HIGH OTF2_GeneralDefinitions.h, [460](#)
 OTF2_Marker.h, [657](#) OTF2_TYPE_FLOAT
 OTF2_SEVERITY_LOW OTF2_GeneralDefinitions.h, [460](#)
 OTF2_Marker.h, [657](#) OTF2_TYPE_GROUP

INDEX

OTF2_GeneralDefinitions.h, [461](#)
OTF2_TYPE_INT16
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_INT32
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_INT64
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_INT8
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_LOCATION
 OTF2_GeneralDefinitions.h, [461](#)
OTF2_TYPE_METRIC
 OTF2_GeneralDefinitions.h, [461](#)
OTF2_TYPE_NONE
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_PARAMETER
 OTF2_GeneralDefinitions.h, [461](#)
OTF2_TYPE_REGION
 OTF2_GeneralDefinitions.h, [461](#)
OTF2_TYPE_RMA_WIN
 OTF2_GeneralDefinitions.h, [461](#)
OTF2_TYPE_STRING
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_UINT16
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_UINT32
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_UINT64
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_TYPE_UINT8
 OTF2_GeneralDefinitions.h, [460](#)
OTF2_WARNING
 OTF2_ErrorCodes.h, [132](#)
OTF2_Archive
 OTF2_Archive.h, [142](#)
OTF2_Archive.h
 OTF2_Archive, [142](#)
 OTF2_Archive_Close, [143](#)
 OTF2_Archive_CloseDefFiles, [143](#)
 OTF2_Archive_CloseDefReader, [143](#)
 OTF2_Archive_CloseDefWriter, [144](#)
 OTF2_Archive_CloseEvtFiles, [144](#)
 OTF2_Archive_CloseEvtReader, [144](#)
 OTF2_Archive_CloseEvtWriter, [145](#)
 OTF2_Archive_CloseGlobalDefReader, [145](#)
 OTF2_Archive_CloseGlobalDefWriter, [145](#)
 OTF2_Archive_CloseGlobalEvtReader, [146](#)
 OTF2_Archive_CloseGlobalSnapReader, [146](#)
 OTF2_Archive_CloseMarkerReader, [146](#)
 OTF2_Archive_CloseMarkerWriter, [147](#)
 OTF2_Archive_CloseSnapFiles, [147](#)
 OTF2_Archive_CloseSnapReader, [148](#)
 OTF2_Archive_CloseSnapWriter, [148](#)
 OTF2_Archive_CloseThumbReader, [149](#)
 OTF2_Archive_GetChunkSize, [149](#)
 OTF2_Archive_GetCompression, [149](#)
 OTF2_Archive_GetCreator, [150](#)
 OTF2_Archive_GetDefReader, [150](#)
 OTF2_Archive_GetDefWriter, [150](#)
 OTF2_Archive_GetDescription, [151](#)
 OTF2_Archive_GetEvtReader, [151](#)
 OTF2_Archive_GetEvtWriter, [151](#)
 OTF2_Archive_GetFileSubstrate, [152](#)
 OTF2_Archive_GetGlobalDefReader, [152](#)
 OTF2_Archive_GetGlobalDefWriter, [152](#)
 OTF2_Archive_GetGlobalEvtReader, [153](#)
 OTF2_Archive_GetGlobalSnapReader, [153](#)
 OTF2_Archive_GetMachineName, [153](#)
 OTF2_Archive_GetMarkerReader, [154](#)
 OTF2_Archive_GetMarkerWriter, [154](#)
 OTF2_Archive_GetNumberOfGlobalDefinitions, [154](#)
 OTF2_Archive_GetNumberOfLocations, [155](#)
 OTF2_Archive_GetNumberOfSnapshots, [155](#)

OTF2_Archive_GetNumberOfThumbFiles, 156
 OTF2_Archive_GetProperty, 156
 OTF2_Archive_GetPropertyNames, 156
 OTF2_Archive_GetSnapReader, 157
 OTF2_Archive_GetSnapWriter, 157
 OTF2_Archive_GetThumbReader, 158
 OTF2_Archive_GetThumbWriter, 158
 OTF2_Archive_GetTraceId, 159
 OTF2_Archive_GetVersion, 159
 OTF2_Archive_Open, 159
 OTF2_Archive_OpenDefFiles, 161
 OTF2_Archive_OpenEvtFiles, 161
 OTF2_Archive_OpenSnapFiles, 161
 OTF2_Archive_SelectLocation, 162
 OTF2_Archive_SetBoolProperty, 162
 OTF2_Archive_SetCollectiveCallbacks, 163
 OTF2_Archive_SetCreator, 163
 OTF2_Archive_SetDescription, 164
 OTF2_Archive_SetFlushCallbacks, 164
 OTF2_Archive_SetMachineName, 164
 OTF2_Archive_SetMemoryCallbacks, 165
 OTF2_Archive_SetNumberOfSnapshots, 165
 OTF2_Archive_SetProperty, 166
 OTF2_Archive_SetSerialCollectiveCallbacks, 166
 OTF2_Archive_SwitchFileMode, 167
 OTF2_CHUNK_SIZE_DEFINITION, 142
 OTF2_CHUNK_SIZE_EVENTS_DEFAULT, 142
 OTF2_Archive_Close, 143
 OTF2_Archive_CloseDefFiles, 143
 OTF2_Archive_CloseDefReader, 143
 OTF2_Archive_CloseDefWriter, 144
 OTF2_Archive_CloseEvtFiles, 144
 OTF2_Archive_CloseEvtReader, 144
 OTF2_Archive_CloseEvtWriter, 145
 OTF2_Archive_CloseGlobalDefReader, 145
 OTF2_Archive_CloseGlobalDefWriter, 145
 OTF2_Archive_CloseGlobalEvtReader, 146
 OTF2_Archive_CloseGlobalSnapReader, 146
 OTF2_Archive_CloseMarkerReader, 146
 OTF2_Archive_CloseMarkerWriter, 147
 OTF2_Archive_CloseSnapFiles, 147
 OTF2_Archive_CloseSnapReader, 148
 OTF2_Archive_CloseSnapWriter, 148
 OTF2_Archive_GetChunkSize, 149
 OTF2_Archive_GetCompression, 149
 OTF2_Archive_GetCreator, 150
 OTF2_Archive_GetDefReader, 150
 OTF2_Archive_GetDefWriter, 150
 OTF2_Archive_GetDescription, 151
 OTF2_Archive_GetEvtReader, 151
 OTF2_Archive_GetEvtWriter, 151
 OTF2_Archive_GetFileSubstrate, 152

INDEX

- OTF2_Archive_GetGlobalDefReader
OTF2_Archive.h, [152](#)
- OTF2_Archive_GetGlobalDefWriter
OTF2_Archive.h, [152](#)
- OTF2_Archive_GetGlobalEvtReader
OTF2_Archive.h, [153](#)
- OTF2_Archive_GetGlobalSnapReader
OTF2_Archive.h, [153](#)
- OTF2_Archive_GetMachineName
OTF2_Archive.h, [153](#)
- OTF2_Archive_GetMarkerReader
OTF2_Archive.h, [154](#)
- OTF2_Archive_GetMarkerWriter
OTF2_Archive.h, [154](#)
- OTF2_Archive_GetNumberOfGlobalDefinitions
OTF2_Archive.h, [154](#)
- OTF2_Archive_GetNumberOfLocations
OTF2_Archive.h, [155](#)
- OTF2_Archive_GetNumberOfSnapshots
OTF2_Archive.h, [155](#)
- OTF2_Archive_GetNumberOfThumbnails
OTF2_Archive.h, [156](#)
- OTF2_Archive_GetProperty
OTF2_Archive.h, [156](#)
- OTF2_Archive_GetPropertyNames
OTF2_Archive.h, [156](#)
- OTF2_Archive_GetSnapReader
OTF2_Archive.h, [157](#)
- OTF2_Archive_GetSnapWriter
OTF2_Archive.h, [157](#)
- OTF2_Archive_GetThumbReader
OTF2_Archive.h, [158](#)
- OTF2_Archive_GetThumbWriter
OTF2_Archive.h, [158](#)
- OTF2_Archive_GetTraceId
OTF2_Archive.h, [159](#)
- OTF2_Archive_GetVersion
OTF2_Archive.h, [159](#)
- OTF2_Archive_Open
OTF2_Archive.h, [159](#)
- OTF2_Archive_OpenDefFiles
OTF2_Archive.h, [161](#)
- OTF2_Archive_OpenEvtFiles
OTF2_Archive.h, [161](#)
- OTF2_Archive_OpenSnapFiles
OTF2_Archive.h, [161](#)
- OTF2_Archive_SelectLocation
OTF2_Archive.h, [162](#)
- OTF2_Archive_SetBoolProperty
OTF2_Archive.h, [162](#)
- OTF2_Archive_SetCollectiveCallbacks
OTF2_Archive.h, [163](#)
- OTF2_Archive_SetCreator
OTF2_Archive.h, [163](#)
- OTF2_Archive_SetDescription
OTF2_Archive.h, [164](#)
- OTF2_Archive_SetFlushCallbacks
OTF2_Archive.h, [164](#)
- OTF2_Archive_SetMachineName
OTF2_Archive.h, [164](#)
- OTF2_Archive_SetMemoryCallbacks
OTF2_Archive.h, [165](#)
- OTF2_Archive_SetNumberOfSnapshots
OTF2_Archive.h, [165](#)
- OTF2_Archive_SetProperty
OTF2_Archive.h, [166](#)
- OTF2_Archive_SetSerialCollectiveCallbacks
OTF2_Archive.h, [166](#)
- OTF2_Archive_SwitchFileMode
OTF2_Archive.h, [167](#)
- OTF2_AttributeList.h
OTF2_AttributeList_AddAttribute, [172](#)
OTF2_AttributeList_AddAttributeRef,
[173](#)
OTF2_AttributeList_AddCommRef,
[173](#)
OTF2_AttributeList_AddDouble, [174](#)
OTF2_AttributeList_AddFloat, [174](#)
OTF2_AttributeList_AddGroupRef,
[174](#)
OTF2_AttributeList_AddInt16, [175](#)
OTF2_AttributeList_AddInt32, [175](#)
OTF2_AttributeList_AddInt64, [176](#)
OTF2_AttributeList_AddInt8, [176](#)
OTF2_AttributeList_AddLocationRef,
[176](#)
OTF2_AttributeList_AddMetricRef,
[177](#)

OTF2_AttributeList_AddParameterRef, 177
 OTF2_AttributeList_AddRegionRef, 177
 OTF2_AttributeList_AddRmaWinRef, 178
 OTF2_AttributeList_AddString, 178
 OTF2_AttributeList_AddStringRef, 179
 OTF2_AttributeList_AddUInt16, 179
 OTF2_AttributeList_AddUInt32, 180
 OTF2_AttributeList_AddUInt64, 180
 OTF2_AttributeList_AddUInt8, 180
 OTF2_AttributeList_Delete, 181
 OTF2_AttributeList_GetAttributeByID, 181
 OTF2_AttributeList_GetAttributeByIndex, 181
 OTF2_AttributeList_GetAttributeRef, 182
 OTF2_AttributeList_GetCommRef, 182
 OTF2_AttributeList_GetDouble, 183
 OTF2_AttributeList_GetFloat, 183
 OTF2_AttributeList_GetGroupRef, 183
 OTF2_AttributeList_GetInt16, 184
 OTF2_AttributeList_GetInt32, 184
 OTF2_AttributeList_GetInt64, 185
 OTF2_AttributeList_GetInt8, 185
 OTF2_AttributeList_GetLocationRef, 185
 OTF2_AttributeList_GetMetricRef, 186
 OTF2_AttributeList_GetNumberOfElements, 186
 OTF2_AttributeList_GetParameterRef, 187
 OTF2_AttributeList_GetRegionRef, 187
 OTF2_AttributeList_GetRmaWinRef, 187
 OTF2_AttributeList_GetString, 188
 OTF2_AttributeList_GetStringRef, 188
 OTF2_AttributeList_GetUInt16, 189
 OTF2_AttributeList_GetUInt32, 189
 OTF2_AttributeList_GetUInt64, 190
 OTF2_AttributeList_GetUInt8, 190
 OTF2_AttributeList_New, 191
 OTF2_AttributeList_PopAttribute, 191
 OTF2_AttributeList_RemoveAllAttributes, 191
 OTF2_AttributeList_RemoveAttribute, 191
 OTF2_AttributeList_TestAttributeByID, 192
 OTF2_AttributeList_AddAttribute
 OTF2_AttributeList.h, 172
 OTF2_AttributeList_AddAttributeRef
 OTF2_AttributeList.h, 173
 OTF2_AttributeList_AddCommRef
 OTF2_AttributeList.h, 173
 OTF2_AttributeList_AddDouble
 OTF2_AttributeList.h, 174
 OTF2_AttributeList_AddFloat
 OTF2_AttributeList.h, 174
 OTF2_AttributeList_AddGroupRef
 OTF2_AttributeList.h, 174
 OTF2_AttributeList_AddInt16
 OTF2_AttributeList.h, 175
 OTF2_AttributeList_AddInt32
 OTF2_AttributeList.h, 175
 OTF2_AttributeList_AddInt64
 OTF2_AttributeList.h, 176
 OTF2_AttributeList_AddInt8
 OTF2_AttributeList.h, 176
 OTF2_AttributeList_AddLocationRef
 OTF2_AttributeList.h, 176
 OTF2_AttributeList_AddMetricRef
 OTF2_AttributeList.h, 177
 OTF2_AttributeList_AddParameterRef
 OTF2_AttributeList.h, 177
 OTF2_AttributeList_AddRegionRef
 OTF2_AttributeList.h, 177
 OTF2_AttributeList_AddRmaWinRef
 OTF2_AttributeList.h, 178
 OTF2_AttributeList_AddString
 OTF2_AttributeList.h, 178

INDEX

- OTF2_AttributeList_AddStringRef
OTF2_AttributeList.h, [179](#)
- OTF2_AttributeList_AddUint16
OTF2_AttributeList.h, [179](#)
- OTF2_AttributeList_AddUint32
OTF2_AttributeList.h, [180](#)
- OTF2_AttributeList_AddUint64
OTF2_AttributeList.h, [180](#)
- OTF2_AttributeList_AddUint8
OTF2_AttributeList.h, [180](#)
- OTF2_AttributeList_Delete
OTF2_AttributeList.h, [181](#)
- OTF2_AttributeList_GetAttributeByID
OTF2_AttributeList.h, [181](#)
- OTF2_AttributeList_GetAttributeByIndex
OTF2_AttributeList.h, [181](#)
- OTF2_AttributeList_GetAttributeRef
OTF2_AttributeList.h, [182](#)
- OTF2_AttributeList_GetCommRef
OTF2_AttributeList.h, [182](#)
- OTF2_AttributeList_GetDouble
OTF2_AttributeList.h, [183](#)
- OTF2_AttributeList_GetFloat
OTF2_AttributeList.h, [183](#)
- OTF2_AttributeList_GetGroupRef
OTF2_AttributeList.h, [183](#)
- OTF2_AttributeList_GetInt16
OTF2_AttributeList.h, [184](#)
- OTF2_AttributeList_GetInt32
OTF2_AttributeList.h, [184](#)
- OTF2_AttributeList_GetInt64
OTF2_AttributeList.h, [185](#)
- OTF2_AttributeList_GetInt8
OTF2_AttributeList.h, [185](#)
- OTF2_AttributeList_GetLocationRef
OTF2_AttributeList.h, [185](#)
- OTF2_AttributeList_GetMetricRef
OTF2_AttributeList.h, [186](#)
- OTF2_AttributeList_GetNumberOfElements
OTF2_AttributeList.h, [186](#)
- OTF2_AttributeList_GetParameterRef
OTF2_AttributeList.h, [187](#)
- OTF2_AttributeList_GetRegionRef
OTF2_AttributeList.h, [187](#)
- OTF2_AttributeList_GetRmaWinRef
OTF2_AttributeList.h, [187](#)
- OTF2_AttributeList_GetString
OTF2_AttributeList.h, [188](#)
- OTF2_AttributeList_GetStringRef
OTF2_AttributeList.h, [188](#)
- OTF2_AttributeList_GetUint16
OTF2_AttributeList.h, [189](#)
- OTF2_AttributeList_GetUint32
OTF2_AttributeList.h, [189](#)
- OTF2_AttributeList_GetUint64
OTF2_AttributeList.h, [190](#)
- OTF2_AttributeList_GetUint8
OTF2_AttributeList.h, [190](#)
- OTF2_AttributeList_New
OTF2_AttributeList.h, [191](#)
- OTF2_AttributeList_PopAttribute
OTF2_AttributeList.h, [191](#)
- OTF2_AttributeList_RemoveAllAttributes
OTF2_AttributeList.h, [191](#)
- OTF2_AttributeList_RemoveAttribute
OTF2_AttributeList.h, [191](#)
- OTF2_AttributeList_TestAttributeByID
OTF2_AttributeList.h, [192](#)
- OTF2_AttributeValue, [121](#)
- OTF2_CallbackCode
OTF2_GeneralDefinitions.h, [455](#)
- OTF2_CartPeriodicity_enum
OTF2_Definitions.h, [200](#)
- OTF2_CHUNK_SIZE_DEFINITIONS_
DEFAULT
OTF2_Archive.h, [142](#)
- OTF2_CHUNK_SIZE_EVENTS_DEFAULT
OTF2_Archive.h, [142](#)
- OTF2_CollectiveCallbacks, [123](#)
- OTF2_CollectiveContext, [123](#)
- OTF2_CollectiveOp_enum
OTF2_Events.h, [272](#)
- OTF2_Collectives_Barrier
Operating OTF2 in an collective con-
text, [97](#)
- OTF2_Collectives_Bcast
Operating OTF2 in an collective con-
text, [98](#)

- OTF2_Collectives_CreateLocalComm
 - Operating OTF2 in an collective context, [98](#)
- OTF2_Collectives_FreeLocalComm
 - Operating OTF2 in an collective context, [98](#)
- OTF2_Collectives_Gather
 - Operating OTF2 in an collective context, [99](#)
- OTF2_Collectives_Gatherv
 - Operating OTF2 in an collective context, [99](#)
- OTF2_Collectives_GetRank
 - Operating OTF2 in an collective context, [99](#)
- OTF2_Collectives_GetSize
 - Operating OTF2 in an collective context, [100](#)
- OTF2_Collectives_Release
 - Operating OTF2 in an collective context, [100](#)
- OTF2_Collectives_Scatter
 - Operating OTF2 in an collective context, [100](#)
- OTF2_Collectives_Scatterv
 - Operating OTF2 in an collective context, [101](#)
- OTF2_Compression_enum
 - OTF2_GeneralDefinitions.h, [456](#)
- OTF2_Definitions.h
 - OTF2_CartPeriodicity_enum, [200](#)
 - OTF2_GroupFlag_enum, [200](#)
 - OTF2_GroupType_enum, [201](#)
 - OTF2_LocationGroupType_enum, [200](#)
 - OTF2_LocationType_enum, [202](#)
 - OTF2_MetricBase_enum, [202](#)
 - OTF2_MetricMode_enum, [203](#)
 - OTF2_MetricOccurrence_enum, [203](#)
 - OTF2_MetricScope_enum, [204](#)
 - OTF2_MetricTiming_enum, [204](#)
 - OTF2_MetricType_enum, [205](#)
 - OTF2_MetricValueProperty_enum, [205](#)
 - OTF2_ParameterType_enum, [206](#)
 - OTF2_RecorderKind_enum, [206](#)
 - OTF2_RegionFlag_enum, [206](#)
 - OTF2_RegionRole_enum, [207](#)
 - OTF2_SystemTreeDomain_enum, [209](#)
- OTF2_DefReader.h
 - OTF2_DefReader_GetLocationID, [210](#)
 - OTF2_DefReader_ReadDefinitions, [210](#)
 - OTF2_DefReader_SetCallbacks, [211](#)
- OTF2_DefReader_GetLocationID
 - OTF2_DefReader.h, [210](#)
- OTF2_DefReader_ReadDefinitions
 - OTF2_DefReader.h, [210](#)
- OTF2_DefReader_SetCallbacks
 - OTF2_DefReader.h, [211](#)
- OTF2_DefReaderCallback_Attribute
 - OTF2_DefReaderCallbacks.h, [218](#)
- OTF2_DefReaderCallback_Callpath
 - OTF2_DefReaderCallbacks.h, [219](#)
- OTF2_DefReaderCallback_Callsite
 - OTF2_DefReaderCallbacks.h, [219](#)
- OTF2_DefReaderCallback_CartCoordinate
 - OTF2_DefReaderCallbacks.h, [220](#)
- OTF2_DefReaderCallback_CartDimension
 - OTF2_DefReaderCallbacks.h, [221](#)
- OTF2_DefReaderCallback_CartTopology
 - OTF2_DefReaderCallbacks.h, [221](#)
- OTF2_DefReaderCallback_ClockOffset
 - OTF2_DefReaderCallbacks.h, [222](#)
- OTF2_DefReaderCallback_Comm
 - OTF2_DefReaderCallbacks.h, [223](#)
- OTF2_DefReaderCallback_Group
 - OTF2_DefReaderCallbacks.h, [224](#)
- OTF2_DefReaderCallback_Location
 - OTF2_DefReaderCallbacks.h, [224](#)
- OTF2_DefReaderCallback_LocationGroup
 - OTF2_DefReaderCallbacks.h, [225](#)
- OTF2_DefReaderCallback_LocationGroupProperty
 - OTF2_DefReaderCallbacks.h, [226](#)
- OTF2_DefReaderCallback_LocationProperty
 - OTF2_DefReaderCallbacks.h, [226](#)
- OTF2_DefReaderCallback_MappingTable
 - OTF2_DefReaderCallbacks.h, [227](#)
- OTF2_DefReaderCallback_MetricClass

INDEX

- OTF2_DefReaderCallbacks.h, [227](#)
- OTF2_DefReaderCallback_MetricClassRecorder [225](#)
- OTF2_DefReaderCallbacks.h, [228](#)
- OTF2_DefReaderCallback_MetricInstance
- OTF2_DefReaderCallbacks.h, [229](#)
- OTF2_DefReaderCallback_MetricMember
- OTF2_DefReaderCallbacks.h, [230](#)
- OTF2_DefReaderCallback_Parameter
- OTF2_DefReaderCallbacks.h, [231](#)
- OTF2_DefReaderCallback_Region
- OTF2_DefReaderCallbacks.h, [231](#)
- OTF2_DefReaderCallback_RmaWin
- OTF2_DefReaderCallbacks.h, [232](#)
- OTF2_DefReaderCallback_String
- OTF2_DefReaderCallbacks.h, [233](#)
- OTF2_DefReaderCallback_SystemTreeNode
- OTF2_DefReaderCallbacks.h, [233](#)
- OTF2_DefReaderCallback_SystemTreeNodeDomain
- OTF2_DefReaderCallbacks.h, [234](#)
- OTF2_DefReaderCallback_SystemTreeNodeProperty
- OTF2_DefReaderCallbacks.h, [234](#)
- OTF2_DefReaderCallback_Unknown
- OTF2_DefReaderCallbacks.h, [235](#)
- OTF2_DefReaderCallbacks.h
- OTF2_DefReaderCallback_Attribute,
- [218](#)
- OTF2_DefReaderCallback_Callpath,
- [219](#)
- OTF2_DefReaderCallback_Callsite,
- [219](#)
- OTF2_DefReaderCallback_CartCoordinate,
- [220](#)
- OTF2_DefReaderCallback_CartDimension,
- [221](#)
- OTF2_DefReaderCallback_CartTopology,
- [221](#)
- OTF2_DefReaderCallback_ClockOffset,
- [222](#)
- OTF2_DefReaderCallback_Comm,
- [223](#)
- OTF2_DefReaderCallback_Group, [224](#)
- OTF2_DefReaderCallback_Location,
- [224](#)
- OTF2_DefReaderCallback_LocationGroup,
- [225](#)
- OTF2_DefReaderCallback_LocationGroupProperty,
- [226](#)
- OTF2_DefReaderCallback_LocationProperty,
- [226](#)
- OTF2_DefReaderCallback_MappingTable,
- [227](#)
- OTF2_DefReaderCallback_MetricClass,
- [227](#)
- OTF2_DefReaderCallback_MetricClassRecorder,
- [228](#)
- OTF2_DefReaderCallback_MetricInstance,
- [229](#)
- OTF2_DefReaderCallback_MetricMember,
- [230](#)
- OTF2_DefReaderCallback_Parameter,
- [231](#)
- OTF2_DefReaderCallback_Region,
- [231](#)
- OTF2_DefReaderCallback_RmaWin,
- [232](#)
- OTF2_DefReaderCallback_String, [233](#)
- OTF2_DefReaderCallback_SystemTreeNode,
- [233](#)
- OTF2_DefReaderCallback_SystemTreeNodeDomain,
- [234](#)
- OTF2_DefReaderCallback_SystemTreeNodeProperty,
- [234](#)
- OTF2_DefReaderCallback_Unknown,
- [235](#)
- OTF2_DefReaderCallbacks_Clear, [236](#)
- OTF2_DefReaderCallbacks_Delete,
- [236](#)
- OTF2_DefReaderCallbacks_New, [236](#)
- OTF2_DefReaderCallbacks_SetAttributeCallback,
- [236](#)
- OTF2_DefReaderCallbacks_SetCallpathCallback,
- [237](#)
- OTF2_DefReaderCallbacks_SetCallsiteCallback,
- [237](#)
- OTF2_DefReaderCallbacks_SetCartCoordinateCallback,
- [238](#)

OTF2_DefReaderCallbacks_SetCartCoordinateCallback
 238 OTF2_DefReaderCallbacks_Clear
 OTF2_DefReaderCallbacks_SetCartTopologyCallback
 239 OTF2_DefReaderCallbacks_Delete
 OTF2_DefReaderCallbacks_SetClockOffsetCallback
 239 OTF2_DefReaderCallbacks_New
 OTF2_DefReaderCallbacks_SetCommCallback
 240 OTF2_DefReaderCallbacks_SetAttributeCallback
 OTF2_DefReaderCallbacks_SetGroupCallback
 240 OTF2_DefReaderCallbacks_SetCallpathCallback
 OTF2_DefReaderCallbacks_SetLocationCallback
 241 OTF2_DefReaderCallbacks_SetCallsiteCallback
 OTF2_DefReaderCallbacks_SetLocationCallback
 241 OTF2_DefReaderCallbacks_SetCartCoordinateCallback
 OTF2_DefReaderCallbacks_SetLocationCallback
 242 OTF2_DefReaderCallbacks_SetCartDimensionCallback
 OTF2_DefReaderCallbacks_SetLocationCallback
 242 OTF2_DefReaderCallbacks_SetCartTopologyCallback
 OTF2_DefReaderCallbacks_SetMappingCallback
 243 OTF2_DefReaderCallbacks_SetClockOffsetCallback
 OTF2_DefReaderCallbacks_SetMetricCallback
 244 OTF2_DefReaderCallbacks_SetCommCallback
 OTF2_DefReaderCallbacks_SetMetricCallback
 244 OTF2_DefReaderCallbacks_SetGroupCallback
 OTF2_DefReaderCallbacks_SetMetricCallback
 245 OTF2_DefReaderCallbacks_SetLocationCallback
 OTF2_DefReaderCallbacks_SetMetricCallback
 245 OTF2_DefReaderCallbacks_SetLocationGroupCallback
 OTF2_DefReaderCallbacks_SetParameterCallback
 246 OTF2_DefReaderCallbacks_SetLocationGroupPropertyCallback
 OTF2_DefReaderCallbacks_SetRegionCallback
 246 OTF2_DefReaderCallbacks_SetLocationPropertyCallback
 OTF2_DefReaderCallbacks_SetRmapCallback
 247 OTF2_DefReaderCallbacks_SetMappingTableCallback
 OTF2_DefReaderCallbacks_SetStringCallback
 247 OTF2_DefReaderCallbacks_SetMetricClassCallback
 OTF2_DefReaderCallbacks_SetSystemCallback
 248 OTF2_DefReaderCallbacks_SetMetricClassRecorderCallback
 OTF2_DefReaderCallbacks_SetSystemCallback
 248 OTF2_DefReaderCallbacks_SetMetricInstanceCallback
 OTF2_DefReaderCallbacks_SetSystemCallback
 249 OTF2_DefReaderCallbacks_SetMetricMemberCallback
 OTF2_DefReaderCallbacks_SetUnknownCallback
 250 OTF2_DefReaderCallbacks_SetParameterCallback
 OTF2_DefReaderCallbacks.h, 236
 OTF2_DefReaderCallbacks.h, 236
 OTF2_DefReaderCallbacks.h, 236
 OTF2_DefReaderCallbacks.h, 236
 OTF2_DefReaderCallbacks.h, 236
 OTF2_DefReaderCallbacks.h, 237
 OTF2_DefReaderCallbacks.h, 237
 OTF2_DefReaderCallbacks.h, 238
 OTF2_DefReaderCallbacks.h, 238
 OTF2_DefReaderCallbacks.h, 239
 OTF2_DefReaderCallbacks.h, 239
 OTF2_DefReaderCallbacks.h, 240
 OTF2_DefReaderCallbacks.h, 240
 OTF2_DefReaderCallbacks.h, 241
 OTF2_DefReaderCallbacks.h, 241
 OTF2_DefReaderCallbacks.h, 242
 OTF2_DefReaderCallbacks.h, 242
 OTF2_DefReaderCallbacks.h, 243
 OTF2_DefReaderCallbacks.h, 244
 OTF2_DefReaderCallbacks.h, 244
 OTF2_DefReaderCallbacks.h, 245
 OTF2_DefReaderCallbacks.h, 245
 OTF2_DefReaderCallbacks.h, 246
 OTF2_DefReaderCallbacks.h, 246
 OTF2_DefReaderCallbacks.h, 247
 OTF2_DefReaderCallbacks.h, 247
 OTF2_DefReaderCallbacks.h, 248
 OTF2_DefReaderCallbacks.h, 248
 OTF2_DefReaderCallbacks.h, 249
 OTF2_DefReaderCallbacks.h, 246

INDEX

OTF2_DefReaderCallbacks_SetRegionCallback, OTF2_DefWriter_WriteMetricMember,
OTF2_DefReaderCallbacks.h, 246 264

OTF2_DefReaderCallbacks_SetRmaWinCallback, OTF2_DefWriter_WriteParameter, 265
OTF2_DefReaderCallbacks.h, 247 OTF2_DefWriter_WriteRegion, 266

OTF2_DefReaderCallbacks_SetStringCallback, OTF2_DefWriter_WriteRmaWin, 266
OTF2_DefReaderCallbacks.h, 247 OTF2_DefWriter_WriteString, 267

OTF2_DefReaderCallbacks_SetSystemTreeNodeCallback, OTF2_DefWriter_WriteSystemTreeNode,
OTF2_DefReaderCallbacks.h, 248 267

OTF2_DefReaderCallbacks_SetSystemTreeNodeDomainCallback, OTF2_DefWriter_WriteSystemTreeNodeDomain,
OTF2_DefReaderCallbacks.h, 248 268

OTF2_DefReaderCallbacks_SetSystemTreeNodePropertyCallback, OTF2_DefWriter_WriteSystemTreeNodeProperty,
OTF2_DefReaderCallbacks.h, 249 269

OTF2_DefReaderCallbacks_SetUnknownCallback, OTF2_DefWriter_GetLocationID
OTF2_DefReaderCallbacks.h, 250 OTF2_DefWriter.h, 253

OTF2_DefWriter.h OTF2_DefWriter_WriteAttribute
OTF2_DefWriter_GetLocationID, 253 OTF2_DefWriter.h, 254

OTF2_DefWriter_WriteAttribute, 254 OTF2_DefWriter_WriteCallpath
OTF2_DefWriter_WriteCallpath, 254 OTF2_DefWriter.h, 254

OTF2_DefWriter_WriteCallsite, 255 OTF2_DefWriter_WriteCallsite
OTF2_DefWriter_WriteCartCoordinate, OTF2_DefWriter.h, 255
255 OTF2_DefWriter_WriteCartCoordinate

OTF2_DefWriter_WriteCartDimension, OTF2_DefWriter.h, 255
256 OTF2_DefWriter_WriteCartDimension

OTF2_DefWriter_WriteCartTopology, OTF2_DefWriter.h, 256
257 OTF2_DefWriter_WriteCartTopology

OTF2_DefWriter_WriteClockOffset, OTF2_DefWriter.h, 257
257 OTF2_DefWriter_WriteClockOffset

OTF2_DefWriter_WriteComm, 258 OTF2_DefWriter.h, 257

OTF2_DefWriter_WriteGroup, 258 OTF2_DefWriter_WriteComm

OTF2_DefWriter_WriteLocation, 259 OTF2_DefWriter.h, 258

OTF2_DefWriter_WriteLocationGroup, OTF2_DefWriter_WriteGroup
260 OTF2_DefWriter.h, 258

OTF2_DefWriter_WriteLocationGroupProperty, OTF2_DefWriter_WriteLocation
260 OTF2_DefWriter.h, 259

OTF2_DefWriter_WriteLocationProperty, OTF2_DefWriter_WriteLocationGroup
261 OTF2_DefWriter.h, 260

OTF2_DefWriter_WriteMappingTable, OTF2_DefWriter_WriteLocationGroupProperty
262 OTF2_DefWriter.h, 260

OTF2_DefWriter_WriteMetricClass, OTF2_DefWriter_WriteLocationProperty
262 OTF2_DefWriter.h, 261

OTF2_DefWriter_WriteMetricClassReference, OTF2_DefWriter_WriteMappingTable
263 OTF2_DefWriter.h, 262

OTF2_DefWriter_WriteMetricInstance, OTF2_DefWriter_WriteMetricClass
263 OTF2_DefWriter.h, 262

OTF2_DefWriter_WriteMetricClassRecorder
 OTF2_DefWriter.h, [263](#)
 OTF2_DefWriter_WriteMetricInstance
 OTF2_DefWriter.h, [263](#)
 OTF2_DefWriter_WriteMetricMember
 OTF2_DefWriter.h, [264](#)
 OTF2_DefWriter_WriteParameter
 OTF2_DefWriter.h, [265](#)
 OTF2_DefWriter_WriteRegion
 OTF2_DefWriter.h, [266](#)
 OTF2_DefWriter_WriteRmaWin
 OTF2_DefWriter.h, [266](#)
 OTF2_DefWriter_WriteString
 OTF2_DefWriter.h, [267](#)
 OTF2_DefWriter_WriteSystemTreeNode
 OTF2_DefWriter.h, [267](#)
 OTF2_DefWriter_WriteSystemTreeNodeDomain
 OTF2_DefWriter.h, [268](#)
 OTF2_DefWriter_WriteSystemTreeNodeProperty
 OTF2_DefWriter.h, [269](#)
 OTF2_Error_GetDescription
 OTF2_ErrorCodes.h, [135](#)
 OTF2_Error_GetName
 OTF2_ErrorCodes.h, [135](#)
 OTF2_Error_RegisterCallback
 OTF2_ErrorCodes.h, [135](#)
 OTF2_ErrorCallback
 OTF2_ErrorCodes.h, [131](#)
 OTF2_ErrorCode
 OTF2_ErrorCodes.h, [131](#)
 OTF2_ErrorCodes.h
 OTF2_Error_GetDescription, [135](#)
 OTF2_Error_GetName, [135](#)
 OTF2_Error_RegisterCallback, [135](#)
 OTF2_ErrorCallback, [131](#)
 OTF2_ErrorCode, [131](#)
 OTF2_Events.h
 OTF2_CollectiveOp_enum, [272](#)
 OTF2_LockType_enum, [273](#)
 OTF2_MeasurementMode_enum, [273](#)
 OTF2_RmaAtomicType_enum, [274](#)
 OTF2_RmaSyncLevel_enum, [275](#)
 OTF2_RmaSyncType_enum, [275](#)
 OTF2_EventSizeEstimator.h
 OTF2_EventSizeEstimator_Delete, [281](#)
 OTF2_EventSizeEstimator_GetSizeOfAttributeList,
 [282](#)
 OTF2_EventSizeEstimator_GetSizeOfBufferFlushEvent,
 [282](#)
 OTF2_EventSizeEstimator_GetSizeOfEnterEvent,
 [282](#)
 OTF2_EventSizeEstimator_GetSizeOfLeaveEvent,
 [283](#)
 OTF2_EventSizeEstimator_GetSizeOfMeasurementOnOffEvent,
 [283](#)
 OTF2_EventSizeEstimator_GetSizeOfMetricEvent,
 [284](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiCollectiveBeginEvent,
 [284](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiCollectiveEndEvent,
 [284](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiIrecvEvent,
 [285](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiIrecvRequestEvent,
 [285](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiIsendCompleteEvent,
 [286](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiIsendEvent,
 [286](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiRecvEvent,
 [286](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiRequestCancelledEvent,
 [287](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiRequestTestEvent,
 [287](#)
 OTF2_EventSizeEstimator_GetSizeOfMpiSendEvent,
 [288](#)
 OTF2_EventSizeEstimator_GetSizeOfOmpAcquireLockEvent,
 [288](#)
 OTF2_EventSizeEstimator_GetSizeOfOmpForkEvent,
 [288](#)
 OTF2_EventSizeEstimator_GetSizeOfOmpJoinEvent,
 [289](#)
 OTF2_EventSizeEstimator_GetSizeOfOmpReleaseLockEvent,
 [289](#)
 OTF2_EventSizeEstimator_GetSizeOfOmpTaskCompleteEvent,
 [290](#)

INDEX

OTF2_EventSizeEstimator_GetSizeOfOmpTaskCreateEvent, 290
OTF2_EventSizeEstimator_GetSizeOfOmpTaskSwitchEvent, 291
OTF2_EventSizeEstimator_GetSizeOfParameterIntEvent, 291
OTF2_EventSizeEstimator_GetSizeOfParameterStringEvent, 292
OTF2_EventSizeEstimator_GetSizeOfParameterUnsignedIntEvent, 292
OTF2_EventSizeEstimator_GetSizeOfRmaAcquireLockEvent, 292
OTF2_EventSizeEstimator_GetSizeOfRmaAtomicEvent, 293
OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveBeginEvent, 293
OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveEndEvent, 293
OTF2_EventSizeEstimator_GetSizeOfRmaGetEvent, 294
OTF2_EventSizeEstimator_GetSizeOfRmaGroupSyncEvent, 294
OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteBlockingEvent, 295
OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteNonBlockingEvent, 295
OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteRemoteEvent, 295
OTF2_EventSizeEstimator_GetSizeOfRmaOpTestEvent, 296
OTF2_EventSizeEstimator_GetSizeOfRmaPutEvent, 296
OTF2_EventSizeEstimator_GetSizeOfRmaReleaseLockEvent, 297
OTF2_EventSizeEstimator_GetSizeOfRmaRequestLockEvent, 297
OTF2_EventSizeEstimator_GetSizeOfRmaSyncEvent, 297
OTF2_EventSizeEstimator_GetSizeOfRmaTryLockEvent, 298
OTF2_EventSizeEstimator_GetSizeOfRmaWaitChangeSizeEvent, 298
OTF2_EventSizeEstimator_GetSizeOfRmaWaitCreateEvent, 299
OTF2_EventSizeEstimator_GetSizeOfRmaWinDestroyEvent, 299
OTF2_EventSizeEstimator_GetSizeOfThreadAcquireLockEvent, 299
OTF2_EventSizeEstimator_GetSizeOfThreadBeginEvent, 300
OTF2_EventSizeEstimator_GetSizeOfThreadCreateEvent, 300
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 301
OTF2_EventSizeEstimator_GetSizeOfThreadForkEvent, 301
OTF2_EventSizeEstimator_GetSizeOfThreadJoinEvent, 301
OTF2_EventSizeEstimator_GetSizeOfThreadReleaseLockEvent, 302
OTF2_EventSizeEstimator_GetSizeOfThreadTaskCompleteEvent, 302
OTF2_EventSizeEstimator_GetSizeOfThreadTaskCreateEvent, 303
OTF2_EventSizeEstimator_GetSizeOfThreadTaskSwitchEvent, 303
OTF2_EventSizeEstimator_GetSizeOfThreadTeamBeginEvent, 303
OTF2_EventSizeEstimator_GetSizeOfThreadTeamEndEvent, 304
OTF2_EventSizeEstimator_GetSizeOfThreadWaitEvent, 304
OTF2_EventSizeEstimator_GetSizeOfTimestamp, 305
OTF2_EventSizeEstimator_New, 305
OTF2_EventSizeEstimator_SetNumberOfAttributeDefinitions, 305
OTF2_EventSizeEstimator_SetNumberOfCommDefinitions, 306
OTF2_EventSizeEstimator_SetNumberOfGroupDefinitions, 306
OTF2_EventSizeEstimator_SetNumberOfLocationDefinitions, 307
OTF2_EventSizeEstimator_SetNumberOfMetricDefinitions, 307
OTF2_EventSizeEstimator_SetNumberOfParameterDefinitions, 308

INDEX

- OTF2_EventSizeEstimator_SetNumberOfOmpJoinEvent
308
OTF2_EventSizeEstimator.h, 289
- OTF2_EventSizeEstimator_SetNumberOfOmpReleaseLockEvent
309
OTF2_EventSizeEstimator.h, 289
- OTF2_EventSizeEstimator_SetNumberOfOmpTaskCompleteEvent
309
OTF2_EventSizeEstimator.h, 290
- OTF2_EventSizeEstimator_Delete
OTF2_EventSizeEstimator_GetSizeOfOmpTaskCreateEvent
OTF2_EventSizeEstimator.h, 281
OTF2_EventSizeEstimator.h, 290
- OTF2_EventSizeEstimator_GetSizeOfAttributeEvent
OTF2_EventSizeEstimator_GetSizeOfOmpTaskSwitchEvent
OTF2_EventSizeEstimator.h, 282
OTF2_EventSizeEstimator.h, 291
- OTF2_EventSizeEstimator_GetSizeOfBufferEvent
OTF2_EventSizeEstimator_GetSizeOfParameterIntEvent
OTF2_EventSizeEstimator.h, 282
OTF2_EventSizeEstimator.h, 291
- OTF2_EventSizeEstimator_GetSizeOfEntryPointEvent
OTF2_EventSizeEstimator_GetSizeOfParameterStringEvent
OTF2_EventSizeEstimator.h, 282
OTF2_EventSizeEstimator.h, 292
- OTF2_EventSizeEstimator_GetSizeOfLeaderEvent
OTF2_EventSizeEstimator_GetSizeOfParameterUnsignedIntEvent
OTF2_EventSizeEstimator.h, 283
OTF2_EventSizeEstimator.h, 292
- OTF2_EventSizeEstimator_GetSizeOfMemoryEvent
OTF2_EventSizeEstimator_GetSizeOfRmaAcquireLockEvent
OTF2_EventSizeEstimator.h, 283
OTF2_EventSizeEstimator.h, 292
- OTF2_EventSizeEstimator_GetSizeOfMemoryEvent
OTF2_EventSizeEstimator_GetSizeOfRmaAtomicEvent
OTF2_EventSizeEstimator.h, 284
OTF2_EventSizeEstimator.h, 293
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveBeginEvent
OTF2_EventSizeEstimator.h, 284
OTF2_EventSizeEstimator.h, 293
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveEndEvent
OTF2_EventSizeEstimator.h, 284
OTF2_EventSizeEstimator.h, 293
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaGetEvent
OTF2_EventSizeEstimator.h, 285
OTF2_EventSizeEstimator.h, 294
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaGroupSyncEvent
OTF2_EventSizeEstimator.h, 285
OTF2_EventSizeEstimator.h, 294
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteBlockingEvent
OTF2_EventSizeEstimator.h, 286
OTF2_EventSizeEstimator.h, 295
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteNonBlockingEvent
OTF2_EventSizeEstimator.h, 286
OTF2_EventSizeEstimator.h, 295
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteRemoteEvent
OTF2_EventSizeEstimator.h, 286
OTF2_EventSizeEstimator.h, 295
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaOpTestEvent
OTF2_EventSizeEstimator.h, 287
OTF2_EventSizeEstimator.h, 296
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaPutEvent
OTF2_EventSizeEstimator.h, 287
OTF2_EventSizeEstimator.h, 296
- OTF2_EventSizeEstimator_GetSizeOfMpiEvent
OTF2_EventSizeEstimator_GetSizeOfRmaReleaseLockEvent
OTF2_EventSizeEstimator.h, 288
OTF2_EventSizeEstimator.h, 297
- OTF2_EventSizeEstimator_GetSizeOfOmpEvent
OTF2_EventSizeEstimator_GetSizeOfRmaRequestLockEvent
OTF2_EventSizeEstimator.h, 288
OTF2_EventSizeEstimator.h, 297
- OTF2_EventSizeEstimator_GetSizeOfOmpEvent
OTF2_EventSizeEstimator_GetSizeOfRmaSyncEvent
OTF2_EventSizeEstimator.h, 288
OTF2_EventSizeEstimator.h, 297

INDEX

OTF2_EventSizeEstimator_GetSizeOfRmaWinClockOffset, 307
OTF2_EventSizeEstimator.h, 298
OTF2_EventSizeEstimator_SetNumberOfLocationDefinitions
OTF2_EventSizeEstimator.h, 307
OTF2_EventSizeEstimator_GetSizeOfRmaWinClockOffset, 307
OTF2_EventSizeEstimator.h, 298
OTF2_EventSizeEstimator_SetNumberOfMetricDefinitions
OTF2_EventSizeEstimator.h, 307
OTF2_EventSizeEstimator_GetSizeOfRmaWinClockOffset, 307
OTF2_EventSizeEstimator.h, 299
OTF2_EventSizeEstimator_SetNumberOfParameterDefinitions
OTF2_EventSizeEstimator.h, 308
OTF2_EventSizeEstimator_GetSizeOfRmaWinClockOffset, 308
OTF2_EventSizeEstimator.h, 299
OTF2_EventSizeEstimator_SetNumberOfRegionDefinitions
OTF2_EventSizeEstimator.h, 308
OTF2_EventSizeEstimator_GetSizeOfRmaWinClockOffset, 308
OTF2_EventSizeEstimator.h, 299
OTF2_EventSizeEstimator_SetNumberOfRmaWinDefinitions
OTF2_EventSizeEstimator.h, 309
OTF2_EventSizeEstimator_GetSizeOfRmaWinClockOffset, 309
OTF2_EventSizeEstimator.h, 300
OTF2_EventSizeEstimator_SetNumberOfStringDefinitions
OTF2_EventSizeEstimator.h, 309
OTF2_EventSizeEstimator_GetSizeOfRmaWinClockOffset, 309
OTF2_EventSizeEstimator.h, 300
OTF2_EvtReader.h
OTF2_EventSizeEstimator.h, 300
OTF2_EvtReader_ApplyClockOffsets,
311
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 311
OTF2_EventSizeEstimator.h, 301
OTF2_EvtReader_ApplyMappingTables,
312
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 312
OTF2_EventSizeEstimator.h, 301
OTF2_EvtReader_GetLocationID, 312
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 312
OTF2_EventSizeEstimator.h, 301
OTF2_EvtReader_GetPos, 312
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 312
OTF2_EventSizeEstimator.h, 302
OTF2_EvtReader_ReadEvents, 313
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 313
OTF2_EventSizeEstimator.h, 302
OTF2_EvtReader_ReadEventsBackward,
313
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 313
OTF2_EventSizeEstimator.h, 302
OTF2_EvtReader_SetCallbacks, 314
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 314
OTF2_EventSizeEstimator.h, 303
OTF2_EvtReader_TimeStampRewrite,
314
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 314
OTF2_EventSizeEstimator.h, 303
OTF2_EvtReader_ApplyClockOffsets
OTF2_EventSizeEstimator.h, 303
OTF2_EvtReader.h, 311
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 311
OTF2_EventSizeEstimator.h, 303
OTF2_EvtReader_ApplyMappingTables
OTF2_EventSizeEstimator.h, 303
OTF2_EvtReader.h, 312
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 312
OTF2_EventSizeEstimator.h, 304
OTF2_EvtReader_GetLocationID
OTF2_EventSizeEstimator.h, 304
OTF2_EvtReader.h, 312
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 312
OTF2_EventSizeEstimator.h, 304
OTF2_EvtReader_GetPos
OTF2_EventSizeEstimator.h, 304
OTF2_EvtReader.h, 312
OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent, 312
OTF2_EventSizeEstimator.h, 305
OTF2_EvtReader_ReadEvents
OTF2_EventSizeEstimator.h, 305
OTF2_EvtReader.h, 313
OTF2_EventSizeEstimator_New
OTF2_EventSizeEstimator.h, 305
OTF2_EvtReader_ReadEventsBackward
OTF2_EventSizeEstimator.h, 305
OTF2_EvtReader.h, 313
OTF2_EventSizeEstimator_SetNumberOfThreadEndEvent, 313
OTF2_EventSizeEstimator.h, 305
OTF2_EvtReader_SetCallbacks
OTF2_EventSizeEstimator.h, 306
OTF2_EvtReader.h, 314
OTF2_EventSizeEstimator_SetNumberOfThreadEndEvent, 314
OTF2_EventSizeEstimator.h, 306
OTF2_EvtReader_TimeStampRewrite
OTF2_EventSizeEstimator.h, 306
OTF2_EvtReader.h, 314

OTF2_EvtReaderCallback_BufferFlush	OTF2_EvtReaderCallback_ParameterInt
OTF2_EvtReaderCallbacks.h, 328	OTF2_EvtReaderCallbacks.h, 344
OTF2_EvtReaderCallback_Enter	OTF2_EvtReaderCallback_ParameterString
OTF2_EvtReaderCallbacks.h, 329	OTF2_EvtReaderCallbacks.h, 345
OTF2_EvtReaderCallback_Leave	OTF2_EvtReaderCallback_ParameterUnsignedInt
OTF2_EvtReaderCallbacks.h, 330	OTF2_EvtReaderCallbacks.h, 346
OTF2_EvtReaderCallback_Measurement	OTF2_EvtReaderCallback_RmaAcquireLock
OTF2_EvtReaderCallbacks.h, 330	OTF2_EvtReaderCallbacks.h, 346
OTF2_EvtReaderCallback_Metric	OTF2_EvtReaderCallback_RmaAtomic
OTF2_EvtReaderCallbacks.h, 331	OTF2_EvtReaderCallbacks.h, 347
OTF2_EvtReaderCallback_MpiCollectiveBegin	OTF2_EvtReaderCallback_RmaCollectiveBegin
OTF2_EvtReaderCallbacks.h, 332	OTF2_EvtReaderCallbacks.h, 348
OTF2_EvtReaderCallback_MpiCollectiveEnd	OTF2_EvtReaderCallback_RmaCollectiveEnd
OTF2_EvtReaderCallbacks.h, 332	OTF2_EvtReaderCallbacks.h, 349
OTF2_EvtReaderCallback_MpiIrecv	OTF2_EvtReaderCallback_RmaGet
OTF2_EvtReaderCallbacks.h, 333	OTF2_EvtReaderCallbacks.h, 349
OTF2_EvtReaderCallback_MpiIrecvRequest	OTF2_EvtReaderCallback_RmaGroupSync
OTF2_EvtReaderCallbacks.h, 334	OTF2_EvtReaderCallbacks.h, 350
OTF2_EvtReaderCallback_MpiIsend	OTF2_EvtReaderCallback_RmaOpCompleteBlocking
OTF2_EvtReaderCallbacks.h, 335	OTF2_EvtReaderCallbacks.h, 351
OTF2_EvtReaderCallback_MpiIsendComplete	OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking
OTF2_EvtReaderCallbacks.h, 335	OTF2_EvtReaderCallbacks.h, 352
OTF2_EvtReaderCallback_MpiRecv	OTF2_EvtReaderCallback_RmaOpCompleteRemote
OTF2_EvtReaderCallbacks.h, 336	OTF2_EvtReaderCallbacks.h, 352
OTF2_EvtReaderCallback_MpiRequestCancel	OTF2_EvtReaderCallback_RmaOpTest
OTF2_EvtReaderCallbacks.h, 337	OTF2_EvtReaderCallbacks.h, 353
OTF2_EvtReaderCallback_MpiRequestTest	OTF2_EvtReaderCallback_RmaPut
OTF2_EvtReaderCallbacks.h, 338	OTF2_EvtReaderCallbacks.h, 354
OTF2_EvtReaderCallback_MpiSend	OTF2_EvtReaderCallback_RmaReleaseLock
OTF2_EvtReaderCallbacks.h, 338	OTF2_EvtReaderCallbacks.h, 355
OTF2_EvtReaderCallback_OmpAcquireLock	OTF2_EvtReaderCallback_RmaRequestLock
OTF2_EvtReaderCallbacks.h, 339	OTF2_EvtReaderCallbacks.h, 355
OTF2_EvtReaderCallback_OmpFork	OTF2_EvtReaderCallback_RmaSync
OTF2_EvtReaderCallbacks.h, 340	OTF2_EvtReaderCallbacks.h, 356
OTF2_EvtReaderCallback_OmpJoin	OTF2_EvtReaderCallback_RmaTryLock
OTF2_EvtReaderCallbacks.h, 341	OTF2_EvtReaderCallbacks.h, 357
OTF2_EvtReaderCallback_OmpReleaseLock	OTF2_EvtReaderCallback_RmaWaitChange
OTF2_EvtReaderCallbacks.h, 341	OTF2_EvtReaderCallbacks.h, 358
OTF2_EvtReaderCallback_OmpTaskComplete	OTF2_EvtReaderCallback_RmaWinCreate
OTF2_EvtReaderCallbacks.h, 342	OTF2_EvtReaderCallbacks.h, 358
OTF2_EvtReaderCallback_OmpTaskCreate	OTF2_EvtReaderCallback_RmaWinDestroy
OTF2_EvtReaderCallbacks.h, 343	OTF2_EvtReaderCallbacks.h, 359
OTF2_EvtReaderCallback_OmpTaskSwitch	OTF2_EvtReaderCallback_ThreadAcquireLock
OTF2_EvtReaderCallbacks.h, 343	OTF2_EvtReaderCallbacks.h, 360

INDEX

OTF2_EvtReaderCallback_ThreadBegin	OTF2_EvtReaderCallback_MpiIsendComplete,
OTF2_EvtReaderCallbacks.h, 361	335
OTF2_EvtReaderCallback_ThreadCreate	OTF2_EvtReaderCallback_MpiRecv,
OTF2_EvtReaderCallbacks.h, 361	336
OTF2_EvtReaderCallback_ThreadEnd	OTF2_EvtReaderCallback_MpiRequestCancelled,
OTF2_EvtReaderCallbacks.h, 362	337
OTF2_EvtReaderCallback_ThreadFork	OTF2_EvtReaderCallback_MpiRequestTest,
OTF2_EvtReaderCallbacks.h, 363	338
OTF2_EvtReaderCallback_ThreadJoin	OTF2_EvtReaderCallback_MpiSend,
OTF2_EvtReaderCallbacks.h, 363	338
OTF2_EvtReaderCallback_ThreadReleaseLock	OTF2_EvtReaderCallback_OmpAcquireLock,
OTF2_EvtReaderCallbacks.h, 364	339
OTF2_EvtReaderCallback_ThreadTaskComplete	OTF2_EvtReaderCallback_OmpFork,
OTF2_EvtReaderCallbacks.h, 365	340
OTF2_EvtReaderCallback_ThreadTaskCreate	OTF2_EvtReaderCallback_OmpJoin,
OTF2_EvtReaderCallbacks.h, 366	341
OTF2_EvtReaderCallback_ThreadTaskSwitch	OTF2_EvtReaderCallback_OmpReleaseLock,
OTF2_EvtReaderCallbacks.h, 366	341
OTF2_EvtReaderCallback_ThreadTeamBegin	OTF2_EvtReaderCallback_OmpTaskComplete,
OTF2_EvtReaderCallbacks.h, 367	342
OTF2_EvtReaderCallback_ThreadTeamEnd	OTF2_EvtReaderCallback_OmpTaskCreate,
OTF2_EvtReaderCallbacks.h, 368	343
OTF2_EvtReaderCallback_ThreadWait	OTF2_EvtReaderCallback_OmpTaskSwitch,
OTF2_EvtReaderCallbacks.h, 369	343
OTF2_EvtReaderCallback_Unknown	OTF2_EvtReaderCallback_ParameterInt,
OTF2_EvtReaderCallbacks.h, 369	344
OTF2_EvtReaderCallbacks.h	OTF2_EvtReaderCallback_ParameterString,
OTF2_EvtReaderCallback_BufferFlush,	345
328	OTF2_EvtReaderCallback_ParameterUnsignedInt,
OTF2_EvtReaderCallback_Enter, 329	346
OTF2_EvtReaderCallback_Leave, 330	OTF2_EvtReaderCallback_RmaAcquireLock,
OTF2_EvtReaderCallback_MeasurementOnOff,	346
330	OTF2_EvtReaderCallback_RmaAtomic,
OTF2_EvtReaderCallback_Metric, 331	347
OTF2_EvtReaderCallback_MpiCollectiveBegin,	OTF2_EvtReaderCallback_RmaCollectiveBegin,
332	348
OTF2_EvtReaderCallback_MpiCollectiveEnd,	OTF2_EvtReaderCallback_RmaCollectiveEnd,
332	349
OTF2_EvtReaderCallback_MpiIrecv,	OTF2_EvtReaderCallback_RmaGet,
333	349
OTF2_EvtReaderCallback_MpiIrecvRequest,	OTF2_EvtReaderCallback_RmaGroupSync,
334	350
OTF2_EvtReaderCallback_MpiIsend,	OTF2_EvtReaderCallback_RmaOpCompleteBlocking,
335	351

OTF2_EvtReaderCallback_RmaOpComplete, 352	OTF2_EvtReaderCallback_ThreadTeamEnd, 368
OTF2_EvtReaderCallback_RmaOpComplete, 352	OTF2_EvtReaderCallback_ThreadWait, 369
OTF2_EvtReaderCallback_RmaOpTest, 353	OTF2_EvtReaderCallback_Unknown, 369
OTF2_EvtReaderCallback_RmaPut, 354	OTF2_EvtReaderCallbacks_Clear, 370
OTF2_EvtReaderCallback_RmaReleaseLock, 355	OTF2_EvtReaderCallbacks_Delete, 370
OTF2_EvtReaderCallback_RmaRequestLock, 355	OTF2_EvtReaderCallbacks_New, 370
OTF2_EvtReaderCallback_RmaSync, 356	OTF2_EvtReaderCallbacks_SetBufferFlushCallback, 370
OTF2_EvtReaderCallback_RmaTryLock, 357	OTF2_EvtReaderCallbacks_SetEnterCallback, 371
OTF2_EvtReaderCallback_RmaWaitChange, 358	OTF2_EvtReaderCallbacks_SetLeaveCallback, 372
OTF2_EvtReaderCallback_RmaWinCreate, 358	OTF2_EvtReaderCallbacks_SetMeasurementOnOffCallback, 372
OTF2_EvtReaderCallback_RmaWinDestroy, 359	OTF2_EvtReaderCallbacks_SetMetricCallback, 373
OTF2_EvtReaderCallback_ThreadAcquireLock, 360	OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback, 373
OTF2_EvtReaderCallback_ThreadBegin, 361	OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback, 374
OTF2_EvtReaderCallback_ThreadCreate, 361	OTF2_EvtReaderCallbacks_SetMpiIrecvCallback, 374
OTF2_EvtReaderCallback_ThreadEnd, 362	OTF2_EvtReaderCallbacks_SetMpiIrecvRequestCallback, 375
OTF2_EvtReaderCallback_ThreadFork, 363	OTF2_EvtReaderCallbacks_SetMpiIsendCallback, 375
OTF2_EvtReaderCallback_ThreadJoin, 363	OTF2_EvtReaderCallbacks_SetMpiIsendCompleteCallback, 376
OTF2_EvtReaderCallback_ThreadReleaseLock, 364	OTF2_EvtReaderCallbacks_SetMpiRecvCallback, 377
OTF2_EvtReaderCallback_ThreadTaskComplete, 365	OTF2_EvtReaderCallbacks_SetMpiRequestCancelledCallback, 377
OTF2_EvtReaderCallback_ThreadTaskCreate, 366	OTF2_EvtReaderCallbacks_SetMpiRequestTestCallback, 378
OTF2_EvtReaderCallback_ThreadTaskSwitch, 366	OTF2_EvtReaderCallbacks_SetMpiSendCallback, 378
OTF2_EvtReaderCallback_ThreadTeamBegin, 367	OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback, 379
	OTF2_EvtReaderCallbacks_SetOmpForkCallback, 379

INDEX

OTF2_EvtReaderCallbacks_SetOmpJoinCallback, 380
OTF2_EvtReaderCallbacks_SetOmpJoinCallback, 392
OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback, 380
OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback, 393
OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback, 381
OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback, 394
OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback, 382
OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback, 394
OTF2_EvtReaderCallbacks_SetOmpTaskSwitchCallback, 382
OTF2_EvtReaderCallbacks_SetOmpTaskSwitchCallback, 395
OTF2_EvtReaderCallbacks_SetParameterCallback, 383
OTF2_EvtReaderCallbacks_SetParameterCallback, 395
OTF2_EvtReaderCallbacks_SetParameterSwitchCallback, 383
OTF2_EvtReaderCallbacks_SetParameterSwitchCallback, 396
OTF2_EvtReaderCallbacks_SetParameterUnlockCallback, 384
OTF2_EvtReaderCallbacks_SetParameterUnlockCallback, 396
OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback, 384
OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback, 397
OTF2_EvtReaderCallbacks_SetRmaAtomCallback, 385
OTF2_EvtReaderCallbacks_SetRmaAtomCallback, 397
OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback, 386
OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback, 398
OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback, 386
OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback, 398
OTF2_EvtReaderCallbacks_SetRmaGetCallback, 387
OTF2_EvtReaderCallbacks_SetRmaGetCallback, 399
OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback, 387
OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback, 400
OTF2_EvtReaderCallbacks_SetRmaOpCompleteBeginCallback, 388
OTF2_EvtReaderCallbacks_SetRmaOpCompleteBeginCallback, 400
OTF2_EvtReaderCallbacks_SetRmaOpCompleteEndCallback, 388
OTF2_EvtReaderCallbacks_SetRmaOpCompleteEndCallback, 401
OTF2_EvtReaderCallbacks_SetRmaOpCompleteRetainCallback, 389
OTF2_EvtReaderCallbacks_SetRmaOpCompleteRetainCallback, 401
OTF2_EvtReaderCallbacks_SetRmaOpTestCallback, 390
OTF2_EvtReaderCallbacks_SetRmaOpTestCallback, 402
OTF2_EvtReaderCallbacks_SetRmaPostCallback, 390
OTF2_EvtReaderCallbacks_Clear, 390
OTF2_EvtReaderCallbacks.h, 370
OTF2_EvtReaderCallbacks_SetRmaPostCallback, 391
OTF2_EvtReaderCallbacks_Delete, 391
OTF2_EvtReaderCallbacks.h, 370
OTF2_EvtReaderCallbacks_SetRmaPostCallback, 391
OTF2_EvtReaderCallbacks_New, 391
OTF2_EvtReaderCallbacks.h, 370
OTF2_EvtReaderCallbacks_SetRmaSyncCallback, 392
OTF2_EvtReaderCallbacks_SetBufferFlushCallback, 392
OTF2_EvtReaderCallbacks.h, 370

OTF2_EvtReaderCallbacks_SetEnterCallback 371
 OTF2_EvtReaderCallbacks_SetLeaveCallback 372
 OTF2_EvtReaderCallbacks_SetMeasurementCallback 372
 OTF2_EvtReaderCallbacks_SetMetricCallback 373
 OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback 373
 OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback 374
 OTF2_EvtReaderCallbacks_SetMpiRecvCallback 374
 OTF2_EvtReaderCallbacks_SetMpiRecvReadyCallback 375
 OTF2_EvtReaderCallbacks_SetMpiSendCallback 375
 OTF2_EvtReaderCallbacks_SetMpiSendCompleteCallback 376
 OTF2_EvtReaderCallbacks_SetMpiRecvCompleteCallback 377
 OTF2_EvtReaderCallbacks_SetMpiRequestCancelCallback 377
 OTF2_EvtReaderCallbacks_SetMpiRequestCompleteCallback 378
 OTF2_EvtReaderCallbacks_SetMpiSendCompleteCallback 378
 OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback 379
 OTF2_EvtReaderCallbacks_SetOmpForkCallback 379
 OTF2_EvtReaderCallbacks_SetOmpJoinCallback 380
 OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback 380
 OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback 381
 OTF2_EvtReaderCallbacks_SetOmpTaskWaitCallback 382
 OTF2_EvtReaderCallbacks_SetOmpTaskWaitEndCallback 382
 OTF2_EvtReaderCallbacks_SetParameterStringCallback 383
 OTF2_EvtReaderCallbacks_SetParameterUnsignedIntCallback 384
 OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback 384
 OTF2_EvtReaderCallbacks_SetRmaAtomicCallback 385
 OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback 386
 OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback 386
 OTF2_EvtReaderCallbacks_SetRmaGetCallback 387
 OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback 387
 OTF2_EvtReaderCallbacks_SetRmaOpCompleteBlockingCallback 388
 OTF2_EvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback 388
 OTF2_EvtReaderCallbacks_SetRmaOpCompleteRemoteCallback 389
 OTF2_EvtReaderCallbacks_SetRmaOpTestCallback 390
 OTF2_EvtReaderCallbacks_SetRmaPutCallback 390
 OTF2_EvtReaderCallbacks_SetRmaReleaseLockCallback 391
 OTF2_EvtReaderCallbacks_SetRmaRequestLockCallback 391
 OTF2_EvtReaderCallbacks_SetRmaSyncCallback 392
 OTF2_EvtReaderCallbacks_SetRmaTryLockCallback 392
 OTF2_EvtReaderCallbacks_SetRmaWaitChangeCallback 393
 OTF2_EvtReaderCallbacks_SetRmaWinCreateCallback 394
 OTF2_EvtReaderCallbacks_SetRmaWinDestroyCallback 394
 OTF2_EvtReaderCallbacks_SetThreadAcquireLockCallback 395
 OTF2_EvtReaderCallbacks_SetThreadBeginCallback 395

INDEX

OTF2_EvtReaderCallbacks_SetThreadCreateCallback, 416
OTF2_EvtReaderCallbacks.h, 396
OTF2_EvtReaderCallbacks_SetThreadEndCallback, 417
OTF2_EvtReaderCallbacks.h, 396
OTF2_EvtReaderCallbacks_SetThreadForkCallback, 418
OTF2_EvtReaderCallbacks.h, 397
OTF2_EvtReaderCallbacks_SetThreadJoinCallback, 419
OTF2_EvtReaderCallbacks.h, 397
OTF2_EvtReaderCallbacks_SetThreadReleaseLockCallback, 420
OTF2_EvtReaderCallbacks.h, 398
OTF2_EvtReaderCallbacks_SetThreadTaskCompleteCallback, 421
OTF2_EvtReaderCallbacks.h, 398
OTF2_EvtReaderCallbacks_SetThreadTaskCreateCallback, 422
OTF2_EvtReaderCallbacks.h, 399
OTF2_EvtReaderCallbacks_SetThreadTaskSwitchCallback, 423
OTF2_EvtReaderCallbacks.h, 400
OTF2_EvtReaderCallbacks_SetThreadTeamBeginCallback, 423
OTF2_EvtReaderCallbacks.h, 400
OTF2_EvtReaderCallbacks_SetThreadTeamEndCallback, 424
OTF2_EvtReaderCallbacks.h, 401
OTF2_EvtReaderCallbacks_SetThreadWaitCallback, 425
OTF2_EvtReaderCallbacks.h, 401
OTF2_EvtReaderCallbacks_SetUnknownCallback, 426
OTF2_EvtReaderCallbacks.h, 402
OTF2_EvtWriter.h
OTF2_EvtWriter_BufferFlush, 409
OTF2_EvtWriter_ClearRewindPoint, 410
OTF2_EvtWriter_Enter, 410
OTF2_EvtWriter_GetLocationID, 411
OTF2_EvtWriter_GetNumberOfEvents, 411
OTF2_EvtWriter_GetUserData, 412
OTF2_EvtWriter_Leave, 412
OTF2_EvtWriter_MeasurementOnOff, 412
OTF2_EvtWriter_Metric, 413
OTF2_EvtWriter_MpiCollectiveBegin, 414
OTF2_EvtWriter_MpiCollectiveEnd, 414
OTF2_EvtWriter_MpiIrecv, 415
OTF2_EvtWriter_MpiIrecvRequest, 416
OTF2_EvtWriter_MpiIsend, 416
OTF2_EvtWriter_MpiIsendComplete, 417
OTF2_EvtWriter_MpiRecv, 417
OTF2_EvtWriter_MpiRequestCancelled, 418
OTF2_EvtWriter_MpiRequestTest, 419
OTF2_EvtWriter_MpiSend, 419
OTF2_EvtWriter_OmpAcquireLock, 420
OTF2_EvtWriter_OmpFork, 421
OTF2_EvtWriter_OmpJoin, 421
OTF2_EvtWriter_OmpReleaseLock, 422
OTF2_EvtWriter_OmpTaskComplete, 423
OTF2_EvtWriter_OmpTaskCreate, 423
OTF2_EvtWriter_OmpTaskSwitch, 424
OTF2_EvtWriter_ParameterInt, 425
OTF2_EvtWriter_ParameterString, 425
OTF2_EvtWriter_ParameterUnsignedInt, 426
OTF2_EvtWriter_Rewind, 427
OTF2_EvtWriter_RmaAcquireLock, 427
OTF2_EvtWriter_RmaAtomic, 428
OTF2_EvtWriter_RmaCollectiveBegin, 429
OTF2_EvtWriter_RmaCollectiveEnd, 429
OTF2_EvtWriter_RmaGet, 430
OTF2_EvtWriter_RmaGroupSync, 430
OTF2_EvtWriter_RmaOpCompleteBlocking, 431
OTF2_EvtWriter_RmaOpCompleteNonBlocking, 432
OTF2_EvtWriter_RmaOpCompleteRemote, 432
OTF2_EvtWriter_RmaOpTest, 433
OTF2_EvtWriter_RmaPut, 434
OTF2_EvtWriter_RmaReleaseLock, 434

OTF2_EvtWriter_RmaRequestLock, OTF2_EvtWriter.h, 412
 435 OTF2_EvtWriter_Leave
 OTF2_EvtWriter_RmaSync, 436 OTF2_EvtWriter.h, 412
 OTF2_EvtWriter_RmaTryLock, 436 OTF2_EvtWriter_MeasurementOnOff
 OTF2_EvtWriter_RmaWaitChange, OTF2_EvtWriter.h, 412
 437 OTF2_EvtWriter_Metric
 OTF2_EvtWriter_RmaWinCreate, 437 OTF2_EvtWriter.h, 413
 OTF2_EvtWriter_RmaWinDestroy, OTF2_EvtWriter_MpiCollectiveBegin
 438 OTF2_EvtWriter.h, 414
 OTF2_EvtWriter_SetLocationID, 439 OTF2_EvtWriter_MpiCollectiveEnd
 OTF2_EvtWriter_SetUserData, 439 OTF2_EvtWriter.h, 414
 OTF2_EvtWriter_StoreRewindPoint, OTF2_EvtWriter_MpiIrecv
 439 OTF2_EvtWriter.h, 415
 OTF2_EvtWriter_ThreadAcquireLock, OTF2_EvtWriter_MpiIrecvRequest
 440 OTF2_EvtWriter.h, 416
 OTF2_EvtWriter_ThreadBegin, 440 OTF2_EvtWriter_MpiIsend
 OTF2_EvtWriter_ThreadCreate, 441 OTF2_EvtWriter.h, 416
 OTF2_EvtWriter_ThreadEnd, 441 OTF2_EvtWriter_MpiIsendComplete
 OTF2_EvtWriter_ThreadFork, 442 OTF2_EvtWriter.h, 417
 OTF2_EvtWriter_ThreadJoin, 443 OTF2_EvtWriter_MpiRecv
 OTF2_EvtWriter_ThreadReleaseLock, OTF2_EvtWriter.h, 417
 443 OTF2_EvtWriter_MpiRequestCancelled
 OTF2_EvtWriter_ThreadTaskComplete, OTF2_EvtWriter.h, 418
 444 OTF2_EvtWriter_MpiRequestTest
 OTF2_EvtWriter_ThreadTaskCreate, OTF2_EvtWriter.h, 419
 444 OTF2_EvtWriter_MpiSend
 OTF2_EvtWriter_ThreadTaskSwitch, OTF2_EvtWriter.h, 419
 445 OTF2_EvtWriter_OmpAcquireLock
 OTF2_EvtWriter_ThreadTeamBegin, OTF2_EvtWriter.h, 420
 446 OTF2_EvtWriter_OmpFork
 OTF2_EvtWriter_ThreadTeamEnd, OTF2_EvtWriter.h, 421
 446 OTF2_EvtWriter_OmpJoin
 OTF2_EvtWriter_ThreadWait, 447 OTF2_EvtWriter.h, 421
 OTF2_EvtWriter_BufferFlush OTF2_EvtWriter_OmpReleaseLock
 OTF2_EvtWriter.h, 409 OTF2_EvtWriter.h, 422
 OTF2_EvtWriter_ClearRewindPoint OTF2_EvtWriter_OmpTaskComplete
 OTF2_EvtWriter.h, 410 OTF2_EvtWriter.h, 423
 OTF2_EvtWriter_Enter OTF2_EvtWriter_OmpTaskCreate
 OTF2_EvtWriter.h, 410 OTF2_EvtWriter.h, 423
 OTF2_EvtWriter_GetLocationID OTF2_EvtWriter_OmpTaskSwitch
 OTF2_EvtWriter.h, 411 OTF2_EvtWriter.h, 424
 OTF2_EvtWriter_GetNumberOfEvents OTF2_EvtWriter_ParameterInt
 OTF2_EvtWriter.h, 411 OTF2_EvtWriter.h, 425
 OTF2_EvtWriter_GetUserData OTF2_EvtWriter_ParameterString

INDEX

OTF2_EvtWriter.h, [425](#)
OTF2_EvtWriter_ParameterUnsignedInt
OTF2_EvtWriter.h, [426](#)
OTF2_EvtWriter_Rewind
OTF2_EvtWriter.h, [427](#)
OTF2_EvtWriter_RmaAcquireLock
OTF2_EvtWriter.h, [427](#)
OTF2_EvtWriter_RmaAtomic
OTF2_EvtWriter.h, [428](#)
OTF2_EvtWriter_RmaCollectiveBegin
OTF2_EvtWriter.h, [429](#)
OTF2_EvtWriter_RmaCollectiveEnd
OTF2_EvtWriter.h, [429](#)
OTF2_EvtWriter_RmaGet
OTF2_EvtWriter.h, [430](#)
OTF2_EvtWriter_RmaGroupSync
OTF2_EvtWriter.h, [430](#)
OTF2_EvtWriter_RmaOpCompleteBlocking
OTF2_EvtWriter.h, [431](#)
OTF2_EvtWriter_RmaOpCompleteNonBlocking
OTF2_EvtWriter.h, [432](#)
OTF2_EvtWriter_RmaOpCompleteRemote
OTF2_EvtWriter.h, [432](#)
OTF2_EvtWriter_RmaOpTest
OTF2_EvtWriter.h, [433](#)
OTF2_EvtWriter_RmaPut
OTF2_EvtWriter.h, [434](#)
OTF2_EvtWriter_RmaReleaseLock
OTF2_EvtWriter.h, [434](#)
OTF2_EvtWriter_RmaRequestLock
OTF2_EvtWriter.h, [435](#)
OTF2_EvtWriter_RmaSync
OTF2_EvtWriter.h, [436](#)
OTF2_EvtWriter_RmaTryLock
OTF2_EvtWriter.h, [436](#)
OTF2_EvtWriter_RmaWaitChange
OTF2_EvtWriter.h, [437](#)
OTF2_EvtWriter_RmaWinCreate
OTF2_EvtWriter.h, [437](#)
OTF2_EvtWriter_RmaWinDestroy
OTF2_EvtWriter.h, [438](#)
OTF2_EvtWriter_SetLocationID
OTF2_EvtWriter.h, [439](#)
OTF2_EvtWriter_SetUserData
OTF2_EvtWriter.h, [439](#)
OTF2_EvtWriter_StoreRewindPoint
OTF2_EvtWriter.h, [439](#)
OTF2_EvtWriter_ThreadAcquireLock
OTF2_EvtWriter.h, [440](#)
OTF2_EvtWriter_ThreadBegin
OTF2_EvtWriter.h, [440](#)
OTF2_EvtWriter_ThreadCreate
OTF2_EvtWriter.h, [441](#)
OTF2_EvtWriter_ThreadEnd
OTF2_EvtWriter.h, [441](#)
OTF2_EvtWriter_ThreadFork
OTF2_EvtWriter.h, [442](#)
OTF2_EvtWriter_ThreadJoin
OTF2_EvtWriter.h, [443](#)
OTF2_EvtWriter_ThreadReleaseLock
OTF2_EvtWriter.h, [443](#)
OTF2_EvtWriter_ThreadTaskComplete
OTF2_EvtWriter.h, [444](#)
OTF2_EvtWriter_ThreadTaskCreate
OTF2_EvtWriter.h, [444](#)
OTF2_EvtWriter_ThreadTaskSwitch
OTF2_EvtWriter.h, [445](#)
OTF2_EvtWriter_ThreadTeamBegin
OTF2_EvtWriter.h, [446](#)
OTF2_EvtWriter_ThreadTeamEnd
OTF2_EvtWriter.h, [446](#)
OTF2_EvtWriter_ThreadWait
OTF2_EvtWriter.h, [447](#)
OTF2_FileMode_enum
OTF2_GeneralDefinitions.h, [456](#)
OTF2_FileSubstrate_enum
OTF2_GeneralDefinitions.h, [456](#)
OTF2_FileType_enum
OTF2_GeneralDefinitions.h, [457](#)
OTF2_FlushCallbacks, [123](#)
OTF2_FlushType_enum
OTF2_GeneralDefinitions.h, [457](#)
OTF2_GeneralDefinitions.h
OTF2_CallbackCode, [455](#)
OTF2_Compression_enum, [456](#)
OTF2_FileMode_enum, [456](#)
OTF2_FileSubstrate_enum, [456](#)
OTF2_FileType_enum, [457](#)

OTF2_FlushType_enum, [457](#)
 OTF2_MappingType_enum, [458](#)
 OTF2_Paradigm_enum, [458](#)
 OTF2_ThumbnailType_enum, [459](#)
 OTF2_Type_enum, [460](#)
 OTF2_GlobalDefReader.h
 OTF2_GlobalDefReader_ReadDefinitions, [462](#)
 OTF2_GlobalDefReader_SetCallbacks, [462](#)
 OTF2_GlobalDefReader_ReadDefinitions
 OTF2_GlobalDefReader.h, [462](#)
 OTF2_GlobalDefReader_SetCallbacks
 OTF2_GlobalDefReader.h, [462](#)
 OTF2_GlobalDefReaderCallback_Attribute
 OTF2_GlobalDefReaderCallbacks.h, [469](#)
 OTF2_GlobalDefReaderCallback_Callpath
 OTF2_GlobalDefReaderCallbacks.h, [470](#)
 OTF2_GlobalDefReaderCallback_Callsite
 OTF2_GlobalDefReaderCallbacks.h, [470](#)
 OTF2_GlobalDefReaderCallback_CartCoordinate
 OTF2_GlobalDefReaderCallbacks.h, [471](#)
 OTF2_GlobalDefReaderCallback_CartDimension
 OTF2_GlobalDefReaderCallbacks.h, [472](#)
 OTF2_GlobalDefReaderCallback_CartTopology
 OTF2_GlobalDefReaderCallbacks.h, [472](#)
 OTF2_GlobalDefReaderCallback_ClockProperties
 OTF2_GlobalDefReaderCallbacks.h, [473](#)
 OTF2_GlobalDefReaderCallback_Comm
 OTF2_GlobalDefReaderCallbacks.h, [474](#)
 OTF2_GlobalDefReaderCallback_Group
 OTF2_GlobalDefReaderCallbacks.h, [475](#)
 OTF2_GlobalDefReaderCallback_Location
 OTF2_GlobalDefReaderCallbacks.h, [475](#)
 OTF2_GlobalDefReaderCallback_LocationGroup
 OTF2_GlobalDefReaderCallbacks.h, [476](#)
 OTF2_GlobalDefReaderCallback_LocationGroupProperty
 OTF2_GlobalDefReaderCallbacks.h, [477](#)
 OTF2_GlobalDefReaderCallback_LocationProperty
 OTF2_GlobalDefReaderCallbacks.h, [477](#)
 OTF2_GlobalDefReaderCallback_MetricClass
 OTF2_GlobalDefReaderCallbacks.h, [478](#)
 OTF2_GlobalDefReaderCallback_MetricClassRecorder
 OTF2_GlobalDefReaderCallbacks.h, [479](#)
 OTF2_GlobalDefReaderCallback_MetricInstance
 OTF2_GlobalDefReaderCallbacks.h, [479](#)
 OTF2_GlobalDefReaderCallback_MetricMember
 OTF2_GlobalDefReaderCallbacks.h, [480](#)
 OTF2_GlobalDefReaderCallback_Parameter
 OTF2_GlobalDefReaderCallbacks.h, [481](#)
 OTF2_GlobalDefReaderCallback_Region
 OTF2_GlobalDefReaderCallbacks.h, [482](#)
 OTF2_GlobalDefReaderCallback_RmaWin
 OTF2_GlobalDefReaderCallbacks.h, [483](#)
 OTF2_GlobalDefReaderCallback_String
 OTF2_GlobalDefReaderCallbacks.h, [483](#)
 OTF2_GlobalDefReaderCallback_SystemTreeNode
 OTF2_GlobalDefReaderCallbacks.h, [484](#)
 OTF2_GlobalDefReaderCallback_SystemTreeNodeDomain
 OTF2_GlobalDefReaderCallbacks.h, [484](#)
 OTF2_GlobalDefReaderCallback_SystemTreeNodeProperty
 OTF2_GlobalDefReaderCallbacks.h, [485](#)
 OTF2_GlobalDefReaderCallback_Unknown

INDEX

- OTF2_GlobalDefReaderCallbacks.h, [486](#)
- OTF2_GlobalDefReaderCallbacks.h
 - OTF2_GlobalDefReaderCallback_-Attribute, [469](#)
 - OTF2_GlobalDefReaderCallback_-Callpath, [470](#)
 - OTF2_GlobalDefReaderCallback_-Callsite, [470](#)
 - OTF2_GlobalDefReaderCallback_-CartCoordinate, [471](#)
 - OTF2_GlobalDefReaderCallback_-CartDimension, [472](#)
 - OTF2_GlobalDefReaderCallback_-CartTopology, [472](#)
 - OTF2_GlobalDefReaderCallback_-ClockProperties, [473](#)
 - OTF2_GlobalDefReaderCallback_-Comm, [474](#)
 - OTF2_GlobalDefReaderCallback_-Group, [475](#)
 - OTF2_GlobalDefReaderCallback_-Location, [475](#)
 - OTF2_GlobalDefReaderCallback_-LocationGroup, [476](#)
 - OTF2_GlobalDefReaderCallback_-LocationGroupProperty, [477](#)
 - OTF2_GlobalDefReaderCallback_-LocationProperty, [477](#)
 - OTF2_GlobalDefReaderCallback_-MetricClass, [478](#)
 - OTF2_GlobalDefReaderCallback_-MetricClassRecorder, [479](#)
 - OTF2_GlobalDefReaderCallback_-MetricInstance, [479](#)
 - OTF2_GlobalDefReaderCallback_-MetricMember, [480](#)
 - OTF2_GlobalDefReaderCallback_-Parameter, [481](#)
 - OTF2_GlobalDefReaderCallback_-Region, [482](#)
 - OTF2_GlobalDefReaderCallback_-RmaWin, [483](#)
 - OTF2_GlobalDefReaderCallback_-String, [483](#)
 - OTF2_GlobalDefReaderCallback_-SystemTreeNode, [484](#)
 - OTF2_GlobalDefReaderCallback_-SystemTreeNodeDomain, [484](#)
 - OTF2_GlobalDefReaderCallback_-SystemTreeNodeProperty, [485](#)
 - OTF2_GlobalDefReaderCallback_-Unknown, [486](#)
 - OTF2_GlobalDefReaderCallbacks_-Clear, [486](#)
 - OTF2_GlobalDefReaderCallbacks_-Delete, [486](#)
 - OTF2_GlobalDefReaderCallbacks_-New, [487](#)
 - OTF2_GlobalDefReaderCallbacks_-SetAttributeCallback, [487](#)
 - OTF2_GlobalDefReaderCallbacks_-SetCallpathCallback, [487](#)
 - OTF2_GlobalDefReaderCallbacks_-SetCallsiteCallback, [488](#)
 - OTF2_GlobalDefReaderCallbacks_-SetCartCoordinateCallback, [489](#)
 - OTF2_GlobalDefReaderCallbacks_-SetCartDimensionCallback, [489](#)
 - OTF2_GlobalDefReaderCallbacks_-SetCartTopologyCallback, [490](#)
 - OTF2_GlobalDefReaderCallbacks_-SetClockPropertiesCallback, [490](#)
 - OTF2_GlobalDefReaderCallbacks_-SetCommCallback, [491](#)
 - OTF2_GlobalDefReaderCallbacks_-SetGroupCallback, [491](#)
 - OTF2_GlobalDefReaderCallbacks_-SetLocationCallback, [492](#)
 - OTF2_GlobalDefReaderCallbacks_-SetLocationGroupCallback, [493](#)
 - OTF2_GlobalDefReaderCallbacks_-SetLocationGroupPropertyCallback, [493](#)
 - OTF2_GlobalDefReaderCallbacks_-SetLocationPropertyCallback, [494](#)

INDEX

OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks.h,
 SetMetricClassCallback, [494](#) [488](#)
 OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetCartCoordinateCallback
 SetMetricClassRecorderCallback, OTF2_GlobalDefReaderCallbacks.h,
 [495](#) [489](#)
 OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetCartDimensionCallback
 SetMetricInstanceCallback, [496](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_- [489](#)
 SetMetricMemberCallback, [496](#) OTF2_GlobalDefReaderCallbacks_SetCartTopologyCallback
 OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks.h,
 SetParameterCallback, [497](#) [490](#)
 OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetClockPropertiesCallback
 SetRegionCallback, [497](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_- [490](#)
 SetRmaWinCallback, [498](#) OTF2_GlobalDefReaderCallbacks_SetCommCallback
 OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks.h,
 SetStringCallback, [499](#) [491](#)
 OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetGroupCallback
 SetSystemTreeNodeCallback, [499](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_- [491](#)
 SetSystemTreeNodeDomainCallback, OTF2_GlobalDefReaderCallbacks_SetLocationCallback
 [500](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_- [492](#)
 SetSystemTreeNodePropertyCallback, OTF2_GlobalDefReaderCallbacks_SetLocationGroupCallback
 [500](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_- [493](#)
 SetUnknownCallback, [501](#) OTF2_GlobalDefReaderCallbacks_SetLocationGroupPropertyCallback
 OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_Clear [493](#)
 OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetLocationPropertyCallback
 [486](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_Delete [494](#)
 OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetMetricClassCallback
 [486](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_New [494](#)
 OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetMetricClassRecorderCallback
 [487](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_SetAttributeCallback [495](#)
 OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetMetricInstanceCallback
 [487](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_SetCallpathCallback [496](#)
 OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetMetricMemberCallback
 [487](#) OTF2_GlobalDefReaderCallbacks.h,
 OTF2_GlobalDefReaderCallbacks_SetCallsiteCallback [496](#)

INDEX

OTF2_GlobalDefReaderCallbacks_SetParameterCallback, 510
OTF2_GlobalDefReaderCallbacks.h, 497
OTF2_GlobalDefReaderCallbacks_SetRegionCallback, 511
OTF2_GlobalDefReaderCallbacks.h, 497
OTF2_GlobalDefReaderCallbacks_SetRmaWinCallback, 513
OTF2_GlobalDefReaderCallbacks.h, 498
OTF2_GlobalDefReaderCallbacks_SetStringCallback, 513
OTF2_GlobalDefReaderCallbacks.h, 499
OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeCallback, 514
OTF2_GlobalDefReaderCallbacks.h, 499
OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeDomainCallback, 515
OTF2_GlobalDefReaderCallbacks.h, 500
OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodePropertyCallback, 516
OTF2_GlobalDefReaderCallbacks.h, 500
OTF2_GlobalDefReaderCallbacks_SetUnknownCallback, 518
OTF2_GlobalDefReaderCallbacks.h, 501
OTF2_GlobalDefWriter.h, 519
OTF2_GlobalDefWriter_GetNumberOfDefinitions, 505
OTF2_GlobalDefWriter_GetNumberOfLocations, 506
OTF2_GlobalDefWriter_WriteAttribute, 506
OTF2_GlobalDefWriter_WriteCallpath, 507
OTF2_GlobalDefWriter_WriteCallsite, 507
OTF2_GlobalDefWriter_WriteCartCoordinate, 508
OTF2_GlobalDefWriter_WriteCartDirection, 509
OTF2_GlobalDefWriter_WriteCartTopology, 509
OTF2_GlobalDefWriter_WriteClockPart, 510
OTF2_GlobalDefWriter_WriteComm, 510
OTF2_GlobalDefWriter_WriteGroup, 511
OTF2_GlobalDefWriter_WriteLocation, 512
OTF2_GlobalDefWriter_WriteLocationGroup, 513
OTF2_GlobalDefWriter_WriteLocationGroupProperty, 513
OTF2_GlobalDefWriter_WriteLocationProperty, 514
OTF2_GlobalDefWriter_WriteMetricClass, 514
OTF2_GlobalDefWriter_WriteMetricClassRecorder, 515
OTF2_GlobalDefWriter_WriteMetricInstance, 516
OTF2_GlobalDefWriter_WriteMetricMember, 516
OTF2_GlobalDefWriter_WriteParameter, 518
OTF2_GlobalDefWriter_WriteRegion, 518
OTF2_GlobalDefWriter_WriteRmaWin, 519
OTF2_GlobalDefWriter_WriteString, 520
OTF2_GlobalDefWriter_WriteSystemTreeNode, 520
OTF2_GlobalDefWriter_WriteSystemTreeNodeDomain, 521
OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty, 521
OTF2_GlobalDefWriter_GetNumberOfDefinitions, 505
OTF2_GlobalDefWriter.h, 505
OTF2_GlobalDefWriter_GetNumberOfLocations, 506
OTF2_GlobalDefWriter.h, 506
OTF2_GlobalDefWriter_WriteAttribute, 506
OTF2_GlobalDefWriter.h, 506
OTF2_GlobalDefWriter_WriteCallpath, 507
OTF2_GlobalDefWriter.h, 507
OTF2_GlobalDefWriter_WriteCallsite, 507
OTF2_GlobalDefWriter.h, 507

OTF2_GlobalDefWriter_WriteCartCoordinate OTF2_GlobalEvtReader_HasEvent,
 OTF2_GlobalDefWriter.h, 508 523
 OTF2_GlobalDefWriter_WriteCartDimension OTF2_GlobalEvtReader_ReadEvent,
 OTF2_GlobalDefWriter.h, 509 523
 OTF2_GlobalDefWriter_WriteCartTopology OTF2_GlobalEvtReader_ReadEvents,
 OTF2_GlobalDefWriter.h, 509 524
 OTF2_GlobalDefWriter_WriteClockProperties OTF2_GlobalEvtReader_SetCallbacks,
 OTF2_GlobalDefWriter.h, 510 524
 OTF2_GlobalDefWriter_WriteComm OTF2_GlobalEvtReader_HasEvent
 OTF2_GlobalDefWriter.h, 510 OTF2_GlobalEvtReader.h, 523
 OTF2_GlobalDefWriter_WriteGroup OTF2_GlobalEvtReader_ReadEvent
 OTF2_GlobalDefWriter.h, 511 OTF2_GlobalEvtReader.h, 523
 OTF2_GlobalDefWriter_WriteLocation OTF2_GlobalEvtReader_ReadEvents
 OTF2_GlobalDefWriter.h, 512 OTF2_GlobalEvtReader.h, 524
 OTF2_GlobalDefWriter_WriteLocationGroup OTF2_GlobalEvtReader_SetCallbacks
 OTF2_GlobalDefWriter.h, 513 OTF2_GlobalEvtReader.h, 524
 OTF2_GlobalDefWriter_WriteLocationGroupProperty OTF2_GlobalEvtReaderCallback_BufferFlush
 OTF2_GlobalDefWriter.h, 513 OTF2_GlobalEvtReaderCallbacks.h,
 538
 OTF2_GlobalDefWriter_WriteLocationProperty OTF2_GlobalEvtReaderCallback_Enter
 OTF2_GlobalDefWriter.h, 514 OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter_WriteMetricClass 538
 OTF2_GlobalDefWriter.h, 514 OTF2_GlobalEvtReaderCallback_Leave
 OTF2_GlobalDefWriter_WriteMetricClassRecorder OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter.h, 515 539
 OTF2_GlobalDefWriter_WriteMetricInstance OTF2_GlobalEvtReaderCallback_MeasurementOnOff
 OTF2_GlobalDefWriter.h, 516 OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter_WriteMetricMember 540
 OTF2_GlobalDefWriter.h, 516 OTF2_GlobalEvtReaderCallback_Metric
 OTF2_GlobalDefWriter_WriteParameter OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter.h, 518 540
 OTF2_GlobalDefWriter_WriteRegion OTF2_GlobalEvtReaderCallback_MpiCollectiveBegin
 OTF2_GlobalDefWriter.h, 518 OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter_WriteRmaWin 541
 OTF2_GlobalDefWriter.h, 519 OTF2_GlobalEvtReaderCallback_MpiCollectiveEnd
 OTF2_GlobalDefWriter_WriteString OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter.h, 520 542
 OTF2_GlobalDefWriter_WriteSystemTreeNode OTF2_GlobalEvtReaderCallback_MpiRecv
 OTF2_GlobalDefWriter.h, 520 OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter_WriteSystemTreeNodeDone 541
 OTF2_GlobalDefWriter.h, 521 OTF2_GlobalEvtReaderCallback_MpiRecvRequest
 OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter.h, 521 543
 OTF2_GlobalEvtReader.h OTF2_GlobalEvtReaderCallback_MpiIsend

INDEX

OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_ParameterUnsignedInt
544 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_MpiIsendComplete 554
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaAcquireLock
545 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_MpiRecv 555
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaAtomic
545 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_MpiRequestCancelled 555
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaCollectiveBegin
546 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_MpiRequestTest 556
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaCollectiveEnd
547 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_MpiSend 557
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaGet
547 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_OmpAcquireLock 558
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaGroupSync
548 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_OmpFork 558
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaOpCompleteBlocking
549 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_OmpJoin 559
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaOpCompleteNonBlocking
549 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_OmpReleaseLock 560
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaOpCompleteRemote
550 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_OmpTaskComplete 561
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaOpTest
551 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_OmpTaskCreate 561
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaPut
551 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_OmpTaskSwitch 562
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaReleaseLock
552 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_ParameterInt 563
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaRequestLock
553 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_ParameterStatus 563
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_RmaSync
553

OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_ThreadTeamBegin
564 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_RmaTryLock 574
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_ThreadTeamEnd
565 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_RmaWaitChange 575
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_ThreadWait
565 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_RmaWinCreate 575
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_Unknown
566 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallback_RmaWinDestroy 576
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks.h
567 OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallback_ThreadAcquireLock, BufferFlush, 538
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_
567 Enter, 538
OTF2_GlobalEvtReaderCallback_ThreadBegin, OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallbacks.h, Leave, 539
568 OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallback_ThreadCreate, MeasurementOnOff, 540
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_
569 Metric, 540
OTF2_GlobalEvtReaderCallback_ThreadEnd, OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallbacks.h, MpiCollectiveBegin, 541
569 OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallback_ThreadFork, MpiCollectiveEnd, 542
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_
570 MpiIrecv, 542
OTF2_GlobalEvtReaderCallback_ThreadJoin, OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallbacks.h, MpiIrecvRequest, 543
571 OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallback_ThreadReleaseLock, MpiIsend, 544
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_
571 MpiIsendComplete, 545
OTF2_GlobalEvtReaderCallback_ThreadTaskComplete, OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallbacks.h, MpiRecv, 545
572 OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallback_ThreadTaskCreate, MpiRequestCancelled, 546
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallback_
573 MpiRequestTest, 547
OTF2_GlobalEvtReaderCallback_ThreadTaskSwitch, OTF2_GlobalEvtReaderCallback_
OTF2_GlobalEvtReaderCallbacks.h, MpiSend, 547
573

INDEX

- OTF2_GlobalEvtReaderCallback_-
OmpAcquireLock, [548](#)
- OTF2_GlobalEvtReaderCallback_-
OmpFork, [549](#)
- OTF2_GlobalEvtReaderCallback_-
OmpJoin, [549](#)
- OTF2_GlobalEvtReaderCallback_-
OmpReleaseLock, [550](#)
- OTF2_GlobalEvtReaderCallback_-
OmpTaskComplete, [551](#)
- OTF2_GlobalEvtReaderCallback_-
OmpTaskCreate, [551](#)
- OTF2_GlobalEvtReaderCallback_-
OmpTaskSwitch, [552](#)
- OTF2_GlobalEvtReaderCallback_-
ParameterInt, [553](#)
- OTF2_GlobalEvtReaderCallback_-
ParameterString, [553](#)
- OTF2_GlobalEvtReaderCallback_-
ParameterUnsignedInt, [554](#)
- OTF2_GlobalEvtReaderCallback_-
RmaAcquireLock, [555](#)
- OTF2_GlobalEvtReaderCallback_-
RmaAtomic, [555](#)
- OTF2_GlobalEvtReaderCallback_-
RmaCollectiveBegin, [556](#)
- OTF2_GlobalEvtReaderCallback_-
RmaCollectiveEnd, [557](#)
- OTF2_GlobalEvtReaderCallback_-
RmaGet, [558](#)
- OTF2_GlobalEvtReaderCallback_-
RmaGroupSync, [558](#)
- OTF2_GlobalEvtReaderCallback_-
RmaOpCompleteBlocking, [559](#)
- OTF2_GlobalEvtReaderCallback_-
RmaOpCompleteNonBlocking,
[560](#)
- OTF2_GlobalEvtReaderCallback_-
RmaOpCompleteRemote, [561](#)
- OTF2_GlobalEvtReaderCallback_-
RmaOpTest, [561](#)
- OTF2_GlobalEvtReaderCallback_-
RmaPut, [562](#)
- OTF2_GlobalEvtReaderCallback_-
RmaReleaseLock, [563](#)
- OTF2_GlobalEvtReaderCallback_-
RmaRequestLock, [563](#)
- OTF2_GlobalEvtReaderCallback_-
RmaSync, [564](#)
- OTF2_GlobalEvtReaderCallback_-
RmaTryLock, [565](#)
- OTF2_GlobalEvtReaderCallback_-
RmaWaitChange, [565](#)
- OTF2_GlobalEvtReaderCallback_-
RmaWinCreate, [566](#)
- OTF2_GlobalEvtReaderCallback_-
RmaWinDestroy, [567](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadAcquireLock, [567](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadBegin, [568](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadCreate, [569](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadEnd, [569](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadFork, [570](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadJoin, [571](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadReleaseLock, [571](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadTaskComplete, [572](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadTaskCreate, [573](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadTaskSwitch, [573](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadTeamBegin, [574](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadTeamEnd, [575](#)
- OTF2_GlobalEvtReaderCallback_-
ThreadWait, [575](#)
- OTF2_GlobalEvtReaderCallback_-
Unknown, [576](#)
- OTF2_GlobalEvtReaderCallbacks_-
Clear, [577](#)

- OTF2_GlobalEvtReaderCallbacks_-
Delete, [577](#)
- OTF2_GlobalEvtReaderCallbacks_-
New, [577](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetBufferFlushCallback, [577](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetEnterCallback, [578](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetLeaveCallback, [579](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMeasurementOnOffCallback,
[579](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMetricCallback, [580](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiCollectiveBeginCallback,
[580](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiCollectiveEndCallback,
[581](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiIrecvCallback, [582](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiIrecvRequestCallback, [582](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiIsendCallback, [583](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiIsendCompleteCallback,
[583](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiRecvCallback, [584](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiRequestCancelledCallback,
[585](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiRequestTestCallback, [585](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetMpiSendCallback, [586](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetOmpAcquireLockCallback,
[586](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetOmpForkCallback, [587](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetOmpJoinCallback, [588](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetOmpReleaseLockCallback, [588](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetOmpTaskCompleteCallback,
[589](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetOmpTaskCreateCallback, [589](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetOmpTaskSwitchCallback, [590](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetParameterIntCallback, [590](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetParameterStringCallback, [591](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetParameterUnsignedIntCallback,
[592](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaAcquireLockCallback, [592](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaAtomicCallback, [593](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaCollectiveBeginCallback,
[593](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaCollectiveEndCallback,
[594](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaGetCallback, [595](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaGroupSyncCallback, [595](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpCompleteBlockingCallback,
[596](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpCompleteNonBlockingCallback,
[597](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpCompleteRemoteCallback,
[597](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpTestCallback, [598](#)

INDEX

- OTF2_GlobalEvtReaderCallbacks_-
SetRmaPutCallback, [598](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaReleaseLockCallback, [599](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaRequestLockCallback, [599](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaSyncCallback, [600](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaTryLockCallback, [601](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaWaitChangeCallback, [601](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaWinCreateCallback, [602](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaWinDestroyCallback, [602](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadAcquireLockCallback, [603](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadBeginCallback, [604](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadCreateCallback, [604](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadEndCallback, [605](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadForkCallback, [605](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadJoinCallback, [606](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadReleaseLockCallback, [607](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTaskCompleteCallback, [607](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTaskCreateCallback, [608](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTaskSwitchCallback, [608](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTeamBeginCallback, [609](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTeamEndCallback, [610](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadWaitCallback, [610](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetUnknownCallback, [611](#)
- OTF2_GlobalEvtReaderCallbacks_Clear
OTF2_GlobalEvtReaderCallbacks.h,
[577](#)
- OTF2_GlobalEvtReaderCallbacks_Delete
OTF2_GlobalEvtReaderCallbacks.h,
[577](#)
- OTF2_GlobalEvtReaderCallbacks_New
OTF2_GlobalEvtReaderCallbacks.h,
[577](#)
- OTF2_GlobalEvtReaderCallbacks_SetBufferFlushCallback
OTF2_GlobalEvtReaderCallbacks.h,
[577](#)
- OTF2_GlobalEvtReaderCallbacks_SetEnterCallback
OTF2_GlobalEvtReaderCallbacks.h,
[578](#)
- OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback
OTF2_GlobalEvtReaderCallbacks.h,
[579](#)
- OTF2_GlobalEvtReaderCallbacks_SetMeasurementOnOffCallback
OTF2_GlobalEvtReaderCallbacks.h,
[579](#)
- OTF2_GlobalEvtReaderCallbacks_SetMetricCallback
OTF2_GlobalEvtReaderCallbacks.h,
[580](#)
- OTF2_GlobalEvtReaderCallbacks_SetMpiCollectiveBeginCallback
OTF2_GlobalEvtReaderCallbacks.h,
[580](#)
- OTF2_GlobalEvtReaderCallbacks_SetMpiCollectiveEndCallback
OTF2_GlobalEvtReaderCallbacks.h,
[581](#)
- OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvCallback
OTF2_GlobalEvtReaderCallbacks.h,
[582](#)
- OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvRequestCallback
OTF2_GlobalEvtReaderCallbacks.h,
[582](#)
- OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCallback

INDEX

OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetParameterUnsignedIntCallback
583 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCompleteCallback
592 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaAcquireLockCallback
583 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiRecvCallback
592 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaAtomicCallback
584 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiRequestCancelledCallback
593 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveBeginCallback
585 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiRequestTestCallback
593 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveEndCallback
585 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiSendCallback
594 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaGetCallback
586 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetOmpAcquireLockCallback
595 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaGroupSyncCallback
586 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetOmpForkCallback
595 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteBlockingCallback
587 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetOmpJoinCallback
596 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback
588 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetOmpReleaseLockCallback
597 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteRemoteCallback
588 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCompleteCallback
597 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaOpTestCallback
589 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCreateCallback
598 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaPutCallback
589 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaRequestLockCallback
598 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaReleaseLockCallback
590 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetParameterIntCallback
599 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetRmaRequestLockCallback
590 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetParameterStringCallback
599 OTF2_GlobalEvtReaderCallbacks.h,
591 OTF2_GlobalEvtReaderCallbacks_SetRmaSyncCallback

INDEX

OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadTeamBeginCallback
600 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaTryLockCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadTeamEndCallback
601 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaWaitChangeCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadWaitCallback
601 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaWinCreateCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetUnknownCallback
602 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaWinDestroyCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReader.h
602 OTF2_GlobalSnapReader_ReadSnapshots,
OTF2_GlobalEvtReaderCallbacks_SetThreadAcquireLockCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReader_SetCallbacks,
603 OTF2_GlobalSnapReader_ReadSnapshots
OTF2_GlobalEvtReaderCallbacks_SetThreadBeginCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReader.h, 612
604 OTF2_GlobalSnapReader_SetCallbacks
OTF2_GlobalEvtReaderCallbacks_SetThreadCreateCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReader.h, 613
604 OTF2_GlobalSnapReaderCallback_Enter
OTF2_GlobalEvtReaderCallbacks_SetThreadEndCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallbacks.h,
605 OTF2_GlobalSnapReaderCallback_MeasurementOnOff
OTF2_GlobalEvtReaderCallbacks_SetThreadForkCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallbacks.h,
605 OTF2_GlobalSnapReaderCallback_Metric
OTF2_GlobalEvtReaderCallbacks_SetThreadJoinCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallbacks.h,
606 OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin
OTF2_GlobalEvtReaderCallbacks_SetThreadReleaseLockCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallbacks.h,
607 OTF2_GlobalSnapReaderCallback_MpiCollectiveEnd
OTF2_GlobalEvtReaderCallbacks_SetThreadTaskCompleteCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallbacks.h,
607 OTF2_GlobalSnapReaderCallback_MpiIrecv
OTF2_GlobalEvtReaderCallbacks_SetThreadTaskCreateCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallbacks.h,
608 OTF2_GlobalSnapReaderCallback_MpiIrecvRequest
OTF2_GlobalEvtReaderCallbacks_SetThreadTaskWaitCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallbacks.h,
608 OTF2_GlobalSnapReaderCallback_MpiIsend

[OTF2_GlobalSnapReaderCallbacks.h](#)
[625](#)

[OTF2_GlobalSnapReaderCallback_MpiIsendComplete](#), [619](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [OTF2_GlobalSnapReaderCallback_-](#)
[626](#) [MeasurementOnOff](#), [620](#)

[OTF2_GlobalSnapReaderCallback_MpiRecv](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [Metric](#), [620](#)
[626](#)

[OTF2_GlobalSnapReaderCallback_MpiSend](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [MpiCollectiveBegin](#), [621](#)
[627](#) [OTF2_GlobalSnapReaderCallback_-](#)
[MpiCollectiveEnd](#), [622](#)

[OTF2_GlobalSnapReaderCallback_OmpAcquireLock](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [MpiIrecv](#), [623](#)
[628](#)

[OTF2_GlobalSnapReaderCallback_OmpFork](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [MpiIrecvRequest](#), [624](#)
[629](#) [OTF2_GlobalSnapReaderCallback_-](#)
[MpiIsend](#), [625](#)

[OTF2_GlobalSnapReaderCallback_OmpTaskCreate](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [MpiIsendComplete](#), [626](#)
[629](#)

[OTF2_GlobalSnapReaderCallback_OmpTaskSwitch](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [MpiRecv](#), [626](#)
[630](#) [OTF2_GlobalSnapReaderCallback_-](#)
[MpiSend](#), [627](#)

[OTF2_GlobalSnapReaderCallback_ParameterInt](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [OmpAcquireLock](#), [628](#)
[631](#)

[OTF2_GlobalSnapReaderCallback_ParameterString](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [OmpFork](#), [629](#)
[632](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OmpTaskCreate](#), [629](#)

[OTF2_GlobalSnapReaderCallback_ParameterUnsignedInt](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [OmpTaskSwitch](#), [630](#)
[632](#)

[OTF2_GlobalSnapReaderCallback_SnapshotEnd](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [ParameterInt](#), [631](#)
[633](#) [OTF2_GlobalSnapReaderCallback_-](#)
[ParameterString](#), [632](#)

[OTF2_GlobalSnapReaderCallback_SnapshotStart](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [ParameterUnsignedInt](#), [632](#)
[634](#)

[OTF2_GlobalSnapReaderCallback_Unknown](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [SnapshotEnd](#), [633](#)
[635](#) [OTF2_GlobalSnapReaderCallback_-](#)
[SnapshotStart](#), [634](#)

[OTF2_GlobalSnapReaderCallbacks](#) [OTF2_GlobalSnapReaderCallback_-](#)
[OTF2_GlobalSnapReaderCallbacks.h](#), [Unknown](#), [635](#)
[635](#)

INDEX

- OTF2_GlobalSnapReaderCallbacks, 635
- OTF2_GlobalSnapReaderCallbacks_- Clear, 636
- OTF2_GlobalSnapReaderCallbacks_- Delete, 636
- OTF2_GlobalSnapReaderCallbacks_- New, 636
- OTF2_GlobalSnapReaderCallbacks_- SetEnterCallback, 637
- OTF2_GlobalSnapReaderCallbacks_- SetMeasurementOnOffCallback, 637
- OTF2_GlobalSnapReaderCallbacks_- SetMetricCallback, 638
- OTF2_GlobalSnapReaderCallbacks_- SetMpiCollectiveBeginCallback, 638
- OTF2_GlobalSnapReaderCallbacks_- SetMpiCollectiveEndCallback, 639
- OTF2_GlobalSnapReaderCallbacks_- SetMpiIrecvCallback, 639
- OTF2_GlobalSnapReaderCallbacks_- SetMpiIrecvRequestCallback, 640
- OTF2_GlobalSnapReaderCallbacks_- SetMpiIsendCallback, 641
- OTF2_GlobalSnapReaderCallbacks_- SetMpiIsendCompleteCallback, 641
- OTF2_GlobalSnapReaderCallbacks_- SetMpiRecvCallback, 642
- OTF2_GlobalSnapReaderCallbacks_- SetMpiSendCallback, 642
- OTF2_GlobalSnapReaderCallbacks_- SetOmpAcquireLockCallback, 643
- OTF2_GlobalSnapReaderCallbacks_- SetOmpForkCallback, 643
- OTF2_GlobalSnapReaderCallbacks_- SetOmpTaskCreateCallback, 644
- OTF2_GlobalSnapReaderCallbacks_- SetOmpTaskSwitchCallback, 645
- OTF2_GlobalSnapReaderCallbacks_- SetParameterIntCallback, 645
- OTF2_GlobalSnapReaderCallbacks_- SetParameterStringCallback, 646
- OTF2_GlobalSnapReaderCallbacks_- SetParameterUnsignedIntCallback, 646
- OTF2_GlobalSnapReaderCallbacks_- SetSnapshotEndCallback, 647
- OTF2_GlobalSnapReaderCallbacks_- SetSnapshotStartCallback, 648
- OTF2_GlobalSnapReaderCallbacks_- SetUnknownCallback, 648
- OTF2_GlobalSnapReaderCallbacks_Clear
- OTF2_GlobalSnapReaderCallbacks.h, 636
- OTF2_GlobalSnapReaderCallbacks_Delete
- OTF2_GlobalSnapReaderCallbacks.h, 636
- OTF2_GlobalSnapReaderCallbacks_New
- OTF2_GlobalSnapReaderCallbacks.h, 636
- OTF2_GlobalSnapReaderCallbacks_SetEnterCallback
- OTF2_GlobalSnapReaderCallbacks.h, 637
- OTF2_GlobalSnapReaderCallbacks_SetMeasurementOnOffCallback
- OTF2_GlobalSnapReaderCallbacks.h, 637
- OTF2_GlobalSnapReaderCallbacks_SetMetricCallback
- OTF2_GlobalSnapReaderCallbacks.h, 638
- OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveBeginCallback
- OTF2_GlobalSnapReaderCallbacks.h, 638
- OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveEndCallback
- OTF2_GlobalSnapReaderCallbacks.h, 639
- OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvCallback
- OTF2_GlobalSnapReaderCallbacks.h, 639
- OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvRequestCallback
- OTF2_GlobalSnapReaderCallbacks.h, 640
- OTF2_GlobalSnapReaderCallbacks_SetMpiIsendCallback

OTF2_GlobalSnapReaderCallbacks.h, OTF2_Definitions.h, 201
 641 OTF2_IdMap
 OTF2_GlobalSnapReaderCallbacks_SetMpiReaderCallback, 650
 OTF2_GlobalSnapReaderCallbacks.h OTF2_IdMap.h
 641 OTF2_IdMap, 650
 OTF2_GlobalSnapReaderCallbacks_SetMpiReaderCallback, 651
 OTF2_GlobalSnapReaderCallbacks.h, OTF2_IdMap_Clear, 651
 642 OTF2_IdMap_Create, 652
 OTF2_GlobalSnapReaderCallbacks_SetMpiReaderCallback, 652
 OTF2_GlobalSnapReaderCallbacks.h, 652
 642 OTF2_IdMap_CreateFromUint32Array,
 OTF2_GlobalSnapReaderCallbacks_SetOmpAcquireLockCallback, 652
 OTF2_GlobalSnapReaderCallbacks.h, OTF2_IdMap_Free, 653
 643 OTF2_IdMap_GetGlobalId, 653
 OTF2_GlobalSnapReaderCallbacks_SetOmpForAllCallback, 654
 OTF2_GlobalSnapReaderCallbacks.h, OTF2_IdMap_GetGlobalIdSave, 654
 643 OTF2_IdMap_GetMode, 654
 OTF2_IdMap_GetSize, 654
 OTF2_GlobalSnapReaderCallbacks_SetOmpTaskCallback, 655
 OTF2_GlobalSnapReaderCallbacks.h, OTF2_IdMapMode, 650
 644 OTF2_IdMapMode_enum, 651
 OTF2_GlobalSnapReaderCallbacks_SetOmpTaskCallback, 651
 OTF2_GlobalSnapReaderCallbacks.h, OTF2_IdMap.h, 651
 645 OTF2_IdMap_Clear
 OTF2_GlobalSnapReaderCallbacks_SetParameterCallback, 651
 OTF2_GlobalSnapReaderCallbacks.h OTF2_IdMap_Create
 645 OTF2_IdMap.h, 652
 OTF2_GlobalSnapReaderCallbacks_SetParameterCallback, 652
 OTF2_GlobalSnapReaderCallbacks.h, OTF2_IdMap.h, 652
 646 OTF2_IdMap_CreateFromUint32Array
 OTF2_GlobalSnapReaderCallbacks_SetParameterCallback, 653
 OTF2_GlobalSnapReaderCallbacks.h OTF2_IdMap_Free
 646 OTF2_IdMap.h, 653
 OTF2_GlobalSnapReaderCallbacks_SetSnapshotsCallback, 654
 OTF2_GlobalSnapReaderCallbacks.h, OTF2_IdMap.h, 653
 647 OTF2_IdMap_GetGlobalId
 OTF2_GlobalSnapReaderCallbacks_SetSnapshotsCallback, 654
 OTF2_GlobalSnapReaderCallbacks.h OTF2_IdMap_GetMode
 648 OTF2_IdMap.h, 654
 OTF2_GlobalSnapReaderCallbacks_SetUnpackCallback, 654
 OTF2_GlobalSnapReaderCallbacks.h, OTF2_IdMap.h, 654
 648 OTF2_IdMap_Traverse
 OTF2_GroupFlag_enum OTF2_IdMap.h, 655
 OTF2_Definitions.h, 200 OTF2_IdMapMode
 OTF2_GroupType_enum OTF2_IdMap.h, 650

INDEX

- OTF2_IdMapMode_enum
 - OTF2_IdMap.h, [651](#)
- OTF2_LocationGroupType_enum
 - OTF2_Definitions.h, [202](#)
- OTF2_LocationType_enum
 - OTF2_Definitions.h, [202](#)
- OTF2_LockType_enum
 - OTF2_Events.h, [273](#)
- OTF2_MappingType_enum
 - OTF2_GeneralDefinitions.h, [458](#)
- OTF2_Marker.h
 - OTF2_MarkerScope_enum, [656](#)
 - OTF2_MarkerSeverity_enum, [657](#)
- OTF2_MarkerReader.h
 - OTF2_MarkerReader_ReadMarkers, [658](#)
 - OTF2_MarkerReader_SetCallbacks, [658](#)
- OTF2_MarkerReader_ReadMarkers
 - OTF2_MarkerReader.h, [658](#)
- OTF2_MarkerReader_SetCallbacks
 - OTF2_MarkerReader.h, [658](#)
- OTF2_MarkerReaderCallback_DefMarker
 - OTF2_MarkerReaderCallbacks.h, [661](#)
- OTF2_MarkerReaderCallback_Marker
 - OTF2_MarkerReaderCallbacks.h, [661](#)
- OTF2_MarkerReaderCallback_Unknown
 - OTF2_MarkerReaderCallbacks.h, [662](#)
- OTF2_MarkerReaderCallbacks.h
 - OTF2_MarkerReaderCallback_DefMarker, [661](#)
 - OTF2_MarkerReaderCallback_Marker, [661](#)
 - OTF2_MarkerReaderCallback_Unknown, [662](#)
 - OTF2_MarkerReaderCallbacks_Clear, [662](#)
 - OTF2_MarkerReaderCallbacks_Delete, [662](#)
 - OTF2_MarkerReaderCallbacks_New, [663](#)
 - OTF2_MarkerReaderCallbacks_SetDefMarkerCallback, [663](#)
- OTF2_MarkerReaderCallbacks_SetMarkerCallback,
 - [664](#)
- OTF2_MarkerReaderCallbacks_SetUnknownCallback,
 - [664](#)
- OTF2_MarkerReaderCallbacks_Clear
 - OTF2_MarkerReaderCallbacks.h, [662](#)
- OTF2_MarkerReaderCallbacks_Delete
 - OTF2_MarkerReaderCallbacks.h, [662](#)
- OTF2_MarkerReaderCallbacks_New
 - OTF2_MarkerReaderCallbacks.h, [663](#)
- OTF2_MarkerReaderCallbacks_SetDefMarkerCallback
 - OTF2_MarkerReaderCallbacks.h, [663](#)
- OTF2_MarkerReaderCallbacks_SetMarkerCallback
 - OTF2_MarkerReaderCallbacks.h, [664](#)
- OTF2_MarkerReaderCallbacks_SetUnknownCallback
 - OTF2_MarkerReaderCallbacks.h, [664](#)
- OTF2_MarkerScope_enum
 - OTF2_Marker.h, [656](#)
- OTF2_MarkerSeverity_enum
 - OTF2_Marker.h, [657](#)
- OTF2_MarkerWriter.h
 - OTF2_MarkerWriter_WriteDefMarker, [666](#)
 - OTF2_MarkerWriter_WriteMarker, [666](#)
- OTF2_MarkerWriter_WriteDefMarker
 - OTF2_MarkerWriter.h, [666](#)
- OTF2_MarkerWriter_WriteMarker
 - OTF2_MarkerWriter.h, [666](#)
- OTF2_MeasurementMode_enum
 - OTF2_Events.h, [273](#)
- OTF2_MemoryAllocate
 - Memory pooling for OTF2, [94](#)
- OTF2_MemoryCallbacks,
 - [124](#)
- OTF2_MemoryFreeAll
 - Memory pooling for OTF2, [94](#)
- OTF2_MetricBase_enum
 - OTF2_Definitions.h, [202](#)
- OTF2_MetricMode_enum
 - OTF2_Definitions.h, [203](#)
- OTF2_MetricOccurrence_enum
 - OTF2_Definitions.h, [203](#)
- OTF2_MarkerMode_enum
 - OTF2_Definitions.h, [204](#)

- OTF2_MetricTiming_enum
 - OTF2_Definitions.h, [204](#)
- OTF2_MetricType_enum
 - OTF2_Definitions.h, [205](#)
- OTF2_MetricValue_union, [125](#)
- OTF2_MetricValueProperty_enum
 - OTF2_Definitions.h, [205](#)
- OTF2_MPI_Archive_SetCollectiveCallbacks
 - OTF2_MPI_Collectives.h, [669](#)
- OTF2_MPI_Archive_SetCollectiveCallbacksSplit
 - OTF2_MPI_Collectives.h, [669](#)
- OTF2_MPI_Collectives.h
 - OTF2_MPI_Archive_SetCollectiveCallbacks, [669](#)
 - OTF2_MPI_Archive_SetCollectiveCallbacksSplit, [669](#)
 - OTF2_MPI_Reader_SetCollectiveCallbacks, [670](#)
- OTF2_MPI_Reader_SetCollectiveCallbacks
 - OTF2_MPI_Collectives.h, [670](#)
- OTF2_MPI_UserData, [125](#)
- OTF2_Paradigm_enum
 - OTF2_GeneralDefinitions.h, [458](#)
- OTF2_ParameterType_enum
 - OTF2_Definitions.h, [206](#)
- OTF2_PostFlushCallback
 - Controlling OTF2 flush behavior in writing mode, [92](#)
- OTF2_PreFlushCallback
 - Controlling OTF2 flush behavior in writing mode, [93](#)
- OTF2_Reader.h
 - OTF2_Reader_Close, [676](#)
 - OTF2_Reader_CloseDefFiles, [676](#)
 - OTF2_Reader_CloseDefReader, [677](#)
 - OTF2_Reader_CloseEvtFiles, [677](#)
 - OTF2_Reader_CloseEvtReader, [677](#)
 - OTF2_Reader_CloseGlobalDefReader, [678](#)
 - OTF2_Reader_CloseGlobalEvtReader, [678](#)
 - OTF2_Reader_CloseGlobalSnapReader, [679](#)
 - OTF2_Reader_CloseMarkerReader, [679](#)
 - OTF2_Reader_CloseMarkerWriter, [679](#)
 - OTF2_Reader_CloseSnapFiles, [680](#)
 - OTF2_Reader_CloseSnapReader, [680](#)
 - OTF2_Reader_CloseThumbReader, [681](#)
 - OTF2_Reader_GetBoolProperty, [681](#)
 - OTF2_Reader_GetChunkSize, [682](#)
 - OTF2_Reader_GetCompression, [682](#)
 - OTF2_Reader_GetCreator, [682](#)
 - OTF2_Reader_GetDefReader, [683](#)
 - OTF2_Reader_GetDescription, [683](#)
 - OTF2_Reader_GetEvtReader, [683](#)
 - OTF2_Reader_GetFileSubstrate, [684](#)
 - OTF2_Reader_GetGlobalDefReader, [684](#)
 - OTF2_Reader_GetGlobalEvtReader, [684](#)
 - OTF2_Reader_GetGlobalSnapReader, [685](#)
 - OTF2_Reader_GetMachineName, [685](#)
 - OTF2_Reader_GetMarkerReader, [685](#)
 - OTF2_Reader_GetMarkerWriter, [686](#)
 - OTF2_Reader_GetNumberOfGlobalDefinitions, [686](#)
 - OTF2_Reader_GetNumberOfLocations, [687](#)
 - OTF2_Reader_GetNumberOfSnapshots, [687](#)
 - OTF2_Reader_GetNumberOfThumbnails, [687](#)
 - OTF2_Reader_GetProperty, [688](#)
 - OTF2_Reader_GetPropertyNames, [688](#)
 - OTF2_Reader_GetSnapReader, [689](#)
 - OTF2_Reader_GetThumbReader, [689](#)
 - OTF2_Reader_GetTraceId, [689](#)
 - OTF2_Reader_GetVersion, [690](#)
 - OTF2_Reader_HasGlobalEvent, [690](#)
 - OTF2_Reader_Open, [691](#)
 - OTF2_Reader_OpenDefFiles, [691](#)
 - OTF2_Reader_OpenEvtFiles, [691](#)
 - OTF2_Reader_OpenSnapFiles, [692](#)

INDEX

OTF2_Reader_ReadAllGlobalDefinitions, OTF2_Reader_SetSerialCollectiveCallbacks,
692 704
OTF2_Reader_ReadAllGlobalEvents, OTF2_Reader_Close
693 OTF2_Reader.h, 676
OTF2_Reader_ReadAllGlobalSnapshots, OTF2_Reader_CloseDefFiles
693 OTF2_Reader.h, 676
OTF2_Reader_ReadAllLocalDefinitions, OTF2_Reader_CloseDefReader
693 OTF2_Reader.h, 677
OTF2_Reader_ReadAllLocalEvents, OTF2_Reader_CloseEvtFiles
694 OTF2_Reader.h, 677
OTF2_Reader_ReadAllLocalSnapshots, OTF2_Reader_CloseEvtReader
694 OTF2_Reader.h, 677
OTF2_Reader_ReadAllMarkers, 695 OTF2_Reader_CloseGlobalDefReader
OTF2_Reader_ReadGlobalDefinitions, OTF2_Reader.h, 678
695 OTF2_Reader_CloseGlobalEvtReader
OTF2_Reader_ReadGlobalEvent, 696 OTF2_Reader.h, 678
OTF2_Reader_ReadGlobalEvents, 696 OTF2_Reader_CloseGlobalSnapReader
OTF2_Reader_ReadGlobalSnapshots, OTF2_Reader.h, 679
696 OTF2_Reader_CloseMarkerReader
OTF2_Reader_ReadLocalDefinitions, OTF2_Reader.h, 679
697 OTF2_Reader_CloseMarkerWriter
OTF2_Reader_ReadLocalEvents, 697 OTF2_Reader.h, 679
OTF2_Reader_ReadLocalEventsBackward, OTF2_Reader_CloseSnapFiles
698 OTF2_Reader.h, 680
OTF2_Reader_ReadLocalSnapshots, OTF2_Reader_CloseSnapReader
698 OTF2_Reader.h, 680
OTF2_Reader_ReadMarkers, 699 OTF2_Reader_CloseThumbReader
OTF2_Reader_RegisterDefCallbacks, OTF2_Reader.h, 681
699 OTF2_Reader_GetBoolProperty
OTF2_Reader_RegisterEvtCallbacks, OTF2_Reader.h, 681
700 OTF2_Reader_GetChunkSize
OTF2_Reader_RegisterGlobalDefCallbacks, OTF2_Reader.h, 682
700 OTF2_Reader_GetCompression
OTF2_Reader_RegisterGlobalEvtCallbacks, OTF2_Reader.h, 682
701 OTF2_Reader_GetCreator
OTF2_Reader_RegisterGlobalSnapCallbacks, OTF2_Reader.h, 682
701 OTF2_Reader_GetDefReader
OTF2_Reader_RegisterMarkerCallbacks, OTF2_Reader.h, 683
702 OTF2_Reader_GetDescription
OTF2_Reader_RegisterSnapCallbacks, OTF2_Reader.h, 683
702 OTF2_Reader_GetEvtReader
OTF2_Reader_SelectLocation, 703 OTF2_Reader.h, 683
OTF2_Reader_SetCollectiveCallbacks, OTF2_Reader_GetFileSubstrate
703 OTF2_Reader.h, 684

OTF2_Reader_GetGlobalDefReader	OTF2_Reader_ReadAllGlobalEvents
OTF2_Reader.h, 684	OTF2_Reader.h, 693
OTF2_Reader_GetGlobalEvtReader	OTF2_Reader_ReadAllGlobalSnapshots
OTF2_Reader.h, 684	OTF2_Reader.h, 693
OTF2_Reader_GetGlobalSnapReader	OTF2_Reader_ReadAllLocalDefinitions
OTF2_Reader.h, 685	OTF2_Reader.h, 693
OTF2_Reader_GetMachineName	OTF2_Reader_ReadAllLocalEvents
OTF2_Reader.h, 685	OTF2_Reader.h, 694
OTF2_Reader_GetMarkerReader	OTF2_Reader_ReadAllLocalSnapshots
OTF2_Reader.h, 685	OTF2_Reader.h, 694
OTF2_Reader_GetMarkerWriter	OTF2_Reader_ReadAllMarkers
OTF2_Reader.h, 686	OTF2_Reader.h, 695
OTF2_Reader_GetNumberOfGlobalDefinitions	OTF2_Reader_ReadGlobalDefinitions
OTF2_Reader.h, 686	OTF2_Reader.h, 695
OTF2_Reader_GetNumberOfLocations	OTF2_Reader_ReadGlobalEvent
OTF2_Reader.h, 687	OTF2_Reader.h, 696
OTF2_Reader_GetNumberOfSnapshots	OTF2_Reader_ReadGlobalEvents
OTF2_Reader.h, 687	OTF2_Reader.h, 696
OTF2_Reader_GetNumberOfThumbnails	OTF2_Reader_ReadGlobalSnapshots
OTF2_Reader.h, 687	OTF2_Reader.h, 696
OTF2_Reader_GetProperty	OTF2_Reader_ReadLocalDefinitions
OTF2_Reader.h, 688	OTF2_Reader.h, 697
OTF2_Reader_GetPropertyNames	OTF2_Reader_ReadLocalEvents
OTF2_Reader.h, 688	OTF2_Reader.h, 697
OTF2_Reader_GetSnapReader	OTF2_Reader_ReadLocalEventsBackward
OTF2_Reader.h, 689	OTF2_Reader.h, 698
OTF2_Reader_GetThumbReader	OTF2_Reader_ReadLocalSnapshots
OTF2_Reader.h, 689	OTF2_Reader.h, 698
OTF2_Reader_GetTraceId	OTF2_Reader_ReadMarkers
OTF2_Reader.h, 689	OTF2_Reader.h, 699
OTF2_Reader_GetVersion	OTF2_Reader_RegisterDefCallbacks
OTF2_Reader.h, 690	OTF2_Reader.h, 699
OTF2_Reader_HasGlobalEvent	OTF2_Reader_RegisterEvtCallbacks
OTF2_Reader.h, 690	OTF2_Reader.h, 700
OTF2_Reader_Open	OTF2_Reader_RegisterGlobalDefCallbacks
OTF2_Reader.h, 691	OTF2_Reader.h, 700
OTF2_Reader_OpenDefFiles	OTF2_Reader_RegisterGlobalEvtCallbacks
OTF2_Reader.h, 691	OTF2_Reader.h, 701
OTF2_Reader_OpenEvtFiles	OTF2_Reader_RegisterGlobalSnapCallbacks
OTF2_Reader.h, 691	OTF2_Reader.h, 701
OTF2_Reader_OpenSnapFiles	OTF2_Reader_RegisterMarkerCallbacks
OTF2_Reader.h, 692	OTF2_Reader.h, 702
OTF2_Reader_ReadAllGlobalDefinitions	OTF2_Reader_RegisterSnapCallbacks
OTF2_Reader.h, 692	OTF2_Reader.h, 702

INDEX

OTF2_Reader_SelectLocation
 OTF2_Reader.h, [703](#)
OTF2_Reader_SetCollectiveCallbacks
 OTF2_Reader.h, [703](#)
OTF2_Reader_SetSerialCollectiveCallbacks
 OTF2_Reader.h, [704](#)
OTF2_RecorderKind_enum
 OTF2_Definitions.h, [206](#)
OTF2_RegionFlag_enum
 OTF2_Definitions.h, [206](#)
OTF2_RegionRole_enum
 OTF2_Definitions.h, [207](#)
OTF2_RmaAtomicType_enum
 OTF2_Events.h, [274](#)
OTF2_RmaSyncLevel_enum
 OTF2_Events.h, [275](#)
OTF2_RmaSyncType_enum
 OTF2_Events.h, [275](#)
OTF2_SnapReader.h
 OTF2_SnapReader_GetLocationID,
 [705](#)
 OTF2_SnapReader_ReadSnapshots,
 [705](#)
 OTF2_SnapReader_Seek, [706](#)
 OTF2_SnapReader_SetCallbacks, [706](#)
OTF2_SnapReader_GetLocationID
 OTF2_SnapReader.h, [705](#)
OTF2_SnapReader_ReadSnapshots
 OTF2_SnapReader.h, [705](#)
OTF2_SnapReader_Seek
 OTF2_SnapReader.h, [706](#)
OTF2_SnapReader_SetCallbacks
 OTF2_SnapReader.h, [706](#)
OTF2_SnapReaderCallback_Enter
 OTF2_SnapReaderCallbacks.h, [713](#)
OTF2_SnapReaderCallback_MeasurementOnOff
 OTF2_SnapReaderCallbacks.h, [713](#)
OTF2_SnapReaderCallback_Metric
 OTF2_SnapReaderCallbacks.h, [714](#)
OTF2_SnapReaderCallback_MpiCollectiveBegin
 OTF2_SnapReaderCallbacks.h, [715](#)
OTF2_SnapReaderCallback_MpiCollectiveEnd
 OTF2_SnapReaderCallbacks.h, [715](#)
OTF2_SnapReaderCallback_MpiIrecv
 OTF2_SnapReaderCallbacks.h, [716](#)
OTF2_SnapReaderCallback_MpiIrecvRequest
 OTF2_SnapReaderCallbacks.h, [717](#)
OTF2_SnapReaderCallback_MpiIsend
 OTF2_SnapReaderCallbacks.h, [718](#)
OTF2_SnapReaderCallback_MpiIsendComplete
 OTF2_SnapReaderCallbacks.h, [719](#)
OTF2_SnapReaderCallback_MpiRecv
 OTF2_SnapReaderCallbacks.h, [720](#)
OTF2_SnapReaderCallback_MpiSend
 OTF2_SnapReaderCallbacks.h, [721](#)
OTF2_SnapReaderCallback_OmpAcquireLock
 OTF2_SnapReaderCallbacks.h, [721](#)
OTF2_SnapReaderCallback_OmpFork
 OTF2_SnapReaderCallbacks.h, [722](#)
OTF2_SnapReaderCallback_OmpTaskCreate
 OTF2_SnapReaderCallbacks.h, [723](#)
OTF2_SnapReaderCallback_OmpTaskSwitch
 OTF2_SnapReaderCallbacks.h, [724](#)
OTF2_SnapReaderCallback_ParameterInt
 OTF2_SnapReaderCallbacks.h, [724](#)
OTF2_SnapReaderCallback_ParameterString
 OTF2_SnapReaderCallbacks.h, [725](#)
OTF2_SnapReaderCallback_ParameterUnsignedInt
 OTF2_SnapReaderCallbacks.h, [726](#)
OTF2_SnapReaderCallback_SnapshotEnd
 OTF2_SnapReaderCallbacks.h, [727](#)
OTF2_SnapReaderCallback_SnapshotStart
 OTF2_SnapReaderCallbacks.h, [727](#)
OTF2_SnapReaderCallback_Unknown
 OTF2_SnapReaderCallbacks.h, [728](#)
OTF2_SnapReaderCallbacks
 OTF2_SnapReaderCallbacks.h, [728](#)
OTF2_SnapReaderCallbacks.h
 OTF2_SnapReaderCallback_Enter, [713](#)
 OTF2_SnapReaderCallback_MeasurementOnOff,
 [713](#)
 OTF2_SnapReaderCallback_Metric,
 [714](#)
 OTF2_SnapReaderCallback_MpiCollectiveBegin, [714](#)
 OTF2_SnapReaderCallback_MpiCollectiveEnd, [715](#)
 OTF2_SnapReaderCallback_MpiIrecv, [715](#)

OTF2_SnapReaderCallback_MpiIrecv, 716
 OTF2_SnapReaderCallback_MpiIrecvRequest, 717
 OTF2_SnapReaderCallback_MpiIsend, 718
 OTF2_SnapReaderCallback_MpiIsendComplete, 719
 OTF2_SnapReaderCallback_MpiRecv, 720
 OTF2_SnapReaderCallback_MpiSend, 721
 OTF2_SnapReaderCallback_OmpAcquireLock, 721
 OTF2_SnapReaderCallback_OmpFork, 722
 OTF2_SnapReaderCallback_OmpTaskCreate, 723
 OTF2_SnapReaderCallback_OmpTaskSwitch, 724
 OTF2_SnapReaderCallback_ParameterInt, 724
 OTF2_SnapReaderCallback_ParameterString, 725
 OTF2_SnapReaderCallback_ParameterUnsignedInt, 726
 OTF2_SnapReaderCallback_SnapshotEnd, 727
 OTF2_SnapReaderCallback_SnapshotStart, 727
 OTF2_SnapReaderCallback_Unknown, 728
 OTF2_SnapReaderCallbacks, 728
 OTF2_SnapReaderCallbacks_Clear, 729
 OTF2_SnapReaderCallbacks_Delete, 729
 OTF2_SnapReaderCallbacks_New, 729
 OTF2_SnapReaderCallbacks_SetEntryPoint, 729
 OTF2_SnapReaderCallbacks_SetMetricCallback, 731
 OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback, 731
 OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback, 732
 OTF2_SnapReaderCallbacks_SetMpiIrecvCallback, 732
 OTF2_SnapReaderCallbacks_SetMpiIrecvRequestCallback, 733
 OTF2_SnapReaderCallbacks_SetMpiIsendCallback, 733
 OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback, 734
 OTF2_SnapReaderCallbacks_SetMpiRecvCallback, 735
 OTF2_SnapReaderCallbacks_SetMpiSendCallback, 735
 OTF2_SnapReaderCallbacks_SetOmpAcquireLockCallback, 736
 OTF2_SnapReaderCallbacks_SetOmpForkCallback, 736
 OTF2_SnapReaderCallbacks_SetOmpTaskCreateCallback, 737
 OTF2_SnapReaderCallbacks_SetOmpTaskSwitchCallback, 737
 OTF2_SnapReaderCallbacks_SetParameterIntCallback, 738
 OTF2_SnapReaderCallbacks_SetParameterStringCallback, 739
 OTF2_SnapReaderCallbacks_SetParameterUnsignedIntCallback, 739
 OTF2_SnapReaderCallbacks_SetSnapshotEndCallback, 740
 OTF2_SnapReaderCallbacks_SetSnapshotStartCallback, 740
 OTF2_SnapReaderCallbacks_SetUnknownCallback, 741
 OTF2_SnapReaderCallbacks_Clear, 729
 OTF2_SnapReaderCallbacks_Delete, 729
 OTF2_SnapReaderCallbacks_New, 729
 OTF2_SnapReaderCallbacks_SetEntryPoint, 729
 OTF2_SnapReaderCallbacks_SetMetricCallback, 731
 OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback, 731
 OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback, 732
 OTF2_SnapReaderCallbacks_SetMpiIrecvCallback, 732
 OTF2_SnapReaderCallbacks_SetMpiIrecvRequestCallback, 733
 OTF2_SnapReaderCallbacks_SetMpiIsendCallback, 733
 OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback, 734
 OTF2_SnapReaderCallbacks_SetMpiRecvCallback, 735
 OTF2_SnapReaderCallbacks_SetMpiSendCallback, 735
 OTF2_SnapReaderCallbacks_SetOmpAcquireLockCallback, 736
 OTF2_SnapReaderCallbacks_SetOmpForkCallback, 736
 OTF2_SnapReaderCallbacks_SetOmpTaskCreateCallback, 737
 OTF2_SnapReaderCallbacks_SetOmpTaskSwitchCallback, 737
 OTF2_SnapReaderCallbacks_SetParameterIntCallback, 738
 OTF2_SnapReaderCallbacks_SetParameterStringCallback, 739
 OTF2_SnapReaderCallbacks_SetParameterUnsignedIntCallback, 739
 OTF2_SnapReaderCallbacks_SetSnapshotEndCallback, 740
 OTF2_SnapReaderCallbacks_SetSnapshotStartCallback, 740
 OTF2_SnapReaderCallbacks_SetUnknownCallback, 741
 OTF2_SnapReaderCallbacks.h, 729

INDEX

OTF2_SnapReaderCallbacks_SetEnterCallback, 729
OTF2_SnapReaderCallbacks_SetMeasurementCallback, 730
OTF2_SnapReaderCallbacks_SetMetricCallback, 731
OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback, 731
OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback, 732
OTF2_SnapReaderCallbacks_SetMpiIrecvRequestCallback, 733
OTF2_SnapReaderCallbacks_SetMpiIsendCallback, 733
OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback, 734
OTF2_SnapReaderCallbacks_SetMpiRecvCallback, 735
OTF2_SnapReaderCallbacks_SetMpiSendCallback, 735
OTF2_SnapReaderCallbacks_SetOmpAcquireLockCallback, 736
OTF2_SnapReaderCallbacks_SetOmpForkCallback, 736
OTF2_SnapReaderCallbacks_SetOmpTaskCreateCallback, 737
OTF2_SnapReaderCallbacks_SetOmpTaskSwitchCallback, 737
OTF2_SnapReaderCallbacks_SetParameterIntCallback, 738
OTF2_SnapReaderCallbacks_SetParameterStringCallback, 739
OTF2_SnapReaderCallbacks_SetParameterUnsignedIntCallback, 739
OTF2_SnapReaderCallbacks_SetSnapshotStartCallback, 740
OTF2_SnapReaderCallbacks_SetSnapshotEndCallback, 740
OTF2_SnapReaderCallbacks_SetUnknownCallback, 741
OTF2_SnapWriter, 744
OTF2_SnapWriter.h, 744
OTF2_SnapWriter_Enter, 745
OTF2_SnapWriter_GetLocationID, 746
OTF2_SnapWriter_MeasurementOnOff, 746
OTF2_SnapWriter_Metric, 746
OTF2_SnapWriter_MpiCollectiveBegin, 747
OTF2_SnapWriter_MpiCollectiveEnd, 748
OTF2_SnapWriter_MpiIrecv, 748
OTF2_SnapWriter_MpiIrecvRequest, 749
OTF2_SnapWriter_MpiIsend, 750
OTF2_SnapWriter_MpiIsendComplete, 751
OTF2_SnapWriter_MpiRecv, 751
OTF2_SnapWriter_MpiSend, 752
OTF2_SnapWriter_OmpAcquireLock, 754
OTF2_SnapWriter_OmpFork, 754
OTF2_SnapWriter_OmpTaskCreate, 755
OTF2_SnapWriter_OmpTaskSwitch, 756
OTF2_SnapWriter_ParameterInt, 756
OTF2_SnapWriter_ParameterString, 756
OTF2_SnapWriter_ParameterUnsignedInt, 757
OTF2_SnapWriter_SnapshotEnd, 758
OTF2_SnapWriter_SnapshotStart, 758
OTF2_SnapWriter.h, 745
OTF2_SnapWriter_MeasurementOnOff, 746
OTF2_SnapWriter_Metric, 746
OTF2_SnapWriter_MpiCollectiveBegin, 747
OTF2_SnapWriter.h, 747

- OTF2_SnapWriter_MpiCollectiveEnd
OTF2_SnapWriter.h, [748](#)
- OTF2_SnapWriter_MpiIrecv
OTF2_SnapWriter.h, [748](#)
- OTF2_SnapWriter_MpiIrecvRequest
OTF2_SnapWriter.h, [749](#)
- OTF2_SnapWriter_MpiIsend
OTF2_SnapWriter.h, [750](#)
- OTF2_SnapWriter_MpiIsendComplete
OTF2_SnapWriter.h, [751](#)
- OTF2_SnapWriter_MpiRecv
OTF2_SnapWriter.h, [751](#)
- OTF2_SnapWriter_MpiSend
OTF2_SnapWriter.h, [752](#)
- OTF2_SnapWriter_OmpAcquireLock
OTF2_SnapWriter.h, [753](#)
- OTF2_SnapWriter_OmpFork
OTF2_SnapWriter.h, [754](#)
- OTF2_SnapWriter_OmpTaskCreate
OTF2_SnapWriter.h, [754](#)
- OTF2_SnapWriter_OmpTaskSwitch
OTF2_SnapWriter.h, [755](#)
- OTF2_SnapWriter_ParameterInt
OTF2_SnapWriter.h, [756](#)
- OTF2_SnapWriter_ParameterString
OTF2_SnapWriter.h, [756](#)
- OTF2_SnapWriter_ParameterUnsignedInt
OTF2_SnapWriter.h, [757](#)
- OTF2_SnapWriter_SnapshotEnd
OTF2_SnapWriter.h, [758](#)
- OTF2_SnapWriter_SnapshotStart
OTF2_SnapWriter.h, [758](#)
- OTF2_SystemTreeDomain_enum
OTF2_Definitions.h, [209](#)
- OTF2_Thumbnail.h
 - OTF2_ThumbReader_GetHeader, [760](#)
 - OTF2_ThumbReader_ReadSample,
[761](#)
 - OTF2_ThumbWriter_WriteSample,
[761](#)
- OTF2_ThumbnailType_enum
OTF2_GeneralDefinitions.h, [459](#)
- OTF2_ThumbReader_GetHeader
OTF2_Thumbnail.h, [760](#)
- OTF2_ThumbReader_ReadSample
OTF2_Thumbnail.h, [761](#)
- OTF2_ThumbWriter_WriteSample
OTF2_Thumbnail.h, [761](#)
- OTF2_Type_enum
OTF2_GeneralDefinitions.h, [460](#)
- Usage in reading mode - a simple exam-
ple, [115](#)
- Usage in reading mode - MPI example,
[101](#)
- Usage in writing mode - a simple exam-
ple, [86](#)
- Usage in writing mode - MPI example,
[108](#)
- Usage of OTF2 tools, [17](#)