

OPEN TRACE FORMAT 2

USER MANUAL

1.2.1 (revision 3238)

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 Tools	5
A.1 Usage of OTF2 tools	5
A.1.1 OTF2 config tool	5
A.1.2 OTF2 print tool	6
A.1.3 OTF2 snapshots tool	6
A.1.4 OTF2 marker tool	6
Appendix B OTF2 INSTALL	9
Appendix C List of all definition records	19
C.1 ClockProperties	19
C.2 MappingTable	20
C.3 ClockOffset	20
C.4 String	21
C.5 Attribute	21
C.6 SystemTreeNode	22
C.7 LocationGroup	22
C.8 Location	23
C.9 Region	23
C.10 Callsite	24
C.11 Callpath	25
C.12 Group	25
C.13 MetricMember	26
C.14 MetricClass	27
C.15 MetricInstance	27
C.16 Comm	28
C.17 Parameter	29
C.18 RmaWin	29

CONTENTS

C.19 MetricClassRecorder	30
C.20 SystemTreeNodeProperty	30
C.21 SystemTreeNodeDomain	31
Appendix D List of all event records	33
D.1 BufferFlush	33
D.2 MeasurementOnOff	33
D.3 Enter	34
D.4 Leave	34
D.5 MpiSend	35
D.6 MpiIsend	35
D.7 MpiIsendComplete	36
D.8 MpiIrecvRequest	37
D.9 MpiRecv	37
D.10 MpiIrecv	38
D.11 MpiRequestTest	38
D.12 MpiRequestCancelled	39
D.13 MpiCollectiveBegin	39
D.14 MpiCollectiveEnd	40
D.15 OmpFork	40
D.16 OmpJoin	41
D.17 OmpAcquireLock	41
D.18 OmpReleaseLock	42
D.19 OmpTaskCreate	43
D.20 OmpTaskSwitch	43
D.21 OmpTaskComplete	44
D.22 Metric	45
D.23 ParameterString	45
D.24 ParameterInt	46
D.25 ParameterUnsignedInt	47
D.26 RmaWinCreate	47
D.27 RmaWinDestroy	48
D.28 RmaCollectiveBegin	48
D.29 RmaCollectiveEnd	49
D.30 RmaGroupSync	49
D.31 RmaRequestLock	50
D.32 RmaAcquireLock	51
D.33 RmaTryLock	51
D.34 RmaReleaseLock	52
D.35 RmaSync	53
D.36 RmaWaitChange	53
D.37 RmaPut	54
D.38 RmaGet	54
D.39 RmaAtomic	55

CONTENTS

D.40 RmaOpCompleteBlocking	56
D.41 RmaOpCompleteNonBlocking	56
D.42 RmaOpTest	57
D.43 RmaOpCompleteRemote	57
D.44 ThreadFork	58
D.45 ThreadJoin	58
D.46 ThreadTeamBegin	59
D.47 ThreadTeamEnd	59
D.48 ThreadAcquireLock	60
D.49 ThreadReleaseLock	60
D.50 ThreadTaskCreate	61
D.51 ThreadTaskSwitch	62
D.52 ThreadTaskComplete	62
Appendix E List of all snapshot records	65
E.1 SnapshotStart	65
E.2 SnapshotEnd	66
E.3 MeasurementOnOffSnap	66
E.4 EnterSnap	67
E.5 MpiSendSnap	67
E.6 MpiIsendSnap	68
E.7 MpiIsendCompleteSnap	69
E.8 MpiRecvSnap	69
E.9 MpiIrecvRequestSnap	70
E.10 MpiIrecvSnap	71
E.11 MpiCollectiveBeginSnap	72
E.12 MpiCollectiveEndSnap	72
E.13 OmpForkSnap	73
E.14 OmpAcquireLockSnap	73
E.15 OmpTaskCreateSnap	74
E.16 OmpTaskSwitchSnap	75
E.17 MetricSnap	75
E.18 ParameterStringSnap	76
E.19 ParameterIntSnap	77
E.20 ParameterUnsignedIntSnap	77
Appendix F Usage in writing mode	79
F.1 Usage in writing mode - a simple example	79
Appendix G Usage in reading mode	83
G.1 Usage in reading mode - a simple example	83
Appendix H Deprecated List	87

Appendix I	Data Structure Documentation	89
I.1	OTF2_AttributeValue Union Reference	89
I.1.1	Detailed Description	90
I.2	OTF2_FileSionCallbacks Struct Reference	91
I.2.1	Detailed Description	91
I.3	OTF2_FlushCallbacks Struct Reference	91
I.3.1	Detailed Description	92
I.4	OTF2_MemoryCallbacks Struct Reference	92
I.4.1	Detailed Description	92
I.5	OTF2_MetricValue Union Reference	92
I.5.1	Detailed Description	92
Appendix J	File Documentation	95
J.1	otf2.h File Reference	95
J.1.1	Detailed Description	95
J.2	OTF2_Archive.h File Reference	95
J.2.1	Detailed Description	101
J.2.2	Define Documentation	101
J.2.3	Typedef Documentation	101
J.2.4	Enumeration Type Documentation	102
J.2.5	Function Documentation	102
J.3	OTF2_AttributeList.h File Reference	124
J.3.1	Detailed Description	129
J.3.2	How to use the attribute list for writing	130
J.3.3	Function Documentation	130
J.4	OTF2_Callbacks.h File Reference	150
J.4.1	Detailed Description	151
J.4.2	Typedef Documentation	152
J.5	OTF2_Definitions.h File Reference	155
J.5.1	Detailed Description	161
J.5.2	Enumeration Type Documentation	161
J.6	OTF2_DefReader.h File Reference	169
J.6.1	Detailed Description	170
J.6.2	Function Documentation	170
J.7	OTF2_DefReaderCallbacks.h File Reference	172
J.7.1	Detailed Description	177
J.7.2	Typedef Documentation	178
J.7.3	Function Documentation	191
J.8	OTF2_DefWriter.h File Reference	202
J.8.1	Detailed Description	204
J.8.2	Function Documentation	204
J.9	OTF2_ErrorCodes.h File Reference	217
J.9.1	Detailed Description	220
J.9.2	Typedef Documentation	221

CONTENTS

J.9.3	Enumeration Type Documentation	221
J.9.4	Function Documentation	225
J.10	OTF2_Events.h File Reference	226
J.10.1	Detailed Description	227
J.10.2	Enumeration Type Documentation	228
J.11	OTF2_EvtReader.h File Reference	230
J.11.1	Detailed Description	231
J.11.2	Function Documentation	232
J.12	OTF2_EvtReaderCallbacks.h File Reference	235
J.12.1	Detailed Description	247
J.12.2	Typedef Documentation	248
J.12.3	Function Documentation	287
J.13	OTF2_EvtWriter.h File Reference	313
J.13.1	Detailed Description	320
J.13.2	Function Documentation	320
J.14	OTF2_GeneralDefinitions.h File Reference	356
J.14.1	Detailed Description	362
J.14.2	Define Documentation	363
J.14.3	Enumeration Type Documentation	363
J.15	OTF2_GlobalDefReader.h File Reference	367
J.15.1	Detailed Description	368
J.15.2	Function Documentation	368
J.16	OTF2_GlobalDefReaderCallbacks.h File Reference	369
J.16.1	Detailed Description	374
J.16.2	Typedef Documentation	375
J.16.3	Function Documentation	388
J.17	OTF2_GlobalDefWriter.h File Reference	399
J.17.1	Detailed Description	401
J.17.2	Function Documentation	402
J.18	OTF2_GlobalEvtReader.h File Reference	415
J.18.1	Detailed Description	415
J.18.2	Function Documentation	416
J.19	OTF2_GlobalEvtReaderCallbacks.h File Reference	418
J.19.1	Detailed Description	430
J.19.2	Typedef Documentation	430
J.19.3	Function Documentation	466
J.20	OTF2_GlobalSnapReader.h File Reference	494
J.20.1	Detailed Description	495
J.20.2	Function Documentation	495
J.21	OTF2_GlobalSnapReaderCallbacks.h File Reference	496
J.21.1	Detailed Description	502
J.21.2	Typedef Documentation	502
J.21.3	Function Documentation	518
J.22	OTF2_IdMap.h File Reference	531

CONTENTS

J.22.1	Detailed Description	532
J.22.2	Typedef Documentation	533
J.22.3	Enumeration Type Documentation	533
J.22.4	Function Documentation	533
J.23	OTF2_Marker.h File Reference	537
J.23.1	Detailed Description	538
J.23.2	Enumeration Type Documentation	538
J.24	OTF2_MarkerReader.h File Reference	539
J.24.1	Detailed Description	539
J.24.2	Function Documentation	540
J.25	OTF2_MarkerReaderCallbacks.h File Reference	541
J.25.1	Detailed Description	542
J.25.2	Typedef Documentation	542
J.25.3	Function Documentation	544
J.26	OTF2_MarkerWriter.h File Reference	547
J.26.1	Detailed Description	547
J.26.2	Function Documentation	548
J.27	OTF2_Reader.h File Reference	549
J.27.1	Detailed Description	554
J.27.2	Function Documentation	554
J.28	OTF2_SnapReader.h File Reference	579
J.28.1	Detailed Description	580
J.28.2	Function Documentation	580
J.29	OTF2_SnapReaderCallbacks.h File Reference	582
J.29.1	Detailed Description	587
J.29.2	Typedef Documentation	587
J.29.3	Function Documentation	604
J.30	OTF2_SnapWriter.h File Reference	616
J.30.1	Detailed Description	619
J.30.2	Typedef Documentation	619
J.30.3	Function Documentation	620
J.31	OTF2_Thumbnail.h File Reference	634
J.31.1	Detailed Description	635
J.31.2	Function Documentation	635

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

[Usage in writing mode](#)

[Usage in reading mode](#)

[Definition records](#)

[Event records](#)

[Snapshot records](#)

[Usage of OTF2 tools](#)

Appendices

Appendix A

OTF2 Tools

A.1 Usage of OTF2 tools

A.1.1 OTF2 config tool

A call to `otf2-config` has the following syntax:

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

Commands:

<code>--cflags</code>	prints additional compiler flags. They already contain the include flags
<code>--cppflags</code>	prints the include flags for the OTF2 headers
<code>--libs</code>	prints the required libraries for linking
<code>--ldflags</code>	prints the required linker flags
<code>--cc</code>	prints the C compiler name
<code>--help</code>	prints this usage information
<code>--version</code>	prints the version number of the OTF2 package and
<code>--otf2-revision</code>	prints the revision number of the OTF2 package
<code>--common-revision</code>	prints the revision number of the common package
<code>--interface-version</code>	prints the interface version number

Options:

<code>--backend</code>	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
<code>--cuda</code>	specifies that the required flags are for the CUDA compiler

A.1.2 OTF2 print tool

A call to `oft2-print` has the following syntax:

Usage: `otf2-print [OPTION]... [--] ANCHORFILE`

Print selected content of the OTF2 archive specified by `ANCHORFILE`.

Options:

<code>-A, --show-all</code>	print all output including definitions and anchor file
<code>-G, --show-global-defs</code>	print all global definitions
<code>-I, --show-info</code>	print information from the anchor file
<code>-T, --show-thumbnails</code>	print the headers from all thumbnails
<code>-M, --show-mappings</code>	print mappings to global definitions
<code>-C, --show-clock-offsets</code>	print clock offsets to global timer
<code>-L, --location <LID></code>	limit output to location <LID>
<code>-s, --step <N></code>	step through output by steps of <N> events
<code>--time <MIN> <MAX></code>	limit output to events within time interval
<code>--system-tree</code>	output system tree to dot-file
<code>--silent</code>	only validate trace and do not print any events
<code>-d, --debug</code>	turn on debug mode
<code>-V, --version</code>	print version information
<code>-h, --help</code>	print this help information

A.1.3 OTF2 snapshots tool

A call to `oft2-snapshots` has the following syntax:

Usage: `otf2-snapshots [OPTION]... ANCHORFILE`

Append snapshots to existing otf2 traces at given 'break' timestamps.

Options:

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

A.1.4 OTF2 marker tool

A call to `oft2-marker` has the following syntax:

Usage: `otf2-marker [OPTION] [ARGUMENTS]... ANCHORFILE`

A.1 Usage of OTF2 tools

Read or edit a marker file.

Options:

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

Argument descriptions:

<GROUP>, <CATEGORY>, <TEXT>	Arbitrary strings.
<SEVERITY>	One of: <ul style="list-style-type: none">* NONE* LOW* MEDIUM* HIGH
<TIME>	One of the following formats: <ul style="list-style-type: none">* <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: <ul style="list-style-type: none">* GLOBAL* LOCATION:<LOCATION-REF>* LOCATION_GROUP:<LOCATION-GROUP-REF>* SYSTEM_TREE_NODE:<SYSTEM-TREE-NODE-REF>* GROUP:<GROUP-REF>* COMM:<COMMUNICATOR-REF>

APPENDIX A. OTF2 TOOLS

<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'.

Appendix B

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 B. 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')
--with-platform=(auto,disabled,<platform>)	autodetect platform [auto], disabled or select one from: altix, aix, arm, bgl, bgp, bgq, crayxt, linux, solaris, mac, necsx.
--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-sionconfig=(yes|no|<path-to-sionconfig>)`
Whether to use sionconfig and where to find it.
"yes" assumes it is in PATH [no].
`--with-otf-prefix=PREFIX`
Prefix where otf is installed (optional)
`--with-otf-exec-prefix=PREFIX`
Exec prefix where otf is installed (optional)
`--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

APPENDIX B. OTF2 INSTALL

CPP	C preprocessor
SIONCONFIG	Absolute path to sionconfig, including "sionconfig".
OTF_CONFIG	config script used for otf
OTF_CFLAGS	CFLAGS used for the otf
OTF_LIBS	LIBS used for the otf
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>.

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

APPENDIX B. OTF2 INSTALL

=====

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

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

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

*Note Defining Variables::, for more details.

Compiling For Multiple Architectures

=====

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

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

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

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

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

Installation Names

=====

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

You can specify separate installation prefixes for

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

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

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

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

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

Optional Features =====

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

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

APPENDIX B. OTF2 INSTALL

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

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

Particular systems
=====

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

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

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

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

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

and if that doesn't work, try

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

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

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

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

Specifying the System Type
=====

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

type, such as `'sun4'`, or a canonical name which has the form:

CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

OS
KERNEL-OS

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

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

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

Sharing Defaults =====

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

Defining Variables =====

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

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

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

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

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

`'configure'` Invocation =====

APPENDIX B. OTF2 INSTALL

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

'--help'

'-h'

Print a summary of all of the options to 'configure', and exit.

'--help=short'

'--help=recursive'

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

'--version'

'-v'

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

'--cache-file=FILE'

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

'--config-cache'

'-C'

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

'--quiet'

'--silent'

'-q'

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

'--srcdir=DIR'

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

'--prefix=DIR'

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

'--no-create'

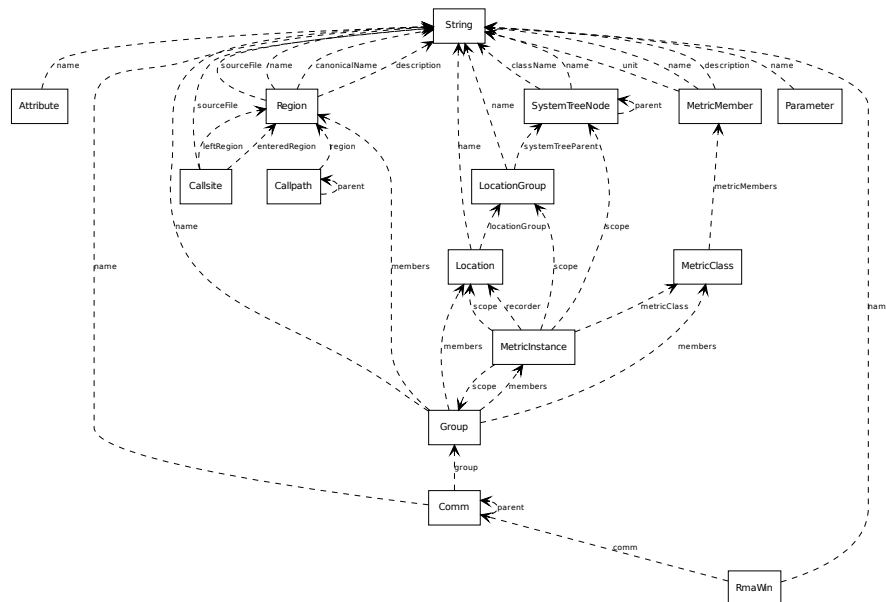
'-n'

Run the configure checks, but stop before creating any output files.

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

Appendix C

List of all definition records



C.1 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.

APPENDIX C. LIST OF ALL DEFINITION RECORDS

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.

See also

[OTF2_GlobalDefWriter_WriteClockProperties\(\)](#)

Since

Version 1.0

C.2 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.3 ClockOffset

Clock offsets are used for clock corrections.

C.5 Attribute

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 <i>time</i> .
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.4 OTF2_StringRef String

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteString\(\)](#)

[OTF2_DefWriter_WriteString\(\)](#)

Since

Version 1.0

C.5 OTF2_AttributeRef Attribute

Attributes

OTF2_StringRef	name	Name of the attribute. References a String definition.
OTF2_Type	type	Type of the attribute value.

APPENDIX C. LIST OF ALL DEFINITION RECORDS

See also

[OTF2_GlobalDefWriter_WriteAttribute\(\)](#)
[OTF2_DefWriter_WriteAttribute\(\)](#)

Since

Version 1.0

C.6 [OTF2_SystemTreeNodeRef](#) SystemTreeNode

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteSystemTreeNode\(\)](#)
[OTF2_DefWriter_WriteSystemTreeNode\(\)](#)

Since

Version 1.0

C.7 [OTF2_LocationGroupRef](#) LocationGroup

Attributes

OTF2_StringRef	name	Name of the group. References a String definition.
OTF2_LocationGroupType	location-GroupType	Type of this group.
OTF2_SystemTreeNodeRef	systemTreeParent	Parent of this location group in the system tree. References a SystemTreeNode definition.

C.9 Region

See also

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

Since

Version 1.0

C.8 OTF2_LocationRef Location

Attributes

OTF2_StringRef	name	Name of the location References a String definition.
OTF2_LocationType	location-Type	Location type.
uint64_t	numberOfEvents	Number of events this location has recorded.
OTF2_LocationGroupRef	location-Group	Location group which includes this location. References a LocationGroup definition.

See also

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

Since

Version 1.0

C.9 OTF2_RegionRef Region

Attributes

OTF2_StringRef	name	Name of the region (demangled name if available). References a String definition.
OTF2_StringRef	canonical-Name	Alternative name of the region (e.g. mangled name). References a String definition. Since version 1.1.
OTF2_StringRef	description	A more detailed description of this region. References a String definition.
OTF2_RegionRole	regionRole	Region role. Since version 1.1.

APPENDIX C. LIST OF ALL DEFINITION RECORDS

OTF2_Paradigm	paradigm	Paradigm. Since version 1.1.
OTF2_RegionFlag	regionFlags	Region flags. Since version 1.1.
OTF2_StringRef	sourceFile	The source file where this region was declared. References a String definition.
uint32_t	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.10 [OTF2_CallsiteRef](#) Callsite

Attributes

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

See also

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

Since

Version 1.0

C.12 Group

C.11 [OTF2_CallpathRef](#) Callpath

Attributes

OTF2_CallpathRef	parent	References a Callpath definition.
OTF2_RegionRef	region	References a Region definition.

See also

[OTF2_GlobalDefWriter_WriteCallpath\(\)](#)

[OTF2_DefWriter_WriteCallpath\(\)](#)

Since

Version 1.0

C.12 [OTF2_GroupRef](#) Group

Attributes

OTF2_StringRef	name	Name of this group References a String definition.
OTF2_GroupType	groupType	The type of this group. Since version 1.2.
OTF2_Paradigm	paradigm	The paradigm of this communication group. Since version 1.2.
OTF2_GroupFlag	groupFlags	Flags for this group. Since version 1.2.
uint32_t	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.13 [OTF2_MetricMemberRef](#) MetricMember

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

Attributes

OTF2_StringRef	name	Name of the metric. References a String definition.
OTF2_StringRef	description	Description of the metric. References a String definition.
OTF2_MetricType	metricType	Metric type: PAPI, etc.
OTF2_MetricMode	metric-Mode	Metric mode: accumulative, fix, relative, etc.
OTF2_Type	valueType	Type of the value: int64_t, uint64_t, or double.
OTF2_MetricBase	metricBase	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
int64_t	exponent	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$, 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.
OTF2_StringRef	unit	Unit of the metric. This needs to be the scale free base unit, ie. "bytes", "operations", or "seconds". In particular this unit should not have any scale prefix. References a String definition.

See also

[OTF2_GlobalDefWriter_WriteMetricMember\(\)](#)

C.15 MetricInstance

[OTF2_DefWriter_WriteMetricMember\(\)](#)

Since

Version 1.0

C.14 [OTF2_MetricRef](#) MetricClass

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

Attributes

uint8_t	numberOfMetrics	Number of metrics within the set.
OTF2_MetricMemberRef	metricMembers [numberOfMetrics]	List of metric members. References a MetricMember definition.
OTF2_MetricOccurrence	metricOccurrence	Defines occurrence of a metric set.
OTF2_RecorderKind	recorderKind	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

See also

[OTF2_GlobalDefWriter_WriteMetricClass\(\)](#)
[OTF2_DefWriter_WriteMetricClass\(\)](#)

Since

Version 1.0

C.15 [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](#).

APPENDIX C. LIST OF ALL DEFINITION RECORDS

Attributes

OTF2_MetricRef	metricClass	The instanced MetricClass . This metric class must be of kind OTF2_RECORDER_KIND_ABSTRACT . References a MetricClass definition.
OTF2_LocationRef	recorder	Recorder of the metric: location ID. References a Location definition.
OTF2_MetricScope	metric-Scope	Defines type of scope: location, location group, system tree node, or a generic group of locations.
uint64_t	scope	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

See also

[OTF2_GlobalDefWriter_WriteMetricInstance\(\)](#)
[OTF2_DefWriter_WriteMetricInstance\(\)](#)

Since

Version 1.0

C.16 [OTF2_CommRef](#) Comm

Attributes

OTF2_StringRef	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 String definition.
OTF2_GroupRef	group	The describing MPI group of this MPI communicator The group needs to be of type <code>OTF2_GROUP_TYPE_MPI_GROUP</code> or <code>OTF2_GROUP_TYPE_MPI_COMM_SELF</code> . References a Group definition.
OTF2_CommRef	parent	The parent MPI communicator from which this communicator was created, if any. Use OTF2_UNDEFINED_COMM to indicate no parent. References a Comm definition.

C.18 RmaWin

See also

[OTF2_GlobalDefWriter_WriteComm\(\)](#)

[OTF2_DefWriter_WriteComm\(\)](#)

Since

Version 1.0

C.17 OTF2_ParameterRef Parameter

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteParameter\(\)](#)

[OTF2_DefWriter_WriteParameter\(\)](#)

Since

Version 1.0

C.18 OTF2_RmaWinRef RmaWin

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

Attributes

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

See also

[OTF2_GlobalDefWriter_WriteRmaWin\(\)](#)

[OTF2_DefWriter_WriteRmaWin\(\)](#)

Since

Version 1.2

C.19 MetricClassRecorder**Attributes**

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

See also[OTF2_GlobalDefWriter_WriteMetricClassRecorder\(\)](#)[OTF2_DefWriter_WriteMetricClassRecorder\(\)](#)**Since**

Version 1.2

C.20 SystemTreeNodeProperty**Attributes**

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

See also[OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty\(\)](#)[OTF2_DefWriter_WriteSystemTreeNodeProperty\(\)](#)**Since**

Version 1.2

C.21 SystemTreeNodeDomain

C.21 SystemTreeNodeDomain

Attributes

OTF2_-SystemTreeNodeRef	sys- temTreeN- ode	Parent SystemTreeNode definition to which this one is a supplementary definition. References a SystemTreeNode definition.
OTF2_-SystemTreeDomain	sys- temTreeDo- main	

See also

[OTF2_GlobalDefWriter_WriteSystemTreeNodeDomain\(\)](#)
[OTF2_DefWriter_WriteSystemTreeNodeDomain\(\)](#)

Since

Version 1.2

APPENDIX C. LIST OF ALL DEFINITION RECORDS

Appendix D

List of all event records

D.1 BufferFlush

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

Attributes

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

See also

[OTF2_EvtWriter_BufferFlush\(\)](#)

Since

Version 1.0

D.2 MeasurementOnOff

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

Attributes

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

APPENDIX D. LIST OF ALL EVENT RECORDS

See also

[OTF2_EvtWriter_MeasurementOnOff\(\)](#)

Since

Version 1.0

D.3 Enter

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

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RegionRef	region	Needs to be defined in a definition record References a Region definition and will be mapped to the global definition if a mapping table of type OTF2-MAPPING_REGION is available.

See also

[OTF2_EvtWriter_Enter\(\)](#)

Since

Version 1.0

D.4 Leave

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

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RegionRef	region	Needs to be defined in a definition record References a Region definition and will be mapped to the global definition if a mapping table of type OTF2-MAPPING_REGION is available.

D.6 MpiIsend

See also

[OTF2_EvtWriter_Leave\(\)](#)

Since

Version 1.0

D.5 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 <i>communicator</i> .
OTF2_CommRef	communicator	Communicator ID. References a Comm definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_COMM is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length

See also

[OTF2_EvtWriter_MpiSend\(\)](#)

Since

Version 1.0

D.6 MpiIsend

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

APPENDIX D. LIST OF ALL EVENT RECORDS

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

See also

[OTF2_EvtWriter_MpiIsend\(\)](#)

Since

Version 1.0

D.7 MpiIsendComplete

Signals the completion of non-blocking send request.

Attributes

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

See also

[OTF2_EvtWriter_MpiIsendComplete\(\)](#)

Since

Version 1.0

D.9 MpiRecv

D.8 MpiRecvRequest

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

Attributes

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

See also

[OTF2_EvtWriter_MpiRecvRequest\(\)](#)

Since

Version 1.0

D.9 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 <i>communicator</i> .
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\(\)](#)

APPENDIX D. LIST OF ALL EVENT RECORDS

Since

Version 1.0

D.10 MpiIrecv

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

Attributes

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

See also

[OTF2_EvtWriter_MpiIrecv\(\)](#)

Since

Version 1.0

D.11 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.
38 uint64_t	requestID	ID of the related request

D.13 MpiCollectiveBegin

See also

[OTF2_EvtWriter_MpiRequestTest\(\)](#)

Since

Version 1.0

D.12 MpiRequestCancelled

This events appears if the program canceled a request.

Attributes

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

See also

[OTF2_EvtWriter_MpiRequestCancelled\(\)](#)

Since

Version 1.0

D.13 MpiCollectiveBegin

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

Attributes

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

See also

[OTF2_EvtWriter_MpiCollectiveBegin\(\)](#)

Since

Version 1.0

D.14 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 <i>communicator</i> .
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

D.15 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.

D.17 OmpAcquireLock

See also

[OTF2_EvtWriter_OmpFork\(\)](#)

Since

Version 1.0

Deprecated

In version 1.2

D.16 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

D.17 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 record.

APPENDIX D. LIST OF ALL EVENT RECORDS

Attributes

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

See also

[OTF2_EvtWriter_OmpAcquireLock\(\)](#)

Since

Version 1.0

Deprecated

In version 1.2

D.18 OmpReleaseLock

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

This event record is superseded by the [ThreadReleaseLock](#) event record and should not be used when the [ThreadReleaseLock](#) event record is in use.

Attributes

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

D.20 OmpTaskSwitch

See also

[OTF2_EvtWriter_OmpReleaseLock\(\)](#)

Since

Version 1.0

Deprecated

In version 1.2

D.19 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

D.20 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.

APPENDIX D. LIST OF ALL EVENT RECORDS

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

D.21 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

D.23 ParameterString

D.22 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.
OTF2_MetricValue	metricValues [numberOfMetrics]	List of metric values.

See also

[OTF2_EvtWriter_Metric\(\)](#)

Since

Version 1.0

D.23 ParameterString

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

APPENDIX D. LIST OF ALL EVENT RECORDS

Attributes

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

See also

[OTF2_EvtWriter_ParameterString\(\)](#)

Since

Version 1.0

D.24 ParameterInt

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

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_-ParameterRef	parameter	Parameter ID. References a Parameter definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_-PARAMETER is available.
int64_t	value	Value of the recorded parameter.

See also

[OTF2_EvtWriter_ParameterInt\(\)](#)

D.26 RmaWinCreate

Since

Version 1.0

D.25 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

D.26 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.

APPENDIX D. LIST OF ALL EVENT RECORDS

See also

[OTF2_EvtWriter_RmaWinCreate\(\)](#)

Since

Version 1.2

D.27 RmaWinDestroy

An RmaWinDestroy record denotes the destruction of an RMA window.

Attributes

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

See also

[OTF2_EvtWriter_RmaWinDestroy\(\)](#)

Since

Version 1.2

D.28 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\(\)](#)

D.30 RmaGroupSync

Since

Version 1.2

D.29 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

D.30 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.
----------------------------------	----------	---

APPENDIX D. LIST OF ALL EVENT RECORDS

OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaSyncLevel	syncLevel	Synchronization level of this collective operation.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a RmaWin definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_RMA_WIN is available.
OTF2_GroupRef	group	Group of remote processes involved in synchronization. References a Group definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_GROUP is available.

See also

[OTF2_EvtWriter_RmaGroupSync\(\)](#)

Since

Version 1.2

D.31 RmaRequestLock

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

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a RmaWin definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_RMA_WIN is available.
uint32_t	remote	Rank of the locked remote process.
uint64_t	lockId	ID of the lock aquired, if multiple locks are defined on a window.
OTF2_LockType	lockType	Type of lock aquired.

D.33 RmaTryLock

See also

[OTF2_EvtWriter_RmaRequestLock\(\)](#)

Since

Version 1.2

D.32 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

D.33 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.
----------------------------------	----------	---

APPENDIX D. LIST OF ALL EVENT RECORDS

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_RmaTryLock\(\)](#)

Since

Version 1.2

D.34 RmaReleaseLock

An RmaReleaseLock record denotes the time the lock was released.

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_RmaWinRef	win	ID of the window used for this operation. References a RmaWin definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_RMA_WIN is available.
uint32_t	remote	Rank of the locked remote process.
uint64_t	lockId	ID of the lock released, if multiple locks are defined on a window.

See also

[OTF2_EvtWriter_RmaReleaseLock\(\)](#)

Since

Version 1.2

D.36 RmaWaitChange

D.35 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

D.36 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\(\)](#)

APPENDIX D. LIST OF ALL EVENT RECORDS

Since

Version 1.2

D.37 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

D.38 RmaGet

An RmaGet 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.

D.39 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

D.39 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

D.40 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

D.41 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.

D.43 RmaOpCompleteRemote

See also

[OTF2_EvtWriter_RmaOpCompleteNonBlocking\(\)](#)

Since

Version 1.2

D.42 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

D.43 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.

APPENDIX D. LIST OF ALL EVENT RECORDS

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_RmaOpCompleteRemote\(\)](#)

Since

Version 1.2

D.44 ThreadFork

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

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_Paradigm	model	
uint32_t	numberOfRequestedThreads	Requested size of the team.

See also

[OTF2_EvtWriter_ThreadFork\(\)](#)

Since

Version 1.2

D.45 ThreadJoin

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

D.47 ThreadTeamEnd

Attributes

OTF2_LocationRef	location	The location where this event happened.
OTF2_TimeStamp	timestamp	The time when this event happened.
OTF2_Paradigm	model	

See also

[OTF2_EvtWriter_ThreadJoin\(\)](#)

Since

Version 1.2

D.46 ThreadTeamBegin

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

D.47 ThreadTeamEnd

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.

APPENDIX D. LIST OF ALL EVENT RECORDS

See also

[OTF2_EvtWriter_ThreadTeamEnd\(\)](#)

Since

Version 1.2

D.48 ThreadAcquireLock

An ThreadAcquireLock record marks that a thread acquires an lock.

Attributes

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

See also

[OTF2_EvtWriter_ThreadAcquireLock\(\)](#)

Since

Version 1.2

D.49 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	

D.50 ThreadTaskCreate

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

D.50 ThreadTaskCreate

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

Attributes

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

See also

[OTF2_EvtWriter_ThreadTaskCreate\(\)](#)

Since

Version 1.2

D.51 ThreadTaskSwitch

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

Attributes

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

See also

[OTF2_EvtWriter_ThreadTaskSwitch\(\)](#)

Since

Version 1.2

D.52 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.

D.52 ThreadTaskComplete

See also

[OTF2_EvtWriter_ThreadTaskComplete\(\)](#)

Since

Version 1.2

APPENDIX D. LIST OF ALL EVENT RECORDS

Appendix E

List of all snapshot records

E.1 SnapshotStart

This record marks the start of a snapshot.

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

Attributes

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

See also

[OTF2_SnapWriter_SnapshotStart\(\)](#)

Since

Version 1.2

E.2 SnapshotEnd

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

Attributes

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

See also

[OTF2_SnapWriter_SnapshotEnd\(\)](#)

Since

Version 1.2

E.3 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

E.5 MpiSendSnap

E.4 EnterSnap

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

Attributes

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

E.5 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 *Mpirecv* 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 <i>communicator</i> .

APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

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

See also

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

Since

Version 1.2

E.6 MpilsendSnap

This record exists for each [MpiSend](#) event where an corresponding [MpiSendComplete](#) or [MpiRequestCancelled](#) event did not occur on this location before the snapshot. Or the corresponding [MpiSendComplete](#) did occurred (the [MpiSendCompleteSnap](#) 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 [Mpilrecv](#) event.)

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent- Time	The original time this event happended.
uint32_t	receiver	MPI rank of receiver in <i>communicator</i> .
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

E.8 MpiRecvSnap

See also

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

Since

Version 1.2

E.7 MpisendCompleteSnap

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

Attributes

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

See also

[MpiIsendComplete](#) event
[OTF2_SnapWriter_MpiIsendComplete\(\)](#)

Since

Version 1.2

E.8 MpiRecvSnap

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

APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

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

See also

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

Since

Version 1.2

E.9 MpiIrecvRequestSnap

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

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.
uint64_t	requestID	ID of the requested receive

E.10 MpiIrecvSnap

See also

[MpiIrecvRequest](#) event
[OTF2_SnapWriter_MpiIrecvRequest\(\)](#)

Since

Version 1.2

E.10 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 [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

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

See also

[MpiIrecv](#) event
[OTF2_SnapWriter_MpiIrecv\(\)](#)

Since

Version 1.2

E.11 `MpiCollectiveBeginSnap`

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

Attributes

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

See also

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

Since

Version 1.2

E.12 `MpiCollectiveEndSnap`

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

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time 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 <i>communicator</i> .
uint64_t	sizeSent	Size of the sent message.
uint64_t	sizeReceived	Size of the received message.

E.14 OmpAcquireLockSnap

See also

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

Since

Version 1.2

E.13 OmpForkSnap

This record exists for each [OmpFork](#) event where the corresponding [OmpJoin](#) did not occurred before this snapshot.

Attributes

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

See also

[OmpFork](#) event
[OTF2_SnapWriter_OmpFork\(\)](#)

Since

Version 1.2

E.14 OmpAcquireLockSnap

This record exists for each [OmpAcquireLock](#) event where the corresponding [OmpReleaseLock](#) did not occurred before this snapshot yet.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
----------------------------------	----------	-------------------------------

APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time 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

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

Since

Version 1.2

E.15 OmpTaskCreateSnap

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

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.
uint64_t	taskID	Identifier of the newly created task instance.

See also

[OmpTaskCreate](#) event
[OTF2_SnapWriter_OmpTaskCreate\(\)](#)

Since

Version 1.2

E.17 MetricSnap

E.16 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

E.17 MetricSnap

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

As an exception for metric classes where the metric mode detontes an *OTF2-METRIC_VALUE_RELATIVE* mode the value indicates the accumulation of all previous metric values recorded.

Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
OTF2_MetricRef	metric	Could be a metric class or a metric instance. References a MetricClass , or a MetricInstance definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING-METRIC is available.

APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

uint8_t	numberOfMetrics	Number of metrics with in the set.
OTF2_Type	typeIDs [numberOfMetrics]	List of metric types.
OTF2_MetricValue	metricValues [numberOfMetrics]	List of metric values.

See also

[Metric](#) event
[OTF2_SnapWriter_Metric\(\)](#)

Since

Version 1.2

E.18 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

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.
OTF2_StringRef	string	Value: Handle of a string definition References a String definition and will be mapped to the global definition if a mapping table of type OTF2_MAPPING_STRING is available.

E.20 ParameterUnsignedIntSnap

See also

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

Since

Version 1.2

E.19 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

E.20 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.

APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

Attributes

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

See also

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

Since

Version 1.2

Appendix F

Usage in writing mode

F.1 Usage in writing mode - a simple example

This is a short example of how to use the OTF2 writing interface.

First include the OTF2 header.

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

For this example an additional include statement is necessary.

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

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

```
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 spend in a buffer flush.

APPENDIX F. USAGE IN WRITING MODE

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

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

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

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

Create new archive handle.

```
OTF2_Archive* archive = OTF2_Archive_Open( "ArchivePath", "ArchiveName",
OTF2_FILEMODE_WRITE, 1024 * 1024, 4 * 1024 * 1024, OTF2_SUBSTRATE_POSIX,
OTF2_COMPRESSION_NONE );
```

Set the flush callbacks.

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

Define archive as master.

```
OTF2_Archive_SetMasterSlaveMode( archive, OTF2_MASTER );
```

Get a local event writer and a local definition writer for location 0. Additionally a global definition writer is needed.

```
OTF2_EvtWriter*    evt_writer      = OTF2_Archive_GetEvtWriter( archiv
e, 0 );
OTF2_DefWriter*    def_writer      = OTF2_Archive_GetDefWriter( archiv
e, 0 );
OTF2_GlobalDefWriter* global_def_writer = OTF2_Archive_GetGlobalDefWriter(
archive );
```

Write an enter and a leave record for region 23 to the local event writer.

F.1 Usage in writing mode - a simple example

```
OTF2_EvtWriter_Enter( evt_writer, NULL, get_time(), 23 );
OTF2_EvtWriter_Leave( evt_writer, NULL, get_time(), 23 );
```

Write definitions for the strings as the first records to the global definition writer.

```
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, 23, 3, 4, 5,
OTF2_REGION_ROLE_FUNCTION, OTF2_PARADIGM_USER, OTF2_REGION_FLAG_NONE, 0, 0, 0 );
```

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

```
OTF2_GlobalDefWriter_WriteSystemTreeNode( global_def_writer, 0, 6, 7,
OTF2_UNDEFINED_SYSTEM_TREE_NODE );
OTF2_GlobalDefWriter_WriteLocationGroup( global_def_writer, 0, 1,
OTF2_LOCATION_GROUP_TYPE_PROCESS, 0 );
```

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

```
OTF2_GlobalDefWriter_WriteLocation( global_def_writer, 0, 2,
OTF2_LOCATION_TYPE_CPU_THREAD, 2, 0 );
```

At the end, close the archive and exit. All opened event and definition writers are closed automatically and the according files are created.

```
OTF2_Archive_Close( archive );

return EXIT_SUCCESS;
}
```

To compile your program use a command like:

```
gcc '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
```

APPENDIX F. USAGE IN WRITING MODE

Appendix G

Usage in reading mode

G.1 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.

First include the OTF2 header.

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

For this example two additional include statements are necessary.

```
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <inttypes.h>
```

Define an event callback for entering and leaving a region.

```
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_SUCCESS;
}
```

APPENDIX G. USAGE IN READING MODE

```
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_SUCCESS;
}
```

Define a definition callback that opens a new local event reader for each found location definition. The global event reader will use only events from opened local event readers. Therefore, if only a subset of locations should be read from, only for those locations a local event reader has to be opened. In addition, open a local definition reader, if there are local definitions present in the trace archive. Local definitions contain location specific definitions. Please note: Local definitions must be read in order to use automated identifier translation. Otherwise, all delivered identifiers are invalid.

```
OTF2_CallbackCode
GlobDefLocation_Register( void*      userData,
                          OTF2_LocationRef location,
                          OTF2_StringRef name,
                          OTF2_LocationType locationType,
                          uint64_t     numberOfEvents,
                          OTF2_LocationGroupRef locationGroup )
{
    OTF2_Reader* reader = ( OTF2_Reader* )userData;
    OTF2_EvtReader* evt_reader = OTF2_Reader_GetEvtReader( reader, location );

    OTF2_DefReader* def_reader = OTF2_Reader_GetDefReader( reader, location );
    uint64_t definitions_read = 0;
    OTF2_Reader_ReadAllLocalDefinitions( reader, def_reader, &definitions_read );
}

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" );
```

Get a global definition reader with the above reader handle as argument.

```
OTF2_GlobalDefReader* global_def_reader = OTF2_Reader_GetGlobalDefReader( reader );
```

G.1 Usage in reading mode - a simple example

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

```
OTF2_GlobalDefReaderCallbacks* global_def_callbacks =
OTF2_GlobalDefReaderCallbacks_New();
OTF2_GlobalDefReaderCallbacks_SetLocationCallback( global_def_callbacks, &G
lobDefLocation_Register );
OTF2_Reader_RegisterGlobalDefCallbacks( reader, global_def_reader, global_d
ef_callbacks, reader );
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, &definitio
ns_read );
```

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( r
eader );
```

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_pri
nt );
OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback( event_callbacks, &Leave_pri
nt );

OTF2_Reader_RegisterGlobalEvtCallbacks( reader, global_evt_reader, event_ca
llbacks, 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 );
```

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

APPENDIX G. USAGE IN READING MODE

```
    OTF2_Reader_Close( reader );  
  
    return EXIT_SUCCESS;  
}
```

To compile your program use a command like:

```
gcc `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 H

Deprecated List

Page [List of all event records](#) 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

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 attribute, ...)
Use [OTF2_AttributeList_GetStringRef\(\)](#) instead.

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_EvtWriter *childWriter, ...)
In version 1.2

Global [OTF2_EvtWriter_OmpJoin](#)(OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_EvtWriter *childWriter, ...)
In version 1.2

APPENDIX H. DEPRECATED LIST

Global **OTF2_EvtWriter_OmpReleaseLock**(OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, C
In version 1.2

Global **OTF2_EvtWriter_OmpTaskComplete**(OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, C
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

Appendix I

Data Structure Documentation

I.1 OTF2_AttributeValue Union Reference

Value container for an attributes.

```
#include <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`.

I.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_AttributeList.h](#)

I.2 OTF2_FileSionCallbacks Struct Reference

I.2 OTF2_FileSionCallbacks Struct Reference

Structure holding the SION callbacks.

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

Data Fields

- [OTF2_FileSionClose](#) `otf2_file_sion_close`
Callback which is called to close a SION file.
- [OTF2_FileSionGetRank](#) `otf2_file_sion_get_rank`
Callback which is called to get the MPI rank in read mode.
- [OTF2_FileSionOpen](#) `otf2_file_sion_open`
Callback which is called to open a SION file.

I.2.1 Detailed Description

Structure holding the SION callbacks.

To be used in a call to [OTF2_Archive_SetFileSionCallbacks](#).

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

- [OTF2_Callbacks.h](#)

I.3 OTF2_FlushCallbacks Struct Reference

Structure holding the flush callbacks.

```
#include <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.

I.3.1 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_Callbacks.h](#)

I.4 OTF2_MemoryCallbacks Struct Reference

Structure holding the memory callbacks.

```
#include <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.

I.4.1 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_Callbacks.h](#)

I.5 OTF2_MetricValue Union Reference

Metric value.

I.5.1 Detailed Description

Metric value.

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

I.5 OTF2_MetricValue Union Reference

- [OTF2_Events.h](#)

APPENDIX I. DATA STRUCTURE DOCUMENTATION

Appendix J

File Documentation

J.1 otf2.h File Reference

Main include file for applications using OTF2.

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

J.1.1 Detailed Description

Main include file for applications using OTF2.

Maintainer:

Michael Wagner <michael.wagner@zih.tu-dresden.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.2 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>
```

APPENDIX J. FILE DOCUMENTATION

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

Defines

- `#define OTF2_CHUNK_SIZE_DEFINITIONS_DEFAULT (4 * 1024 * 1024)`
Default size for OTF2's internal event chunk memory handling.
- `#define OTF2_CHUNK_SIZE_EVENTS_DEFAULT (1024 * 1024)`
Default size for OTF2's internal event chunk memory handling.

Typedefs

- `typedef struct OTF2_Archive_struct OTF2_Archive`
Keeps all meta-data for an OTF2 archive.
- `typedef uint8_t OTF2_MasterSlaveMode`
Defines whether a location is master or slave.

Enumerations

- `enum OTF2_MasterSlaveMode_enum {`
 `OTF2_SLAVE = 0,`
 `OTF2_MASTER = 1 }`
Defines whether a location is master or slave.

J.2 OTF2_Archive.h File Reference

Functions

- [OTF2_ErrorCode OTF2_Archive_Close \(OTF2_Archive *archive\)](#)
Close an opened archive.
- [OTF2_ErrorCode OTF2_Archive_CloseDefReader \(OTF2_Archive *archive, OTF2_DefReader *reader\)](#)
Close an opened local definition reader.
- [OTF2_ErrorCode OTF2_Archive_CloseDefWriter \(OTF2_Archive *archive, OTF2_DefWriter *writer\)](#)
Close an opened local definition writer.
- [OTF2_ErrorCode OTF2_Archive_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_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_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.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_Archive_GetChunkSize](#) ([OTF2_Archive](#) *archive, [uint64_t](#) *chunkSizeEvents, [uint64_t](#) *chunkSizeDefs)
Get the chunksize.
- [OTF2_ErrorCode](#) [OTF2_Archive_GetCompression](#) ([OTF2_Archive](#) *archive, [OTF2_Compression](#) *compression)
Get compression mode (none or zlib)
- [OTF2_ErrorCode](#) [OTF2_Archive_GetCreator](#) ([OTF2_Archive](#) *archive, char **creator)
Get creator information.
- [OTF2_DefReader](#) * [OTF2_Archive_GetDefReader](#) ([OTF2_Archive](#) *archive, [OTF2_LocationRef](#) location)
Get a local definition reader.
- [OTF2_DefWriter](#) * [OTF2_Archive_GetDefWriter](#) ([OTF2_Archive](#) *archive, [OTF2_LocationRef](#) location)
Get a local definition writer.
- [OTF2_ErrorCode](#) [OTF2_Archive_GetDescription](#) ([OTF2_Archive](#) *archive, char **description)
Get description.
- [OTF2_EvtReader](#) * [OTF2_Archive_GetEvtReader](#) ([OTF2_Archive](#) *archive, [OTF2_LocationRef](#) location)
Get a local event reader.
- [OTF2_EvtWriter](#) * [OTF2_Archive_GetEvtWriter](#) ([OTF2_Archive](#) *archive, [OTF2_LocationRef](#) location)
Get a local event writer.
- [OTF2_ErrorCode](#) [OTF2_Archive_GetFileSubstrate](#) ([OTF2_Archive](#) *archive, [OTF2_FileSubstrate](#) *substrate)
Get the file substrate (posix, sion, none)
- [OTF2_GlobalDefReader](#) * [OTF2_Archive_GetGlobalDefReader](#) ([OTF2_Archive](#) *archive)
Get a global definition reader.
- [OTF2_GlobalDefWriter](#) * [OTF2_Archive_GetGlobalDefWriter](#) ([OTF2_Archive](#) *archive)
Get a global definition writer.
- [OTF2_GlobalEvtReader](#) * [OTF2_Archive_GetGlobalEvtReader](#) ([OTF2_Archive](#) *archive)
Get a global event reader.
- [OTF2_GlobalSnapReader](#) * [OTF2_Archive_GetGlobalSnapReader](#) ([OTF2_Archive](#) *archive)
Get a global snap reader.

J.2 OTF2_Archive.h File Reference

- [OTF2_ErrorCode](#) [OTF2_Archive_GetMachineName](#) ([OTF2_Archive](#) *archive, char **machineName)
Get machine name.
- [OTF2_MarkerReader](#) * [OTF2_Archive_GetMarkerReader](#) ([OTF2_Archive](#) *archive)
Get a marker reader.
- [OTF2_MarkerWriter](#) * [OTF2_Archive_GetMarkerWriter](#) ([OTF2_Archive](#) *archive)
Get a marker writer.
- [OTF2_ErrorCode](#) [OTF2_Archive_GetMasterSlaveMode](#) ([OTF2_Archive](#) *archive, [OTF2_MasterSlaveMode](#) *masterOrSlave)
Get master slave mode.
- [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.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ThumbWriter](#) * [OTF2_Archive_GetThumbWriter](#) ([OTF2_Archive](#) *archive, const char *name, const char *description, [OTF2_ThumbnailType](#) type, uint32_t numberOfSamples, uint32_t numberOfMetrics, const uint64_t *refsToDefs)

Get a thumb writer.

- [OTF2_ErrorCode](#) [OTF2_Archive_GetTraceId](#) ([OTF2_Archive](#) *archive, uint64_t *id)

Get the identifier of the trace file.

- [OTF2_ErrorCode](#) [OTF2_Archive_GetVersion](#) ([OTF2_Archive](#) *archive, uint8_t *major, uint8_t *minor, uint8_t *bugfix)

Get format version.

- [OTF2_Archive](#) * [OTF2_Archive_Open](#) (const char *archivePath, const char *archiveName, const [OTF2_FileMode](#) fileMode, const uint64_t chunkSizeEvents, const uint64_t chunkSizeDefs, const [OTF2_FileSubstrate](#) fileSubstrate, const [OTF2_Compression](#) compression)

Create a new archive.

- [OTF2_ErrorCode](#) [OTF2_Archive_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_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_SetFileSionCallbacks](#) ([OTF2_Archive](#) *archive, const [OTF2_FileSionCallbacks](#) *fileSionCallbacks, void *fileSionData)

Set the SION callbacks for the archive.

- [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_SetMasterSlaveMode](#) ([OTF2_Archive](#) *archive, [OTF2_MasterSlaveMode](#) masterOrSlave)

Set master slave mode.

- [OTF2_ErrorCode](#) [OTF2_Archive_SetMemoryCallbacks](#) ([OTF2_Archive](#) *archive, const [OTF2_MemoryCallbacks](#) *memoryCallbacks, void *memoryData)

Set the memory callbacks for the archive.

J.2 OTF2_Archive.h File Reference

- [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_SwitchFileMode](#) ([OTF2_Archive](#) *archive, [OTF2_FileMode](#) newFileMode)
Switch file mode of the archive.

J.2.1 Detailed Description

Writing interface for OTF2 archives.

Maintainer:

Michael Wagner <michael.wagner@zih.tu-dresden.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.2.2 Define Documentation

J.2.2.1 `#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.

J.2.2.2 `#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.

J.2.3 Typedef Documentation

J.2.3.1 `typedef struct OTF2_Archive_struct OTF2_Archive`

Keeps all meta-data for an OTF2 archive.

APPENDIX J. FILE DOCUMENTATION

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.

J.2.3.2 `typedef uint8_t OTF2_MasterSlaveMode`

Defines whether a location is master or slave.

The master of creates the directory layout and writes the anchor file. Therefore, only one archive handle can be the master, e.g. the MPI rank 0. All other archive handles must be defined as slaves.

Please see `OTF2_MasterSlaveMode_enum` for a description of available values.

J.2.4 Enumeration Type Documentation

J.2.4.1 `enum OTF2_MasterSlaveMode_enum`

Defines whether a location is master or slave.

Enumerator:

OTF2_SLAVE Location is slave.

OTF2_MASTER Location is master.

J.2.5 Function Documentation

J.2.5.1 `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.

J.2 OTF2_Archive.h File Reference

J.2.5.2 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.

Returns

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

J.2.5.3 OTF2_ErrorCode 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.

J.2.5.4 OTF2_ErrorCode 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.

Returns

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

APPENDIX J. FILE DOCUMENTATION

J.2.5.5 **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.

J.2.5.6 **OTF2_ErrorCode** **OTF2_Archive_CloseGlobalDefReader** (**OTF2_Archive** * *archive*, **OTF2_GlobalDefReader** * *globalDefReader*)

Closes the global definition reader.

Parameters

<i>archive</i>	Archive handle.
<i>globalDefReader</i>	The global definition reader.

Returns

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

J.2.5.7 **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.

J.2 OTF2_Archive.h File Reference

Returns

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

J.2.5.8 **OTF2_StatusCode** OTF2_Archive_CloseGlobalSnapReader (OTF2_Archive * *archive*, OTF2_GlobalSnapReader * *globalSnapReader*)

Close the opened global snapshot 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.

J.2.5.9 **OTF2_StatusCode** 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.

APPENDIX J. FILE DOCUMENTATION

J.2.5.10 **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.

J.2.5.11 **OTF2_ErrorCode** **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

J.2.5.12 **OTF2_ErrorCode** **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.

J.2 OTF2_Archive.h File Reference

Since

Version 1.2

Returns

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

J.2.5.13 `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.

J.2.5.14 `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.

APPENDIX J. FILE DOCUMENTATION

J.2.5.15 **OTF2_ErrorCode** **OTF2_Archive_GetCompression** (**OTF2_Archive** * *archive*, **OTF2_Compression** * *compression*)

Get compression mode (none or zlib)

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.

J.2.5.16 **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.

J.2.5.17 **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.

J.2 OTF2_Archive.h File Reference

J.2.5.18 `OTF2_DefWriter* OTF2_Archive_GetDefWriter (OTF2_Archive * archive,
OTF2_LocationRef location)`

Get a local definition writer.

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.

J.2.5.19 `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.

J.2.5.20 `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.

APPENDIX J. FILE DOCUMENTATION

J.2.5.21 **OTF2_EvtWriter*** **OTF2_Archive_GetEvtWriter** (**OTF2_Archive** * *archive*,
OTF2_LocationRef *location*)

Get a local event writer.

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.

J.2.5.22 **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.

J.2.5.23 **OTF2_GlobalDefReader*** **OTF2_Archive_GetGlobalDefReader** (
OTF2_Archive * *archive*)

Get a global definition reader.

Parameters

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

Returns

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

J.2 OTF2_Archive.h File Reference

J.2.5.24 OTF2_GlobalDefWriter* OTF2_Archive_GetGlobalDefWriter (OTF2_Archive * *archive*)

Get a global definition writer.

Parameters

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

Returns

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

J.2.5.25 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.

J.2.5.26 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.

APPENDIX J. FILE DOCUMENTATION

J.2.5.27 **OTF2_ErrorCode** **OTF2_Archive_GetMachineName** (**OTF2_Archive** *
archive, char ** *machineName*)

Get machine name.

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.

J.2.5.28 **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.

J.2.5.29 **OTF2_MarkerWriter*** **OTF2_Archive_GetMarkerWriter** (**OTF2_Archive** *
archive)

Get a marker writer.

Parameters

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

Since

Version 1.2

J.2 OTF2_Archive.h File Reference

Returns

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

J.2.5.30 `OTF2_ErrorCode OTF2_Archive.GetMasterSlaveMode (OTF2_Archive *
archive, OTF2_MasterSlaveMode * masterOrSlave)`

Get master slave mode.

Parameters

	<i>archive</i>	Archive handle.
out	<i>mas- terOrSlave</i>	Return pointer to the master slave mode.

Returns

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

J.2.5.31 `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>num- berOfDefi- nitions</i>	Return pointer to the number of global definitions.

Returns

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

J.2.5.32 `OTF2_ErrorCode OTF2_Archive.GetNumberOfLocations (OTF2_Archive
* archive, uint64_t * numberOfLocations)`

Get the number of locations.

Parameters

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

APPENDIX J. FILE DOCUMENTATION

out	<i>numberOfLocations</i>	Return pointer to the number of locations.
-----	--------------------------	--

Returns

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

J.2.5.33 **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.

J.2.5.34 **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.

J.2 OTF2_Archive.h File Reference

J.2.5.35 `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

J.2.5.36 `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.

J.2.5.37 `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.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.2

Returns

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

J.2.5.38 **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.

J.2.5.39 **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.

J.2 OTF2_Archive.h File Reference

J.2.5.40 **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.

Since

Version 1.2

Returns

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

J.2.5.41 **OTF2_ErrorCode** **OTF2_Archive_GetTraceId** (**OTF2_Archive ****archive*,
uint64_t **id*)

Get the identifier of the trace file.

Note

This call is only allowed when the archive was opened with mode **OTF2_FILEMODE_READ**.

Parameters

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

Returns

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

APPENDIX J. FILE DOCUMENTATION

J.2.5.42 `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.

J.2.5.43 `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 <i>OTF2_FILEMODE_WRITE</i> or <i>OTF2_FILEMODE_READ</i> .
<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 <i>OTF2_CHUNK_SIZE_EVENTS_DEFAULT</i> . In reading mode this value is ignored because the correct chunk size is extracted from the anchor file.

J.2 OTF2_Archive.h File Reference

<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 <i>OTF2_CHUNK_SIZE_DEFINITIONS_DEFAULT</i> . 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 <i>OTF2_SUBSTRATE_POSIX</i> to use the standard Posix interface, <i>OTF2_SUBSTRATE_SION</i> to use an installed SION library to store multiple logical files into fewer or one physical file, and <i>OTF2_SUBSTRATE_NONE</i> 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 <i>OTF2_COMPRESSION_ZLIB</i> to use an installed zlib and <i>OTF2_COMPRESSION_NONE</i> to disable compression. In reading mode this value is ignored because the correct file compression is extracted from the anchor file.

Returns

Returns an archive handle if successful, NULL otherwise.

J.2.5.44 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.

APPENDIX J. FILE DOCUMENTATION

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

J.2.5.45 `OTF2_ErrorCode OTF2_Archive.SetCreator (OTF2_Archive * archive,
const char * creator)`

Set creator.

Sets information about the creator of the trace archive. This value is optional. It only needs to be set for an archive handle marked as 'master' or does not need to be set at all.

Parameters

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

Returns

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

J.2.5.46 `OTF2_ErrorCode OTF2_Archive.SetDescription (OTF2_Archive * archive,
const char * description)`

Set a description.

Sets a description for a trace archive. This value is optional. It only needs to be set for an archive handle marked as 'master' or does not need to be set at all.

Parameters

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

Returns

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

J.2 OTF2_Archive.h File Reference

J.2.5.47 `OTF2_ErrorCode OTF2_Archive.SetFileSionCallbacks (OTF2_Archive * archive, const OTF2_FileSionCallbacks * fileSionCallbacks, void * fileSionData)`

Set the SION callbacks for the archive.

Parameters

<i>archive</i>	Archive handle.
<i>fileSion-Callbacks</i>	Struct holding the SION callback functions.
<i>fileSion-Data</i>	Data passed to the SION callbacks in the <code>userData</code> argument.

Returns

OTF2_ErrorCode, or error code.

J.2.5.48 `OTF2_ErrorCode 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_ErrorCode, or error code.

J.2.5.49 `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.

APPENDIX J. FILE DOCUMENTATION

Parameters

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

Returns

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

J.2.5.50 `OTF2_StatusCode OTF2_Archive.SetMasterSlaveMode (OTF2_Archive *
archive, OTF2_MasterSlaveMode masterOrSlave)`

Set master slave mode.

Sets master slave mode for a location. If `OTF2_MASTER` is passed, the location creates the directory structure for the trace files to store. Therefore, exactly one location can be master, all other locations must be slaves.

Please note: This call is only allowed in writing mode.

Parameters

<i>archive</i>	Archive handle.
<i>masterOrSlave</i>	Master or slave. Available values are <i>OTF2_MASTER</i> and <i>OTF2_SLAVE</i> .

Returns

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

J.2.5.51 `OTF2_StatusCode 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.

J.2 OTF2_Archive.h File Reference

Returns

OTF2_ErrorCode, or error code.

J.2.5.52 OTF2_ErrorCode OTF2_Archive.SetNumberOfSnapshots (OTF2_Archive * *archive*, uint32_t *number*)

Set 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.

J.2.5.53 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 <i>name</i> 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

J.2.5.54 *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 and in this case, the master/slave mode is reset and must be set again with *OTF2_Archive.SetMasterSlaveMode*. 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

J.3 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>
```

J.3 OTF2_AttributeList.h File Reference

```
#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.

APPENDIX J. FILE DOCUMENTATION

- [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_TYPE_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.

J.3 OTF2_AttributeList.h File Reference

- [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)

APPENDIX J. FILE DOCUMENTATION

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.

J.3 OTF2_AttributeList.h File Reference

- [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.

J.3.1 Detailed Description

This layer enables dynamic appending of arbitrary attributes to any type of event record.

Source Template:

template/OTF2_AttributeList.tmpl.h

Maintainer:

Michael Wagner <michael.wagner@zih.tu-dresden.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

APPENDIX J. FILE DOCUMENTATION

J.3.2 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, ... );
```

J.3.3 Function Documentation

J.3.3.1 **OTF2_ErrorCode OTF2_AttributeList_AddAttribute (OTF2_AttributeList * *attributeList*, OTF2_AttributeRef *attribute*, OTF2_Type *type*, OTF2_AttributeValue *attributeValue*)**

Add an attribute to an attribute list.

Adds an attribute to an attribute list. If the attribute already exists, it fails and returns an error.

Parameters

<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.

J.3 OTF2_AttributeList.h File Reference

Returns

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

J.3.3.2 **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.

J.3.3.3 **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.

J.3.3.4 **OTF2_ErrorCode** **OTF2_AttributeList_AddDouble** (**OTF2_AttributeList** *
 attributeList, **OTF2_AttributeRef** *attribute*, **double** *float64Value*)

Add an OTF2_TYPE_DOUBLE attribute to an attribute list.

APPENDIX J. FILE DOCUMENTATION

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.

J.3.3.5 **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.

J.3.3.6 **OTF2_ErrorCode** **OTF2_AttributeList_AddGroupRef** (**OTF2_AttributeList** * *attributeList*, **OTF2_AttributeRef** *attribute*, **OTF2_GroupRef** *groupRef*)

Add an OTF2_TYPE_GROUP attribute to an attribute list.

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.

J.3 OTF2_AttributeList.h File Reference

Returns

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

J.3.3.7 **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.

J.3.3.8 **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.

J.3.3.9 **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*.

APPENDIX J. FILE DOCUMENTATION

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.

J.3.3.10 **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.

J.3.3.11 **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.
<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.

J.3 OTF2_AttributeList.h File Reference

J.3.3.12 **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.

J.3.3.13 **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.

J.3.3.14 **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*.

APPENDIX J. FILE DOCUMENTATION

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.

J.3.3.15 **OTF2_StatusCode** **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.

J.3.3.16 **OTF2_StatusCode** **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.

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.

J.3 OTF2_AttributeList.h File Reference

Returns

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

J.3.3.17 `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.

J.3.3.18 `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.

J.3.3.19 `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*.

APPENDIX J. FILE DOCUMENTATION

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.

J.3.3.20 `OTF2_StatusCode 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.

J.3.3.21 `OTF2_StatusCode 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.

Returns

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

J.3 OTF2_AttributeList.h File Reference

J.3.3.22 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.

J.3.3.23 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.

J.3.3.24 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.

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.

APPENDIX J. FILE DOCUMENTATION

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.

J.3.3.25 **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.

J.3.3.26 **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.

Returns

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

J.3 OTF2_AttributeList.h File Reference

J.3.3.27 `OTF2_StatusCode 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.

J.3.3.28 `OTF2_StatusCode 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.

J.3.3.29 `OTF2_StatusCode 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*.

APPENDIX J. FILE DOCUMENTATION

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.

J.3.3.30 `OTF2_StatusCode 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.

J.3.3.31 `OTF2_StatusCode OTF2_AttributeList.GetInt32 (const OTF2_AttributeList
* attributeList, OTF2_AttributeRef attribute, int32_t * int32Value)`

Get an OTF2_TYPE_INT32 attribute from an attribute list by attribute ID.

Convenient function around *OTF2_AttributeList_GetAttributeByID*.

Parameters

	<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.

J.3 OTF2_AttributeList.h File Reference

J.3.3.32 **OTF2_StatusCode** 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.

J.3.3.33 **OTF2_StatusCode** 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.

J.3.3.34 **OTF2_StatusCode** 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.
--	----------------------	------------------------

APPENDIX J. FILE DOCUMENTATION

	<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.

J.3.3.35 `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.

J.3.3.36 `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.
----------------------	------------------------

Returns

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

J.3 OTF2_AttributeList.h File Reference

J.3.3.37 `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.

J.3.3.38 `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.

J.3.3.39 `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*.

APPENDIX J. FILE DOCUMENTATION

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.

J.3.3.40 `OTF2_StatusCode 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.

J.3.3.41 `OTF2_StatusCode 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.

J.3 OTF2_AttributeList.h File Reference

Returns

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

J.3.3.42 `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.

J.3.3.43 `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*.

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.

APPENDIX J. FILE DOCUMENTATION

J.3.3.44 `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.

J.3.3.45 `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.

J.3.3.46 `OTF2_AttributeList* OTF2_AttributeList_New (void)`

Create a new attribute list handle.

Returns

Returns a handle to the attribute list if successful, NULL otherwise.

J.3 OTF2_AttributeList.h File Reference

J.3.3.47 **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.

J.3.3.48 **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.

J.3.3.49 **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.

J.3.3.50 `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.

J.4 OTF2_Callbacks.h File Reference

This header file provides all user callbacks.

```
#include <stdio.h>
#include <stdbool.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
```

Data Structures

- struct [OTF2_FileSionCallbacks](#)
Structure holding the SION callbacks.
- struct [OTF2_FlushCallbacks](#)
Structure holding the flush callbacks.
- struct [OTF2_MemoryCallbacks](#)
Structure holding the memory callbacks.

Typedefs

- typedef int(* [OTF2_FileSionClose](#))(void *userData, [OTF2_FileType](#) file-
Type, [OTF2_LocationRef](#) location, int sid)

J.4 OTF2_Callbacks.h File Reference

Callbacks to wrap `sion_parclose_mpi()` for the OTF2 SION substrate.

- typedef [OTF2_ErrorCode](#)(* [OTF2_FileSionGetRank](#))(void *userData, [OTF2_FileType](#) fileType, [OTF2_LocationRef](#) location, int32_t *rank)

Provides location->rank translation, when using the SION substrate.

- typedef int(* [OTF2_FileSionOpen](#))(void *userData, [OTF2_FileType](#) fileType, [OTF2_LocationRef](#) location, const char *fname, const char *fileMode, long long int *chunkSize, int *fsblkSize, FILE **filePtr)

Callbacks to wrap `sion_paropen_mpi()` for the OTF2 SION substrate. Every parameter that can be given by OTF2 is named equally like the according parameter of `sion_paropen_mpi()`. Therefore, these given parameters MUST be given to SION.

- 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.

J.4.1 Detailed Description

This header file provides all user callbacks.

Maintainer:

Michael Wagner <michael.wagner@zih.tu-dresden.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

APPENDIX J. FILE DOCUMENTATION

J.4.2 Typedef Documentation

J.4.2.1 `typedef int(* OTF2_FileSionClose)(void *userData, OTF2_FileType
fileType, OTF2_LocationRef location, int sid)`

Callbacks to wrap `sion_parclose_mpi()` for the OTF2 SION substrate.

Parameters

<i>userData</i>	Data passed to the call OTF2_Archive_SetFileSionCallbacks .
<i>fileType</i>	The file type for which the file close is called.
<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>sid</i>	Sion file handle.

Returns

Return value of `sion_parclose_mpi()`

J.4.2.2 `typedef OTF2_ErrorCode(* OTF2_FileSionGetRank)(void *userData,
OTF2_FileType fileType, OTF2_LocationRef location, int32_t *rank)`

Provides location->rank translation, when using the SION substrate.

In case no `OTF2_FileSionOpen` and no `OTF2_FileSionClose` callback is given, the SION substrate still needs information what rank the current location has.

Parameters

	<i>userData</i>	Data passed to the call OTF2_Archive_SetFileSionCallbacks .
	<i>fileType</i>	The file type for which the file close is called.
	<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).
out	<i>rank</i>	The associated MPI rank for the <code>location</code> .

Returns

[OTF2_SUCCESS](#), or error code.

J.4 OTF2_Callbacks.h File Reference

J.4.2.3 `typedef int(* OTF2_FileSionOpen)(void *userData, OTF2_FileType fileType, OTF2_LocationRef location, const char *fname, const char *fileMode, long long int *chunkSize, int *fsblkSize, FILE **filePtr)`

Callbacks to wrap `sion_paropen_mpi()` for the OTF2 SION substrate. Every parameter that can be given by OTF2 is named equally like the the according parameter of `sion_paropen_mpi()`. Therefore, these given parameters **MUST** be given to SION.

Parameters

	<i>userData</i>	Data passed to the call OTF2_Archive_SetFileSionCallbacks .
	<i>fileType</i>	The file type for which the file open is called.
	<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>fname</i>	Name of file, should equal on all tasks.
	<i>fileMode</i>	Like the type parameter of <code>fopen</code> .
in, out	<i>chunkSize</i>	Requested space for this task.
in, out	<i>fsblkSize</i>	Blocksize of filesystem, must be equal on all processors.
out	<i>filePtr</i>	Filepointer for this task.

Returns

sion file handle integer (0, ...) -1 if error occurred

J.4.2.4 `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 behaviour 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).

APPENDIX J. FILE DOCUMENTATION

<i>perBufferData</i>	A writeable 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.

J.4.2.5 `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 behaviour 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 writeable 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.

J.4.2.6 `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.

J.5 OTF2_Definitions.h File Reference

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.

J.4.2.7 `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](#).

J.5 OTF2_Definitions.h File Reference

Data types used in the definition records.

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
```

Typedefs

- 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](#).

J.5 OTF2_Definitions.h File Reference

Enumerations

- 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 possible definitions of type LocationGroup.
- enum `OTF2_LocationGroupType_enum` {
 `OTF2_LOCATION_GROUP_TYPE_UNKNOWN` = 0,
 `OTF2_LOCATION_GROUP_TYPE_PROCESS` = 1 }
 List of possible definitions of type Location.
- 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 }
 Metric base types.
- 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`,

APPENDIX J. FILE DOCUMENTATION

`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 }
- 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,

J.5 OTF2_Definitions.h File Reference

- OTF2_METRIC_TYPE_RUSAGE = 2,
OTF2_METRIC_TYPE_USER = 3 }
- enum OTF2_MetricValueProperty_enum {
OTF2_METRIC_VALUE_ACCUMULATED = 0,
OTF2_METRIC_VALUE_ABSOLUTE = 1,
OTF2_METRIC_VALUE_RELATIVE = 2,
OTF2_METRIC_VALUE_MASK = 15 }
Information about whether the metric value is accumulated, absolute, or relative. Used for the lower half-byte of OTF2_MetricMode.
- enum OTF2_ParameterType_enum {
OTF2_PARAMETER_TYPE_STRING = 0,
OTF2_PARAMETER_TYPE_INT64 = 1,
OTF2_PARAMETER_TYPE_UINT64 = 2 }
List of possible for definitions of type Parameter.
- enum OTF2_RecorderKind_enum {
OTF2_RECORDER_KIND_UNKNOWN = 0,
OTF2_RECORDER_KIND_ABSTRACT = 1,
OTF2_RECORDER_KIND_CPU = 2,
OTF2_RECORDER_KIND_GPU = 3 }
List of possible kinds a MetricClass can be recorded by.
- enum OTF2_RegionFlag_enum {
OTF2_REGION_FLAG_NONE = 0,
OTF2_REGION_FLAG_DYNAMIC = (1 << 0),
OTF2_REGION_FLAG_PHASE = (1 << 1) }
List of possible flags to specify special characteristics of a Region.
- enum OTF2_RegionRole_enum {
OTF2_REGION_ROLE_UNKNOWN = 0,
OTF2_REGION_ROLE_FUNCTION = 1,
OTF2_REGION_ROLE_WRAPPER = 2,
OTF2_REGION_ROLE_LOOP = 3,
OTF2_REGION_ROLE_CODE = 4,
OTF2_REGION_ROLE_PARALLEL = 5,
OTF2_REGION_ROLE_SECTIONS = 6,
OTF2_REGION_ROLE_SECTION = 7,
OTF2_REGION_ROLE_WORKSHARE = 8,

OTF2_REGION_ROLE_SINGLE = 9,
OTF2_REGION_ROLE_SINGLE_SBLOCK = 10,
OTF2_REGION_ROLE_MASTER = 11,
OTF2_REGION_ROLE_CRITICAL = 12,
OTF2_REGION_ROLE_CRITICAL_SBLOCK = 13,
OTF2_REGION_ROLE_ATOMIC = 14,
OTF2_REGION_ROLE_BARRIER = 15,
OTF2_REGION_ROLE_IMPLICIT_BARRIER = 16,
OTF2_REGION_ROLE_FLUSH = 17,
OTF2_REGION_ROLE_ORDERED = 18,
OTF2_REGION_ROLE_ORDERED_SBLOCK = 19,
OTF2_REGION_ROLE_TASK = 20,
OTF2_REGION_ROLE_TASK_CREATE = 21,
OTF2_REGION_ROLE_TASK_WAIT = 22,
OTF2_REGION_ROLE_COLL_ONE2ALL = 23,
OTF2_REGION_ROLE_COLL_ALL2ONE = 24,
OTF2_REGION_ROLE_COLL_ALL2ALL = 25,
OTF2_REGION_ROLE_COLL_OTHER = 26,
OTF2_REGION_ROLE_FILE_IO = 27,
OTF2_REGION_ROLE_POINT2POINT = 28,
OTF2_REGION_ROLE_RMA = 29,
OTF2_REGION_ROLE_DATA_TRANSFER = 30,
OTF2_REGION_ROLE_ARTIFICIAL = 31 }

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 }

J.5 OTF2_Definitions.h File Reference

J.5.1 Detailed Description

Data types used in the definition records.

Source Template:

templates/OTF2_Definitions.tmpl.h

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.5.2 Enumeration Type Documentation

J.5.2.1 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 needs to be done when a group of type [OTF2_GROUP_TYPE_COMM_GROUP](#) has this flag.

J.5.2.2 enum OTF2_GroupType_enum

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 location IDs, which are MPI ranks. The size of this group should match the size of `MPI_COMM_WORLD`. Each entry in the list is a location ID, where the index of the entry is equal to the rank in `MPI_COMM_WORLD`. (Ie. 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_MPI_GROUP*.

Note: This does not makes sense in local definitions.

OTF2_GROUP_TYPE_COMM_GROUP MPI group.

OTF2_GROUP_TYPE_COMM_SELF Special group type to efficiently handle MPI self-like communicators.

J.5.2.3 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.

J.5.2.4 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.

J.5 OTF2_Definitions.h File Reference

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.

J.5.2.5 enum OTF2_MetricBase_enum

Metric base types.

Since

Version 1.0

Enumerator:

OTF2_BASE_BINARY Binary base.

OTF2_BASE_DECIMAL Decimal base.

J.5.2.6 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.

J.5.2.7 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.

J.5.2.8 enum OTF2_MetricScope_enum

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.

J.5.2.9 enum OTF2_MetricTiming_enum

Determines when the values have been collected or for which interval of time they are valid. Used for the upper half-byte of OTF2_MetricMode.

Since

Version 1.0

Enumerator:

OTF2_METRIC_TIMING_START Metric value belongs to the time interval since the beginning of the measurement.

J.5 OTF2_Definitions.h File Reference

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.

J.5.2.10 enum OTF2_MetricType_enum

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.

J.5.2.11 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.

J.5.2.12 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.

J.5.2.13 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 Only *MetricInstances* will record this metric class.

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*](#).

J.5 OTF2_Definitions.h File Reference

J.5.2.14 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.

J.5.2.15 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 J. 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.

J.6 OTF2_DefReader.h File Reference

J.5.2.16 enum OTF2_SystemTreeDomain_enum

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, ...)

J.6 OTF2_DefReader.h File Reference

This is the local definition reader, which reads location dependend definitions, and can also be used to get the mapping information from the local definition file. Local definitions are always assigned to a location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_DefReaderCallbacks.h>
```

Functions

- [OTF2_ErrorCode OTF2_DefReader_GetLocationID](#) (const OTF2_DefReader *reader, OTF2_LocationRef *location)
Get the location ID of this reader object.

- [OTF2_ErrorCode](#) [OTF2_DefReader_ReadDefinitions](#) ([OTF2_DefReader](#) *reader, uint64_t recordsToRead, uint64_t *recordsRead)
Reads the given number of records from the definition reader.
- [OTF2_ErrorCode](#) [OTF2_DefReader_SetCallbacks](#) ([OTF2_DefReader](#) *reader, const [OTF2_DefReaderCallbacks](#) *callbacks, void *userData)
Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

J.6.1 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.

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.6.2 Function Documentation

J.6.2.1 [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.

J.6 OTF2_DefReader.h File Reference

J.6.2.2 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

J.6.2.3 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. OTF2_DefReaderCallbacks_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.

J.7 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_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_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.

J.7 OTF2_DefReaderCallbacks.h File Reference

- 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_MappingTable)(void *userData, OTF2_MappingType mappingType, const OTF2_IdMap *idMap)

Function pointer definition for the callback which is triggered by a [MappingTable](#) definition record.

- typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_MetricClass)(void *userData, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_MetricMemberRef *metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)

Function pointer definition for the callback which is triggered by a [MetricClass](#) definition record.

- typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_MetricClassRecorder)(void *userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)

Function pointer definition for the callback which is triggered by a [MetricClass-Recorder](#) definition record.

- typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_MetricInstance)(void *userData, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)

Function pointer definition for the callback which is triggered by a [MetricInstance](#) definition record.

- typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_MetricMember)(void *userData, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_MetricType metricType, OTF2_MetricMode

metricMode, [OTF2_Type](#) valueType, [OTF2_MetricBase](#) metricBase, int64_t exponent, [OTF2_StringRef](#) unit)

Function pointer definition for the callback which is triggered by a [MetricMember](#) definition record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_DefReaderCallback_Parameter](#))(void *userData, [OTF2_ParameterRef](#) self, [OTF2_StringRef](#) name, [OTF2_ParameterType](#) parameterType)

Function pointer definition for the callback which is triggered by a [Parameter](#) definition record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_DefReaderCallback_Region](#))(void *userData, [OTF2_RegionRef](#) self, [OTF2_StringRef](#) name, [OTF2_StringRef](#) canonicalName, [OTF2_StringRef](#) description, [OTF2_RegionRole](#) regionRole, [OTF2_Paradigm](#) paradigm, [OTF2_RegionFlag](#) regionFlags, [OTF2_StringRef](#) sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)

Function pointer definition for the callback which is triggered by a [Region](#) definition record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_DefReaderCallback_RmaWin](#))(void *userData, [OTF2_RmaWinRef](#) self, [OTF2_StringRef](#) name, [OTF2_CommRef](#) comm)

Function pointer definition for the callback which is triggered by a [RmaWin](#) definition record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_DefReaderCallback_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)

J.7 OTF2_DefReaderCallbacks.h File Reference

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_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.
- [OTF2_ErrorCode](#) [OTF2_DefReaderCallbacks_SetLocationGroupCallback](#) ([OTF2_DefReaderCallbacks](#) *[defReaderCallbacks](#), [OTF2_DefReaderCallback_LocationGroup](#) [locationGroupCallback](#))
Registers the callback for the [LocationGroup](#) 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.

J.7 OTF2_DefReaderCallbacks.h File Reference

- [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.

J.7.1 Detailed Description

This defines the callbacks for the definition reader.

Source Template:

templates/OTF2_DefReaderCallbacks.tmpl.h

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.7.2 Typedef Documentation

J.7.2.1 `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_
Attribute)(void *userData, OTF2_AttributeRef self, OTF2_StringRef
name, OTF2_Type type)`

Function pointer definition for the callback which is triggered by a [Attribute](#) definition record.

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>type</i>	Type of the attribute value.

Since

Version 1.0

Returns

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

J.7.2.2 `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.

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>	References a Callpath definition.
<i>region</i>	References a Region definition.

Since

Version 1.0

J.7 OTF2_DefReaderCallbacks.h File Reference

Returns

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

J.7.2.3 `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.

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*](#).

J.7.2.4 `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

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

APPENDIX J. FILE DOCUMENTATION

<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*](#).

J.7.2.5 `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.

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 <code>OTF2_GROUP_TYPE_MPI_GROUP</code> or <code>OTF2_GROUP_TYPE_MPI_COMM_SELF</code> . References a Group 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 Comm definition.

Since

Version 1.0

Returns

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

J.7 OTF2_DefReaderCallbacks.h File Reference

J.7.2.6 `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.

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](#).

J.7.2.7 `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.

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.

APPENDIX J. FILE DOCUMENTATION

<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](#).

J.7.2.8 `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.

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>sys- temTreePar- ent</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](#).

J.7 OTF2_DefReaderCallbacks.h File Reference

J.7.2.9 `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](#).

J.7.2.10 `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>numberOf-Metrics</i>	Number of metrics within the set.

APPENDIX J. FILE DOCUMENTATION

<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](#).

J.7.2.11 `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 [MetricClassRecorder](#) definition record.

Parameters

<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](#).

J.7 OTF2_DefReaderCallbacks.h File Reference

J.7.2.12 `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](#).

J.7.2.13 `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

APPENDIX J. FILE DOCUMENTATION

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: int64_t, uint64_t, or double.
<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](#).

J.7.2.14 `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.

Parameters

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

J.7 OTF2_DefReaderCallbacks.h File Reference

<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](#).

J.7.2.15 `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.

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*](#).

J.7.2.16 `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.
<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*](#).

J.7.2.17 `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.

Parameters

J.7 OTF2_DefReaderCallbacks.h File Reference

<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](#).

J.7.2.18 `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.

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](#).

APPENDIX J. FILE DOCUMENTATION

J.7.2.19 `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.

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.

Since

Version 1.2

Returns

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

J.7.2.20 `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.

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

Since

Version 1.2

J.7 OTF2_DefReaderCallbacks.h File Reference

Returns

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

J.7.2.1 `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 OTF2_Reader_RegisterDefCallbacks or OTF2_DefReader_SetCallbacks .
-----------------	---

Returns

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

J.7.3 Function Documentation

J.7.3.1 `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 .
----------------------------	--

J.7.3.2 `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 .
----------------------------	--

APPENDIX J. FILE DOCUMENTATION

J.7.3.3 OTF2_DefReaderCallbacks* OTF2_DefReaderCallbacks.New (void)

Allocates a new struct for the definition callbacks.

Returns

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

J.7.3.4 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.

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.5 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 Callpath definitions.

Returns

[OTF2_SUCCESS](#) if successful

J.7 OTF2_DefReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.7.3.6 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 Callsite definitions.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.7.3.7 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetClockOffsetCallback
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_ClockOffset *clockOffsetCallback*)

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

Parameters

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

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.7.3.8 OTF2_ErrorCode OTF2.DefReaderCallbacks_SetCommCallback
(OTF2_DefReaderCallbacks * *defReaderCallbacks*,
OTF2_DefReaderCallback_Comm *commCallback*)

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

Parameters

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.9 OTF2_ErrorCode OTF2.DefReaderCallbacks_SetGroupCallback
(OTF2_DefReaderCallbacks * *defReaderCallbacks*,
OTF2_DefReaderCallback_Group *groupCallback*)

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

Parameters

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.10 OTF2_ErrorCode OTF2.DefReaderCallbacks_SetLocationCallback
(OTF2_DefReaderCallbacks * *defReaderCallbacks*,
OTF2_DefReaderCallback_Location *locationCallback*)

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

J.7 OTF2_DefReaderCallbacks.h File Reference

Parameters

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.11 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetLocationGroupCallback (OTF2_DefReaderCallbacks * defReaderCallbacks, OTF2_DefReaderCallback_LocationGroup locationGroupCallback)`

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

Parameters

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.12 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMappingTableCallback (OTF2_DefReaderCallbacks * defReaderCallbacks, OTF2_DefReaderCallback_MappingTable mappingTableCallback)`

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

Parameters

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

APPENDIX J. FILE DOCUMENTATION

<i>mappingTable-Callback</i>	Function which should be called for all MappingTable definitions.
------------------------------	---

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.13 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricClassCallback (OTF2_DefReaderCallbacks * defReaderCallbacks, OTF2_DefReaderCallback_MetricClass metricClassCallback)`

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

Parameters

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.14 `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetMetricClassRecorderCallback (OTF2_DefReaderCallbacks * defReaderCallbacks, OTF2_DefReaderCallback_MetricClassRecorder metricClassRecorderCallback)`

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

Parameters

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

J.7 OTF2_DefReaderCallbacks.h File Reference

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.15 `OTF2_ErrorCode 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>metricIn- stanceCall- back</i>	Function which should be called for all MetricInstance definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.16 `OTF2_ErrorCode 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 J. FILE DOCUMENTATION

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.17 `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 Parameter definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.18 `OTF2_StatusCode OTF2_DefReaderCallbacks_SetRegionCallback`
(`OTF2_DefReaderCallbacks * defReaderCallbacks`,
`OTF2_DefReaderCallback_Region regionCallback`)

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

Parameters

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

J.7 OTF2_DefReaderCallbacks.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful

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

J.7.3.19 `OTF2_StatusCode OTF2_DefReaderCallbacks_SetRmaWinCallback`
(`OTF2_DefReaderCallbacks * defReaderCallbacks`,
`OTF2_DefReaderCallback_RmaWin rmaWinCallback`)

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

Parameters

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

Returns

[*OTF2_SUCCESS*](#) if successful

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

J.7.3.20 `OTF2_StatusCode OTF2_DefReaderCallbacks_SetStringCallback`
(`OTF2_DefReaderCallbacks * defReaderCallbacks`,
`OTF2_DefReaderCallback_String stringCallback`)

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

Parameters

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

Returns

[*OTF2_SUCCESS*](#) if successful

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

APPENDIX J. FILE DOCUMENTATION

J.7.3.21 **OTF2_ErrorCode** **OTF2_DefReaderCallbacks_SetSystemTreeNodeCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_SystemTreeNode *systemTreeNodeCallback*)

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

Parameters

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.22 **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>systemTreeNodeDomainCallback</i>	Function which should be called for all SystemTreeNodeDomain definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.7 OTF2_DefReaderCallbacks.h File Reference

J.7.3.23 **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>systemTreeNodePropertyCallback</i>	Function which should be called for all SystemTreeNodeProperty definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.7.3.24 **OTF2_***ErrorCode* **OTF2_DefReaderCallbacks_SetUnknownCallback**
(**OTF2_DefReaderCallbacks** * *defReaderCallbacks*,
OTF2_DefReaderCallback_Unknown *unknownCallback*)

Registers the callback for an unknown definition.

Parameters

<i>defReader-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

J.8 OTF2_DefWriter.h File Reference

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

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_IdMap.h>
```

Typedefs

- typedef struct OTF2_DefWriter_struct [OTF2_DefWriter](#)
Handle definition for the external definition writer.

Functions

- [OTF2_ErrorCode OTF2_DefWriter_GetLocationID](#) (const [OTF2_DefWriter](#) *writer, [OTF2_LocationRef](#) *location)
Returns the location ID of the location which is related to the writer object.
- [OTF2_ErrorCode OTF2_DefWriter_WriteAttribute](#) ([OTF2_DefWriter](#) *writer, [OTF2_AttributeRef](#) self, [OTF2_StringRef](#) name, [OTF2_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_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,

J.8 OTF2_DefWriter.h File Reference

[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_WriteMappingTable](#) ([OTF2_DefWriter](#) *writer, [OTF2_MappingType](#) mappingType, [const OTF2_IdMap](#) *idMap)

Writes a MappingTable definition record into the DefWriter.

- [OTF2_ErrorCode](#) [OTF2_DefWriter_WriteMetricClass](#) ([OTF2_DefWriter](#) *writer, [OTF2_MetricRef](#) self, [uint8_t](#) numberOfMetrics, [const OTF2_MetricMemberRef](#) *metricMembers, [OTF2_MetricOccurrence](#) metricOccurrence, [OTF2_RecorderKind](#) recorderKind)

Writes a MetricClass definition record into the DefWriter.

- [OTF2_ErrorCode](#) [OTF2_DefWriter_WriteMetricClassRecorder](#) ([OTF2_DefWriter](#) *writer, [OTF2_MetricRef](#) metricClass, [OTF2_LocationRef](#) recorder)

Writes a MetricClassRecorder definition record into the DefWriter.

- [OTF2_ErrorCode](#) [OTF2_DefWriter_WriteMetricInstance](#) ([OTF2_DefWriter](#) *writer, [OTF2_MetricRef](#) self, [OTF2_MetricRef](#) metricClass, [OTF2_LocationRef](#) recorder, [OTF2_MetricScope](#) metricScope, [uint64_t](#) scope)

Writes a MetricInstance definition record into the DefWriter.

- [OTF2_ErrorCode](#) [OTF2_DefWriter_WriteMetricMember](#) ([OTF2_DefWriter](#) *writer, [OTF2_MetricMemberRef](#) self, [OTF2_StringRef](#) name, [OTF2_StringRef](#) description, [OTF2_MetricType](#) metricType, [OTF2_MetricMode](#) metricMode, [OTF2_Type](#) valueType, [OTF2_MetricBase](#) metricBase, [int64_t](#) exponent, [OTF2_StringRef](#) unit)

Writes a MetricMember definition record into the DefWriter.

- [OTF2_ErrorCode](#) [OTF2_DefWriter_WriteParameter](#) ([OTF2_DefWriter](#) *writer, [OTF2_ParameterRef](#) self, [OTF2_StringRef](#) name, [OTF2_ParameterType](#) parameterType)

Writes a Parameter definition record into the DefWriter.

- [OTF2_ErrorCode](#) [OTF2_DefWriter_WriteRegion](#) ([OTF2_DefWriter](#) *writer, [OTF2_RegionRef](#) self, [OTF2_StringRef](#) name, [OTF2_StringRef](#) canonicalName, [OTF2_StringRef](#) description, [OTF2_RegionRole](#) regionRole, [OTF2_Paradigm](#) paradigm, [OTF2_RegionFlag](#) regionFlags, [OTF2_StringRef](#) sourceFile, [uint32_t](#) beginLineNumber, [uint32_t](#) endLineNumber)

APPENDIX J. FILE DOCUMENTATION

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_SystemTreeNodeDomain](#) systemTreeNodeDomain)

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.

J.8.1 Detailed Description

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

Source Template:

templates/OTF2_DefWriter.tmpl.h

J.8.2 Function Documentation

J.8.2.1 [OTF2_ErrorCode](#) [OTF2_DefWriter_GetLocationID](#) ([const](#) [OTF2_DefWriter](#) * *writer*, [OTF2_LocationRef](#) * *location*)

Returns the location ID of the location which is related to the writer object.

Parameters

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

J.8 OTF2_DefWriter.h File Reference

Returns

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

J.8.2.2 **OTF2_ErrorCode** OTF2_DefWriter_WriteAttribute (OTF2_DefWriter *
writer, OTF2_AttributeRef *self*, OTF2_StringRef *name*, OTF2_Type
type)

Writes a Attribute definition record into the DefWriter.

Parameters

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

Since

Version 1.0

Returns

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

J.8.2.3 **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.

Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this Callpath definition.
<i>parent</i>	References a Callpath definition.
<i>region</i>	References a Region definition.

Since

Version 1.0

Returns

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

APPENDIX J. FILE DOCUMENTATION

J.8.2.4 **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.

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.

J.8.2.5 **OTF2_ErrorCode** **OTF2.DefWriter.WriteClockOffset** (**OTF2_DefWriter** * *writer*, **OTF2_TimeStamp** *time*, **int64_t** *offset*, **double** *standardDeviation*)

Writes a ClockOffset definition record into the DefWriter.

Clock offsets are used for clock corrections.

Parameters

<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.

J.8 OTF2_DefWriter.h File Reference

J.8.2.6 `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.

Parameters

<i>writer</i>	Writer object.
<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 <code>OTF2_GROUP_TYPE_MPI_GROUP</code> or <code>OTF2_GROUP_TYPE_MPI_COMM_SELF</code> . References a Group definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <code>OTF2_UNDEFINED_COMM</code> to indicate no parent. References a Comm definition.

Since

Version 1.0

Returns

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

J.8.2.7 `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.

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.

APPENDIX J. FILE DOCUMENTATION

<i>num-berOfMembers</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.

J.8.2.8 **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.

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>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_SUCCESS*](#) if successful, an error code if an error occurs.

J.8 OTF2_DefWriter.h File Reference

J.8.2.9 **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.

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.

J.8.2.10 **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

APPENDIX J. FILE DOCUMENTATION

Returns

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

J.8.2.11 `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 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.

J.8.2.12 `OTF2_ErrorCode OTF2_DefWriter_WriteMetricClassRecorder (OTF2_DefWriter * writer, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)`

Writes a MetricClassRecorder definition record into the DefWriter.

Parameters

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

J.8 OTF2_DefWriter.h File Reference

<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.

J.8.2.13 `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

<i>writer</i>	Writer object.
<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_SUCCESS](#) if successful, an error code if an error occurs.

APPENDIX J. FILE DOCUMENTATION

J.8.2.14 `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 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: int64_t, uint64_t, or double.
<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.

J.8 OTF2_DefWriter.h File Reference

J.8.2.15 `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.

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.

J.8.2.16 `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.

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.

APPENDIX J. FILE DOCUMENTATION

<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.

J.8.2.17 `OTF2_ErrorCode OTF2_DefWriter_WriteRmaWin (OTF2_DefWriter
* writer, OTF2_RmaWinRef self, OTF2_StringRef name,
OTF2_CommRef comm)`

Writes a RmaWin definition record into the DefWriter.

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

Parameters

<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.

J.8 OTF2_DefWriter.h File Reference

J.8.2.18 `OTF2_ErrorCode OTF2_DefWriter_WriteString (OTF2_DefWriter * writer,
OTF2_StringRef self, const char * string)`

Writes a String definition record into the DefWriter.

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.

J.8.2.19 `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.

Parameters

<i>writer</i>	Writer object.
<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_SUCCESS](#) if successful, an error code if an error occurs.

APPENDIX J. FILE DOCUMENTATION

J.8.2.20 **OTF2_ErrorCode** **OTF2_DefWriter_WriteSystemTreeNodeDomain** (
 OTF2_DefWriter * *writer*, **OTF2_SystemTreeNodeRef** *systemTreeNode*,
 OTF2_SystemTreeDomain *systemTreeDomain*)

Writes a SystemTreeNodeDomain definition record into the DefWriter.

Parameters

<i>writer</i>	Writer object.
<i>systemTreeNode</i>	Parent SystemTreeNode definition to which this one is a supplementary definition. References a SystemTreeNode definition.

Since

Version 1.2

Returns

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

J.8.2.21 **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.

Parameters

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

Since

Version 1.2

Returns

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

J.9 OTF2_ErrorCodes.h File Reference

J.9 OTF2_ErrorCodes.h File Reference

Error codes and error handling.

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

Typedefs

- typedef [OTF2_ErrorCode](#)(* [OTF2_ErrorCallback](#))(void *userData, const char *file, uint64_t line, const char *function, [OTF2_ErrorCode](#) errorCode, const char *msgFormatString, va_list va)

Enumerations

- enum [OTF2_ErrorCode](#) {
 [OTF2_DEPRECATED](#) = -3,
 [OTF2_ABORT](#) = -2,
 [OTF2_WARNING](#) = -1,
 [OTF2_SUCCESS](#) = 0,
 [OTF2_ERROR_INVALID](#) = 1,
 [OTF2_ERROR_E2BIG](#),
 [OTF2_ERROR_EACCES](#),
 [OTF2_ERROR_EADDRNOTAVAIL](#),
 [OTF2_ERROR_EAFNOSUPPORT](#),
 [OTF2_ERROR_EAGAIN](#),
 [OTF2_ERROR_EALREADY](#),
 [OTF2_ERROR_EBADF](#),
 [OTF2_ERROR_EBADMSG](#),
 [OTF2_ERROR_EBUSY](#),
 [OTF2_ERROR_ECANCELED](#),
 [OTF2_ERROR_ECHILD](#),
 [OTF2_ERROR_ECONNREFUSED](#),
 [OTF2_ERROR_ECONNRESET](#),
 [OTF2_ERROR_EDEADLK](#),

OTF2_ERROR_EDESTADDRREQ,
OTF2_ERROR_EDOM,
OTF2_ERROR_EDQUOT,
OTF2_ERROR_EEXIST,
OTF2_ERROR_EFAULT,
OTF2_ERROR_EFBIG,
OTF2_ERROR_EINPROGRESS,
OTF2_ERROR_EINTR,
OTF2_ERROR_EINVAL,
OTF2_ERROR_EIO,
OTF2_ERROR_EISCONN,
OTF2_ERROR_EISDIR,
OTF2_ERROR_ELOOP,
OTF2_ERROR_EMFILE,
OTF2_ERROR_EMLINK,
OTF2_ERROR EMSGSIZE,
OTF2_ERROR_EMULTIHOP,
OTF2_ERROR_ENAMETOOLONG,
OTF2_ERROR_ENETDOWN,
OTF2_ERROR_ENETRESET,
OTF2_ERROR_ENETUNREACH,
OTF2_ERROR_ENFILE,
OTF2_ERROR_ENOBUFS,
OTF2_ERROR_ENODATA,
OTF2_ERROR_ENODEV,
OTF2_ERROR_ENOENT,
OTF2_ERROR_ENOEXEC,
OTF2_ERROR_ENOLCK,
OTF2_ERROR_ENOLINK,
OTF2_ERROR_ENOMEM,
OTF2_ERROR_ENOMSG,
OTF2_ERROR_ENOPROTOOPT,
OTF2_ERROR_ENOSPC,

J.9 OTF2_ErrorCodes.h File Reference

OTF2_ERROR_ENOSR,
OTF2_ERROR_ENOSTR,
OTF2_ERROR_ENOSYS,
OTF2_ERROR_ENOTCONN,
OTF2_ERROR_ENOTDIR,
OTF2_ERROR_ENOTEMPTY,
OTF2_ERROR_ENOTSOCK,
OTF2_ERROR_ENOTSUP,
OTF2_ERROR_ENOTTY,
OTF2_ERROR_ENXIO,
OTF2_ERROR_EOPNOTSUPP,
OTF2_ERROR_EOVERFLOW,
OTF2_ERROR_EPERM,
OTF2_ERROR_EPIPE,
OTF2_ERROR_EPROTO,
OTF2_ERROR_EPROTONOSUPPORT,
OTF2_ERROR_EPROTOTYPE,
OTF2_ERROR_ERANGE,
OTF2_ERROR_EROFS,
OTF2_ERROR_ESPIPE,
OTF2_ERROR_ESRCH,
OTF2_ERROR_ESTALE,
OTF2_ERROR_ETIME,
OTF2_ERROR_ETIMEDOUT,
OTF2_ERROR_ETXTBSY,
OTF2_ERROR_EWOULDBLOCK,
OTF2_ERROR_EXDEV,
OTF2_ERROR_END_OF_FUNCTION,
OTF2_ERROR_INVALID_CALL,
OTF2_ERROR_INVALID_ARGUMENT,
OTF2_ERROR_INVALID_RECORD,
OTF2_ERROR_INVALID_DATA,
OTF2_ERROR_INVALID_SIZE_GIVEN,

APPENDIX J. FILE DOCUMENTATION

```
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 }
```

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)`

J.9.1 Detailed Description

Error codes and error handling.

Maintainer:

Daniel Lorenz <d.lorenz@fz-juelich.de>

File Status:

ALPHA

Author

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

J.9 OTF2_ErrorCodes.h File Reference

J.9.2 Typedef Documentation

J.9.2.1 `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

J.9.3 Enumeration Type Documentation

J.9.3.1 `enum OTF2_ErrorCode`

This is the list of error codes for OTF2.

Enumerator:

OTF2_DEPRECATED Special marker for error messages which indicates an deprecation.

OTF2_ABORT Special marker when the application will be aborted.

OTF2_WARNING Special marker for error messages which are only warnings.

OTF2_SUCCESS Operation successful

OTF2_ERROR_INVALID Invalid error code

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

OTF2_ERROR_E2BIG The list of arguments is to long

OTF2_ERROR_EACCES Not enough rights

OTF2_ERROR_EADDRNOTAVAIL Address is not available

APPENDIX J. FILE DOCUMENTATION

OTF2_ERROR_EAFNOSUPPORT Address family is not supported
OTF2_ERROR_EAGAIN Resource temporary 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
OTF2_ERROR_EMLINK Too many links
OTF2_ERROR EMSGSIZE Message buffer is too small
OTF2_ERROR_EMULTIHOP Reserved
OTF2_ERROR_ENAMETOOLONG Filename is too 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 Too much opened files
OTF2_ERROR_ENOBUFS No buffer space available

J.9 OTF2_ErrorCodes.h File Reference

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 Can not execute binary

OTF2_ERROR_ENOLCK Locking failed

OTF2_ERROR_ENOLINK Reserved

OTF2_ERROR_ENOMEM Not enough main memory available

OTF2_ERROR_ENOMSG Message has not the expected type

OTF2_ERROR_ENOPROTOOPT This protocol is not available

OTF2_ERROR_ENOSPC No space left on device

OTF2_ERROR_ENOSR No stream available

OTF2_ERROR_ENOSTR This is not a stream

OTF2_ERROR_ENOSYS Requested function is not implemented

OTF2_ERROR_ENOTCONN Socket is not connected

OTF2_ERROR_ENOTDIR This is not an directory

OTF2_ERROR_ENOTEMPTY This directory is not empty

OTF2_ERROR_ENOTSOCK No socket

OTF2_ERROR_ENOTSUP This operation is not supported

OTF2_ERROR_ENOTTY This IOCTL is not supported by the device

OTF2_ERROR_ENXIO Device is not yet configured

OTF2_ERROR_EOPNOTSUPP Operation is not supported by this socket

OTF2_ERROR_EOVERFLOW Value is to long for the datatype

OTF2_ERROR_EPERM Operation is not permitted

OTF2_ERROR_EPIPE Broken pipe

OTF2_ERROR_EPROTO Protocoll error

OTF2_ERROR_EPROTONOSUPPORT Protocoll is not supported

OTF2_ERROR_EPROTOTYPE Wrong protocoll type for this socket

OTF2_ERROR_ERANGE Value is out of range

OTF2_ERROR_EROFS Filesystem is read only

OTF2_ERROR_ESPIPE This seek is not allowed

OTF2_ERROR_ESRCH No matching process found

OTF2_ERROR_ESTALE Reserved

OTF2_ERROR_ETIME Timeout in file stream or IOCTL

OTF2_ERROR_ETIMEDOUT Connection timed out

APPENDIX J. FILE DOCUMENTATION

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 can not be used

OTF2_ERROR_UNKNOWN_TYPE The given type is not known

OTF2_ERROR_INTEGRITY_FAULT The structural integrity is not given

OTF2_ERROR_MEM_FAULT This could not be done with the given memory

OTF2_ERROR_MEM_ALLOC_FAILED Memory allocation failed

OTF2_ERROR_PROCESSED_WITH_FAULTS An error appeared when data was processed

OTF2_ERROR_INDEX_OUT_OF_BOUNDS Index out of bounds

OTF2_ERROR_INVALID_LINENO Invalid source code line number

OTF2_ERROR_END_OF_BUFFER End of buffer/file reached

OTF2_ERROR_FILE_INTERACTION Invalid file operation

OTF2_ERROR_FILE_CAN_NOT_OPEN Unable to open file

OTF2_ERROR_INTERRUPTED_BY_CALLBACK Record reading interrupted by reader callback

OTF2_ERROR_PROPERTY_NAME_INVALID Property name does not conform to the naming scheme

OTF2_ERROR_PROPERTY_EXISTS Property already exists

OTF2_ERROR_PROPERTY_NOT_FOUND Property not found found in this archive

OTF2_ERROR_PROPERTY_VALUE_INVALID Property value does not have the expected value

OTF2_ERROR_FILE_COMPRESSION_NOT_SUPPORTED Missing library support for requested compression mode

OTF2_ERROR_DUPLICATE_MAPPING_TABLE Multiple definitions for the same mapping type

OTF2_ERROR_INVALID_FILE_MODE_TRANSITION File mode transition not permitted

J.9 OTF2_ErrorCodes.h File Reference

J.9.4 Function Documentation

J.9.4.1 `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.

J.9.4.2 `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.

J.9.4.3 `OTF2_ErrorCallback OTF2_Error_RegisterCallback (OTF2_ErrorCallback errorCallbackIn, void * userData)`

Register a programmers callback function for error handling.

Parameters

<i>errorCallbackIn</i>	: Fuction 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.

J.10 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_CREATE_HANDLE](#) = 17,
 [OTF2_COLLECTIVE_OP_DESTROY_HANDLE](#) = 18,
 [OTF2_COLLECTIVE_OP_ALLOCATE](#) = 19,
 [OTF2_COLLECTIVE_OP_DEALLOCATE](#) = 20,
 [OTF2_COLLECTIVE_OP_CREATE_HANDLE_AND_ALLOCATE](#) = 21,
 [OTF2_COLLECTIVE_OP_DESTROY_HANDLE_AND_DEALLOCATE](#) =
 22 }
Types of collective operations.

J.10 OTF2_Events.h File Reference

- 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](#)
 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.

J.10.1 Detailed Description

Enums and types used in event records.

Source Template:

templates/OTF2_Events.tmpl.h

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.10.2 Enumeration Type Documentation

J.10.2.1 enum OTF2_CollectiveOp_enum

Types of collective operations.

Since

Version 1.0

Enumerator:

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.

J.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.

J.10 OTF2_Events.h File Reference

J.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.

J.10.2.4 enum OTF2_RmaAtomicType_enum

RMA Atomic Operation Type.

Since

Version 1.2

J.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).

J.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.

J.11 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)

J.11 OTF2_EvtReader.h File Reference

The following function can be used to get the position (number of the event in the stream) of last read event.

- [OTF2_ErrorCode OTF2_EvtReader_ReadEvents](#) ([OTF2_EvtReader](#) *reader, uint64_t recordsToRead, uint64_t *recordsRead)

After callback registration, the local events could be read with the following function. Readn reads recordsToRead records. The reader indicates that it reached the end of the trace by just reading less records than requested.

- [OTF2_ErrorCode OTF2_EvtReader_ReadEventsBackward](#) ([OTF2_EvtReader](#) *reader, uint64_t recordsToRead, uint64_t *recordsRead)

This functions reads recordsRead events backwards from the current position.

- [OTF2_ErrorCode OTF2_EvtReader_Seek](#) ([OTF2_EvtReader](#) *reader, uint64_t position)

Seek jumps to an event position.

- [OTF2_ErrorCode OTF2_EvtReader_SetCallbacks](#) ([OTF2_EvtReader](#) *reader, const [OTF2_EvtReaderCallbacks](#) *callbacks, void *userData)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

- [OTF2_ErrorCode OTF2_EvtReader_TimeStampRewrite](#) ([OTF2_EvtReader](#) *reader, [OTF2_TimeStamp](#) time)

The following function rewrites the timestamp from the event on the actual reading position if the buffer is in OTF2_BUFFER_MODIFY mode. It also modifies the timestamp for all other events in the same timestamp bundle. This function also has to keep track that not only the last timestamp, but all records equal to the last timestamp has to be modified. This is done by a position list, if there has no seek appeared before. In this case a position list can be easily generated because of that the reader has seen all related timestamps before. This not the case if there has a seek appeared before. In this case the related timestamp positions are generated by a linear search.

J.11.1 Detailed Description

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

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

APPENDIX J. FILE DOCUMENTATION

J.11.2 Function Documentation

J.11.2.1 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 eventes 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.

J.11.2.2 OTF2_ErrorCode OTF2_EvtReader.ApplyMappingTables (OTF2_EvtReader * *reader*, bool *action*)

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

This setting has no effect if the eventes 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.

J.11.2.3 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.
out	<i>location</i>	ID of the location.

J.11 OTF2_EvtReader.h File Reference

Returns

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

J.11.2.4 `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

	<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.

J.11.2.5 `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 were really read.

Returns

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

J.11.2.6 `OTF2_ErrorCode OTF2_EvtReader_ReadEventsBackward (OTF2_EvtReader * reader, uint64_t recordsToRead, uint64_t * recordsRead)`

This function reads *recordsRead* events backwards from the current position.

APPENDIX J. FILE DOCUMENTATION

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 were really read.

Returns

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

J.11.2.7 `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.

J.11.2.8 `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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

Returns

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

J.11.2.9 OTF2_ErrorCode OTF2_EvtReader_TimeStampRewrite (OTF2_EvtReader * reader, OTF2_TimeStamp time)

The following function rewrites the timestamp from the event on the actual reading position if the buffer is in OTF2_BUFFER_MODIFY mode. It also modifies the timestamp for all other events in the same timestamp bundle. This function also has to keep track that not only the last timestamp, but all records equal to the last timestamp has to be modified. This is done by a position list, if there has no seek appeared before. In this case a position list can be easily generated because of that the reader has seen all related timestamps before. This not the case if there has a seek appeared before. In this case the related timestamp positions are generated by a linear search.

Parameters

<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.

J.12 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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

- typedef [OTF2_CallbackCode](#)(* [OTF2_EvtReaderCallback_MpiIsend](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) time, uint64_t eventPosition, void *userData, [OTF2_AttributeList](#) *attributeList, uint32_t receiver, [OTF2_CommRef](#) communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)

Callback for the MpiIsend event record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_EvtReaderCallback_MpiIsendComplete](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) time, uint64_t eventPosition, void *userData, [OTF2_AttributeList](#) *attributeList, uint64_t requestID)

Callback for the MpiIsendComplete event record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_EvtReaderCallback_MpiRecv](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) time, uint64_t eventPosition, void *userData, [OTF2_AttributeList](#) *attributeList, uint32_t sender, [OTF2_CommRef](#) communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiRecv event record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_EvtReaderCallback_MpiRequestCancelled](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) time, uint64_t eventPosition, void *userData, [OTF2_AttributeList](#) *attributeList, uint64_t requestID)

Callback for the MpiRequestCancelled event record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_EvtReaderCallback_MpiRequestTest](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) time, uint64_t eventPosition, void *userData, [OTF2_AttributeList](#) *attributeList, uint64_t requestID)

Callback for the MpiRequestTest event record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_EvtReaderCallback_MpiSend](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) time, uint64_t eventPosition, void *userData, [OTF2_AttributeList](#) *attributeList, uint32_t receiver, [OTF2_CommRef](#) communicator, uint32_t msgTag, uint64_t msgLength)

Callback for the MpiSend event record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_EvtReaderCallback_OmpAcquireLock](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) time, uint64_t eventPosition, void *userData, [OTF2_AttributeList](#) *attributeList, uint32_t lockID, uint32_t acquisitionOrder)

Callback for the OmpAcquireLock event record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_EvtReaderCallback_OmpFork](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) time, uint64_t eventPosition, void *userData, [OTF2_AttributeList](#) *attributeList, uint32_t numberOfRequestedThreads)

Callback for the OmpFork event record.

- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_OmpJoin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)
Callback for the OmpJoin event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_OmpReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID, uint32_t acquisitionOrder)
Callback for the OmpReleaseLock event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_OmpTaskComplete)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)
Callback for the OmpTaskComplete event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)
Callback for the OmpTaskCreate event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)
Callback for the OmpTaskSwitch event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, int64_t value)
Callback for the ParameterInt event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, OTF2_StringRef string)
Callback for the ParameterString event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, uint64_t value)
Callback for the ParameterUnsignedInt event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)
Callback for the RmaAcquireLock event record.

J.12 OTF2_EvtReaderCallbacks.h File Reference

- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaAtomic)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)
Callback for the RmaAtomic event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)
Callback for the RmaCollectiveBegin event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)
Callback for the RmaCollectiveEnd event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaGet)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)
Callback for the RmaGet event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaGroupSync)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)
Callback for the RmaGroupSync event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaOpCompleteBlocking)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)
Callback for the RmaOpCompleteBlocking event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)
Callback for the RmaOpCompleteNonBlocking event record.
- typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaOpCompleteRemote)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)
Callback for the RmaOpCompleteRemote event record.

APPENDIX J. FILE DOCUMENTATION

- 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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

- 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_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 J. 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_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.

- 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)

J.12 OTF2_EvtReaderCallbacks.h File Reference

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.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetMpiSendCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_MpiSend](#) [mpiSendCallback](#))
Registers the callback for the MpiSend event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_OmpAcquireLock](#) [ompAcquireLockCallback](#))
Registers the callback for the OmpAcquireLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetOmpForkCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_OmpFork](#) [ompForkCallback](#))
Registers the callback for the OmpFork event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetOmpJoinCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_OmpJoin](#) [ompJoinCallback](#))
Registers the callback for the OmpJoin event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_OmpReleaseLock](#) [ompReleaseLockCallback](#))
Registers the callback for the OmpReleaseLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_OmpTaskComplete](#) [ompTaskCompleteCallback](#))
Registers the callback for the OmpTaskComplete event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_OmpTaskCreate](#) [ompTaskCreateCallback](#))
Registers the callback for the OmpTaskCreate event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetOmpTaskSwitchCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_OmpTaskSwitch](#) [ompTaskSwitchCallback](#))
Registers the callback for the OmpTaskSwitch event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetParameterIntCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ParameterInt](#) [parameterIntCallback](#))
Registers the callback for the ParameterInt event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetParameterStringCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ParameterString](#) [parameterStringCallback](#))
Registers the callback for the ParameterString event.

J.12 OTF2_EvtReaderCallbacks.h File Reference

- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetParameterUnsignedIntCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ParameterUnsignedInt](#) [parameterUnsignedIntCallback](#))
Registers the callback for the ParameterUnsignedInt event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaAcquireLock](#) [rmaAcquireLockCallback](#))
Registers the callback for the RmaAcquireLock event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaAtomicCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaAtomic](#) [rmaAtomicCallback](#))
Registers the callback for the RmaAtomic event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaCollectiveBegin](#) [rmaCollectiveBeginCallback](#))
Registers the callback for the RmaCollectiveBegin event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaCollectiveEnd](#) [rmaCollectiveEndCallback](#))
Registers the callback for the RmaCollectiveEnd event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaGetCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaGet](#) [rmaGetCallback](#))
Registers the callback for the RmaGet event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaGroupSync](#) [rmaGroupSyncCallback](#))
Registers the callback for the RmaGroupSync event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteBlockingCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaOpCompleteBlocking](#) [rmaOpCompleteBlockingCallback](#))
Registers the callback for the RmaOpCompleteBlocking event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking](#) [rmaOpCompleteNonBlockingCallback](#))
Registers the callback for the RmaOpCompleteNonBlocking event.
- [OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaOpCompleteRemoteCallback](#) ([OTF2_EvtReaderCallbacks *evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaOpCompleteRemote](#) [rmaOpCompleteRemoteCallback](#))
Registers the callback for the RmaOpCompleteRemote event.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaOpTestCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaOpTest](#) [rmaOpTestCallback](#))
Registers the callback for the RmaOpTest event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaPutCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaPut](#) [rmaPutCallback](#))
Registers the callback for the RmaPut event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaReleaseLockCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaReleaseLock](#) [rmaReleaseLockCallback](#))
Registers the callback for the RmaReleaseLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaRequestLockCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaRequestLock](#) [rmaRequestLockCallback](#))
Registers the callback for the RmaRequestLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaSyncCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaSync](#) [rmaSyncCallback](#))
Registers the callback for the RmaSync event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaTryLockCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaTryLock](#) [rmaTryLockCallback](#))
Registers the callback for the RmaTryLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaWaitChangeCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaWaitChange](#) [rmaWaitChangeCallback](#))
Registers the callback for the RmaWaitChange event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaWinCreateCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaWinCreate](#) [rmaWinCreateCallback](#))
Registers the callback for the RmaWinCreate event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetRmaWinDestroyCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_RmaWinDestroy](#) [rmaWinDestroyCallback](#))
Registers the callback for the RmaWinDestroy event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadAcquireLockCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadAcquireLock](#) [threadAcquireLockCallback](#))
Registers the callback for the ThreadAcquireLock event.

J.12 OTF2_EvtReaderCallbacks.h File Reference

- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadForkCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadFork](#) [threadForkCallback](#))
Registers the callback for the ThreadFork event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadJoinCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadJoin](#) [threadJoinCallback](#))
Registers the callback for the ThreadJoin event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadReleaseLockCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadReleaseLock](#) [threadReleaseLockCallback](#))
Registers the callback for the ThreadReleaseLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadTaskCompleteCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadTaskComplete](#) [threadTaskCompleteCallback](#))
Registers the callback for the ThreadTaskComplete event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadTaskCreateCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadTaskCreate](#) [threadTaskCreateCallback](#))
Registers the callback for the ThreadTaskCreate event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadTaskSwitchCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadTaskSwitch](#) [threadTaskSwitchCallback](#))
Registers the callback for the ThreadTaskSwitch event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadTeamBeginCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadTeamBegin](#) [threadTeamBeginCallback](#))
Registers the callback for the ThreadTeamBegin event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetThreadTeamEndCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_ThreadTeamEnd](#) [threadTeamEndCallback](#))
Registers the callback for the ThreadTeamEnd event.
- [OTF2_ErrorCode](#) [OTF2_EvtReaderCallbacks_SetUnknownCallback](#) ([OTF2_EvtReaderCallbacks](#) *[evtReaderCallbacks](#), [OTF2_EvtReaderCallback_Unknown](#) [unknownCallback](#))
Registers the callback for the Unknown event.

J.12.1 Detailed Description

This defines the callbacks for the event reader.

APPENDIX J. FILE DOCUMENTATION

Source Template:

templates/OTF2_EvtReaderCallbacks.tmpl.h

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.12.2 Typedef Documentation

J.12.2.1 `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.
<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](#).

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.2.2 `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](#).

J.12.2.3 `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.

APPENDIX J. 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>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](#).

J.12.2.4 `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.
<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](#).

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.2.5 `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.
<i>metricValues</i>	List of metric values.

Since

Version 1.0

Returns

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

J.12.2.6 `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.

APPENDIX J. FILE DOCUMENTATION

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 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](#).

J.12.2.7 `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: 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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

<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 <i>communicator</i> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeRe- ceived</i>	Size of the received message.

Since

Version 1.0

Returns

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

J.12.2.8 `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

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosi- tion</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 <i>communicator</i> .
<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*](#).

J.12.2.9 `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 <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 requested receive

Since

Version 1.0

Returns

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

J.12.2.10 `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,

J.12 OTF2_EvtReaderCallbacks.h File Reference

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 OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver 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
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

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

J.12.2.11 `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 .

APPENDIX J. FILE DOCUMENTATION

<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*](#).

J.12.2.12 `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.
<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>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 <i>OTF2_MAPPING_COMM</i> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.0

J.12 OTF2_EvtReaderCallbacks.h File Reference

Returns

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

J.12.2.13 `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 <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*](#).

J.12.2.14 `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.
-----------------	---

APPENDIX J. FILE DOCUMENTATION

<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](#).

J.12.2.15 `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).

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 <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

J.12 OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

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

J.12.2.16 `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 record.

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>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*](#).

APPENDIX J. FILE DOCUMENTATION

J.12.2.17 `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 OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_SetCallbacks .
<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](#).

J.12.2.18 `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.

J.12 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.

Since

Version 1.0

Returns

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

J.12.2.19 `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.
<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*](#).

J.12.2.20 `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 <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 completed task instance.

Since

Version 1.0

Returns

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

J.12.2.21 `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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

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 OTF2_Reader_RegisterEvtCallbacks or OTF2_EvtReader_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](#).

J.12.2.22 `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.

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 .

APPENDIX J. FILE DOCUMENTATION

<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*](#).

J.12.2.23 `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 <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 Parameter 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*](#).

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.2.24 `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](#).

J.12.2.25 `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.

APPENDIX J. 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>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](#).

J.12.2.26 `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.

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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

<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](#).

J.12.2.27 `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.
<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*](#).

J.12.2.28 `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 beginnig 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 <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.

Since

Version 1.2

Returns

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

J.12.2.29 `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

J.12 OTF2_EvtReaderCallbacks.h File Reference

<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](#).

J.12.2.30 `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.

An RmaGet 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 .

APPENDIX J. FILE DOCUMENTATION

<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](#).

J.12.2.31 `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 .
<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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

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

J.12.2.32 `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*](#).

APPENDIX J. FILE DOCUMENTATION

J.12.2.33 `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 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](#).

J.12.2.34 `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

J.12 OTF2_EvtReaderCallbacks.h File Reference

<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](#).

J.12.2.35 `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

<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](#).

J.12.2.36 `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.

Since

Version 1.2

Returns

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

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.2.37 `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 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 released, if multiple locks are defined on a window.

Since

Version 1.2

Returns

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

J.12.2.38 `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

APPENDIX J. 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>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](#).

J.12.2.39 `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.

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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

<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](#).

J.12.2.40 `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.
<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](#).

J.12.2.41 `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 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.

Since

Version 1.2

Returns

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

J.12.2.42 `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

J.12 OTF2_EvtReaderCallbacks.h File Reference

<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](#).

J.12.2.43 `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.
<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 destroyed. 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*](#).

J.12.2.44 `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 <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<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 1.2

Returns

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

J.12.2.45 `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.

J.12 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>numberOfRequestedThreads</i>	Requested size of the team.

Since

Version 1.2

Returns

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

J.12.2.46 `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadIdJoin)(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.

Since

Version 1.2

Returns

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

J.12.2.47 `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.
<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>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*](#).

J.12.2.48 `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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

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](#).

J.12.2.49 `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.

APPENDIX J. 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>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. (This is redundant, remove?)
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

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

J.12.2.50 `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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

<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*](#).

J.12.2.51 `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.

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 Comm definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.

Since

Version 1.2

Returns

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

APPENDIX J. FILE DOCUMENTATION

J.12.2.52 `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.

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](#).

J.12.2.53 `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_
Unknown)(OTF2_LocationRef location, OTF2_TimeStamp time,
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown event record.

Parameters

<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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

Returns

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

J.12.3 Function Documentation

J.12.3.1 void OTF2_EvtReaderCallbacks_Clear (OTF2_EvtReaderCallbacks *
 evtReaderCallbacks)

Clears a struct for the event callbacks.

Parameters

<i>evtReader- Callbacks</i>	Handle to a struct previously allocated with OTF2_- EvtReaderCallbacks_New .
---------------------------------	--

J.12.3.2 void OTF2_EvtReaderCallbacks_Delete (OTF2_EvtReaderCallbacks *
 evtReaderCallbacks)

Deallocates a struct for the event callbacks.

Parameters

<i>evtReader- Callbacks</i>	Handle to a struct previously allocated with OTF2_- EvtReaderCallbacks_New .
---------------------------------	--

J.12.3.3 OTF2_EvtReaderCallbacks* OTF2_EvtReaderCallbacks_New (void)

Allocates a new struct for the event callbacks.

Returns

A newly allocated struct of type [OTF2_EvtReaderCallbacks](#).

J.12.3.4 OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetBufferFlushCallback
(OTF2_EvtReaderCallbacks * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_BufferFlush *bufferFlushCallback*)

Registers the callback for the BufferFlush event.

Parameters

APPENDIX J. FILE DOCUMENTATION

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>bufferFlushCallback</i>	Function which should be called for all BufferFlush events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.5 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetEnterCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_Enter *enterCallback*)

Registers the callback for the Enter event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.6 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetLeaveCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_Leave *leaveCallback*)

Registers the callback for the Leave event.

Parameters

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

J.12 OTF2_EvtReaderCallbacks.h File Reference

<i>leaveCall- back</i>	Function which should be called for all Leave events.
----------------------------	---

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.7 **OTF2_ErrorCode** **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.8 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMetricCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_Metric *metricCallback*)

Registers the callback for the Metric event.

Parameters

<i>evtReader- Callbacks</i>	Struct for all callbacks.
<i>metricCall- back</i>	Function which should be called for all Metric events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.9 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiCollectiveBeginCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*, **OTF2_-**
EvtReaderCallback_MpiCollectiveBegin *mpiCollectiveBeginCallback*
)

Registers the callback for the `MpiCollectiveBegin` event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBegin-Callback</i>	Function which should be called for all <code>MpiCollectiveBegin</code> events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.10 **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.

J.12 OTF2_EvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.11 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiIrecvCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiIrecv *mpilrecvCallback*)

Registers the callback for the `MpiIrecv` event.

Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all <code>MpiIrecv</code> events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.12 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiIrecvRequestCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiIrecvRequest *mpilrecvRequestCallback*
)

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 <code>MpiIrecvRequest</code> events.

Returns

OTF2_SUCCESS if successful

APPENDIX J. FILE DOCUMENTATION

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.13 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiIsendCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiIsend *mpisendCallback*)

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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.14 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiIsendCompleteCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiIsendComplete
mpisendCompleteCallback)

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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.15 **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 MpiRecv events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.16 **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 MpiRequestCancelled events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.17 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiRequestTestCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiRequestTest *mpiRequestTestCallback*)

Registers the callback for the MpiRequestTest event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiRequestTest-Callback</i>	Function which should be called for all MpiRequestTest events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.18 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetMpiSendCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_MpiSend *mpiSendCallback*)

Registers the callback for the MpiSend event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiSend-Callback</i>	Function which should be called for all MpiSend events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.19 **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>ompAcquireLock-Callback</i>	Function which should be called for all OmpAcquireLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.20 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetOmpForkCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>ompFork-Callback</i>	Function which should be called for all OmpFork events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.21 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetOmpJoinCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpJoin *ompJoinCallback*)

Registers the callback for the OmpJoin event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>ompJoin-Callback</i>	Function which should be called for all OmpJoin events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.22 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.23 **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>omp-TaskCompleteCallback</i>	Function which should be called for all OmpTaskComplete events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.24 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpTaskCreate *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>omp-TaskCreateCallback</i>	Function which should be called for all OmpTaskCreate events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.25 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetOmpTaskSwitchCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_OmpTaskSwitch *ompTaskSwitchCallback*)

Registers the callback for the OmpTaskSwitch event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>omp-TaskSwitch-Callback</i>	Function which should be called for all OmpTaskSwitch events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.26 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.27 **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.
<i>parameter-StringCall-back</i>	Function which should be called for all ParameterString events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.28 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.-SetParameterUnsignedIntCallback** (**OTF2_EvtReaderCallbacks**
* *evtReaderCallbacks*, **OTF2_EvtReaderCallback_-ParameterUnsignedInt** *parameterUnsignedIntCallback*
)

Registers the callback for the ParameterUnsignedInt event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>parameterUnsignedInt-Callback</i>	Function which should be called for all ParameterUnsignedInt events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.29 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.30 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.31 **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 RmaCollectiveBegin events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.32 **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 RmaCollectiveEnd events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.33 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaGetCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaGet *rmaGetCallback*)

Registers the callback for the RmaGet event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaGet-Callback</i>	Function which should be called for all RmaGet events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.34 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaGroupSyncCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaGroupSync *rmaGroupSyncCallback*)

Registers the callback for the RmaGroupSync event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaGroup-SyncCallback</i>	Function which should be called for all RmaGroupSync events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.35 OTF2_ErrorCode OTF2_EvtReaderCallbacks_-
SetRmaOpCompleteBlockingCallback (OTF2_EvtReaderCallbacks
*** *evtReaderCallbacks*, OTF2_EvtReaderCallback_-**
RmaOpCompleteBlocking *rmaOpCompleteBlockingCallback*
)

Registers the callback for the RmaOpCompleteBlocking event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaOp-Complete-Blocking-Callback</i>	Function which should be called for all RmaOpCompleteBlocking events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.36 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.

Returns

OTF2_SUCCESS if successful

APPENDIX J. FILE DOCUMENTATION

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.37 **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- emoteCall- back</i>	Function which should be called for all RmaOpCompleteRemote events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.38 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks_**
SetRmaOpTestCallback
(**OTF2_EvtReaderCallbacks** * ***evtReaderCallbacks***,
OTF2_EvtReaderCallback_RmaOpTest ***rmaOpTestCallback***)

Registers the callback for the RmaOpTest event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaOpTest- Callback</i>	Function which should be called for all RmaOpTest events.

Returns

OTF2_SUCCESS if successful

J.12 OTF2_EvtReaderCallbacks.h File Reference

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.39 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaPutCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaPut *rmaPutCallback*)

Registers the callback for the RmaPut event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaPut-Callback</i>	Function which should be called for all RmaPut events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.40 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaReleaseLockCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_RmaReleaseLock *rmaReleaseLockCallback*
)

Registers the callback for the RmaReleaseLock event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaReleaseLock-Callback</i>	Function which should be called for all RmaReleaseLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.41 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaRequestLockCallback**
 (**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
 OTF2_EvtReaderCallback_RmaRequestLock *rmaRequestLockCallback*
)

Registers the callback for the RmaRequestLock event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaRequestLock-Callback</i>	Function which should be called for all RmaRequestLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.42 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetRmaSyncCallback**
 (**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
 OTF2_EvtReaderCallback_RmaSync *rmaSyncCallback*)

Registers the callback for the RmaSync event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaSync-Callback</i>	Function which should be called for all RmaSync events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.43 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.44 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.45 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.46 **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-stroyCall-back</i>	Function which should be called for all RmaWinDestroy events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.47 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadAcquireLockCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadAcquireLock
threadAcquireLockCallback)

Registers the callback for the ThreadAcquireLock event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>threadAc-quireLock-Callback</i>	Function which should be called for all ThreadAcquireLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.12.3.48 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadForkCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadFork *threadForkCallback*)

Registers the callback for the ThreadFork event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>threadFork-Callback</i>	Function which should be called for all ThreadFork events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.49 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadJoinCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadJoin *threadJoinCallback*)

Registers the callback for the ThreadJoin event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>threadJoin-Callback</i>	Function which should be called for all ThreadJoin events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.50 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadReleaseLockCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadReleaseLock
threadReleaseLockCallback)

Registers the callback for the ThreadReleaseLock event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-Release-LockCallback</i>	Function which should be called for all ThreadReleaseLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12 OTF2_EvtReaderCallbacks.h File Reference

J.12.3.51 **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- TaskCom- pleteCall- back</i>	Function which should be called for all ThreadTaskComplete events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`
argument

J.12.3.52 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`
argument

APPENDIX J. FILE DOCUMENTATION

J.12.3.53 **OTF2_ErrorCode** **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks*
argument

J.12.3.54 **OTF2_ErrorCode** **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks*
argument

J.13 OTF2_EvtWriter.h File Reference

J.12.3.55 **OTF2_ErrorCode** **OTF2_EvtReaderCallbacks.SetThreadTeamEndCallback**
(**OTF2_EvtReaderCallbacks** * *evtReaderCallbacks*,
OTF2_EvtReaderCallback_ThreadTeamEnd *threadTeamEndCallback*)

Registers the callback for the ThreadTeamEnd event.

Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>threadTeamEndCallback</i>	Function which should be called for all ThreadTeamEnd events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.12.3.56 **OTF2_ErrorCode** **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>unknownCallback</i>	Function which should be called for all unknown events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.13 OTF2_EvtWriter.h File Reference

This lowest user-visible layer provides write routines to write event data of a single location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_AttributeList.h>
```

Typedefs

- typedef struct OTF2_EvtWriter_struct [OTF2_EvtWriter](#)
Keeps all necessary information about the event writer. See OTF2_EvtWriter_struct for detailed information.

Functions

- [OTF2_ErrorCode OTF2_EvtWriter_BufferFlush](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_TimeStamp](#) stopTime)
Records an BufferFlush event.
- [OTF2_ErrorCode OTF2_EvtWriter_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)

J.13 OTF2_EvtWriter.h File Reference

Records an MeasurementOnOff event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_Metric](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_MetricRef](#) metric, [uint8_t](#) numberOfMetrics, [const](#) [OTF2_Type](#) *typeIDs, [const](#) [OTF2_MetricValue](#) *metricValues)

Records an Metric event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiCollectiveBegin](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time)

Records an MpiCollectiveBegin event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiCollectiveEnd](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_CollectiveOp](#) collectiveOp, [OTF2_CommRef](#) communicator, [uint32_t](#) root, [uint64_t](#) sizeSent, [uint64_t](#) sizeReceived)

Records an MpiCollectiveEnd event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiIrecv](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [uint32_t](#) sender, [OTF2_CommRef](#) communicator, [uint32_t](#) msgTag, [uint64_t](#) msgLength, [uint64_t](#) requestID)

Records an MpiIrecv event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiIrecvRequest](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [uint64_t](#) requestID)

Records an MpiIrecvRequest event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiIsend](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [uint32_t](#) receiver, [OTF2_CommRef](#) communicator, [uint32_t](#) msgTag, [uint64_t](#) msgLength, [uint64_t](#) requestID)

Records an MpiIsend event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiIsendComplete](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [uint64_t](#) requestID)

Records an MpiIsendComplete event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiRecv](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [uint32_t](#) sender, [OTF2_CommRef](#) communicator, [uint32_t](#) msgTag, [uint64_t](#) msgLength)

Records an MpiRecv event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiRequestCancelled](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [uint64_t](#) requestID)

Records an MpiRequestCancelled event.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiRequestTest](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, uint64_t requestID)

Records an MpiRequestTest event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_MpiSend](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, uint32_t receiver, [OTF2_CommRef](#) communicator, uint32_t msgTag, uint64_t msgLength)

Records an MpiSend event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_OmpAcquireLock](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, uint32_t lockID, uint32_t acquisitionOrder)

Records an OmpAcquireLock event.

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_OmpFork](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, uint32_t 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.

J.13 OTF2_EvtWriter.h File Reference

- [OTF2_ErrorCode OTF2_EvtWriter_ParameterString](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_ParameterRef](#) parameter, [OTF2_StringRef](#) string)
Records an ParameterString event.
- [OTF2_ErrorCode OTF2_EvtWriter_ParameterUnsignedInt](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_ParameterRef](#) parameter, [uint64_t](#) value)
Records an ParameterUnsignedInt event.
- [OTF2_ErrorCode OTF2_EvtWriter_Rewind](#) ([OTF2_EvtWriter](#) *writer, [uint32_t](#) rewindId)
Please give me a documantation.
- [OTF2_ErrorCode OTF2_EvtWriter_RmaAcquireLock](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint32_t](#) remote, [uint64_t](#) lockId, [OTF2_LockType](#) lockType)
Records an RmaAcquireLock event.
- [OTF2_ErrorCode OTF2_EvtWriter_RmaAtomic](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint32_t](#) remote, [OTF2_RmaAtomicType](#) type, [uint64_t](#) bytesSent, [uint64_t](#) bytesReceived, [uint64_t](#) matchingId)
Records an RmaAtomic event.
- [OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveBegin](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time)
Records an RmaCollectiveBegin event.
- [OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveEnd](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_CollectiveOp](#) collectiveOp, [OTF2_RmaSyncLevel](#) syncLevel, [OTF2_RmaWinRef](#) win, [uint32_t](#) root, [uint64_t](#) bytesSent, [uint64_t](#) bytesReceived)
Records an RmaCollectiveEnd event.
- [OTF2_ErrorCode OTF2_EvtWriter_RmaGet](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint32_t](#) remote, [uint64_t](#) bytes, [uint64_t](#) matchingId)
Records an RmaGet event.
- [OTF2_ErrorCode OTF2_EvtWriter_RmaGroupSync](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaSyncLevel](#) syncLevel, [OTF2_RmaWinRef](#) win, [OTF2_GroupRef](#) group)
Records an RmaGroupSync event.
- [OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteBlocking](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint64_t](#) matchingId)
Records an RmaOpCompleteBlocking event.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaOpCompleteNonBlocking](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint64_t](#) matchingId)
Records an RmaOpCompleteNonBlocking event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaOpCompleteRemote](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint64_t](#) matchingId)
Records an RmaOpCompleteRemote event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaOpTest](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint64_t](#) matchingId)
Records an RmaOpTest event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaPut](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint32_t](#) remote, [uint64_t](#) bytes, [uint64_t](#) matchingId)
Records an RmaPut event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaReleaseLock](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint32_t](#) remote, [uint64_t](#) lockId)
Records an RmaReleaseLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaRequestLock](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint32_t](#) remote, [uint64_t](#) lockId, [OTF2_LockType](#) lockType)
Records an RmaRequestLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaSync](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint32_t](#) remote, [OTF2_RmaSyncType](#) syncType)
Records an RmaSync event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaTryLock](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win, [uint32_t](#) remote, [uint64_t](#) lockId, [OTF2_LockType](#) lockType)
Records an RmaTryLock event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaWaitChange](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win)
Records an RmaWaitChange event.
- [OTF2_ErrorCode](#) [OTF2_EvtWriter_RmaWinCreate](#) ([OTF2_EvtWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) time, [OTF2_RmaWinRef](#) win)
Records an RmaWinCreate event.

J.13 OTF2_EvtWriter.h File Reference

- [OTF2_ErrorCode OTF2_EvtWriter_RmaWinDestroy](#) (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win)
Records an RmaWinDestroy event.
- [OTF2_ErrorCode OTF2_EvtWriter_SetLocationID](#) (OTF2_EvtWriter *writer, OTF2_LocationRef location)
The location ID is not always known on measurment start, and only needed on the first buffer flush to generate the file name. This function enables setting of the location ID after generating the buffer object.
- [OTF2_ErrorCode OTF2_EvtWriter_SetUserData](#) (OTF2_EvtWriter *writer, void *userData)
Function to set user defined data to a writer object.
- [OTF2_ErrorCode OTF2_EvtWriter_StoreRewindPoint](#) (OTF2_EvtWriter *writer, uint32_t rewindId)
Please give me a documantation.
- [OTF2_ErrorCode OTF2_EvtWriter_ThreadAcquireLock](#) (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)
Records an ThreadAcquireLock event.
- [OTF2_ErrorCode OTF2_EvtWriter_ThreadFork](#) (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)
Records an ThreadFork event.
- [OTF2_ErrorCode OTF2_EvtWriter_ThreadJoin](#) (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_Paradigm model)
Records an ThreadJoin event.
- [OTF2_ErrorCode OTF2_EvtWriter_ThreadReleaseLock](#) (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)
Records an ThreadReleaseLock event.
- [OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskComplete](#) (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)
Records an ThreadTaskComplete event.
- [OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskCreate](#) (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)
Records an ThreadTaskCreate event.

APPENDIX J. FILE DOCUMENTATION

- [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.

J.13.1 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

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.13.2 Function Documentation

J.13.2.1 [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.
---------------	----------------

J.13 OTF2_EvtWriter.h File Reference

<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.

J.13.2.2 `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.

J.13.2.3 `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.

APPENDIX J. 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 1.0

Returns

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

J.13.2.4 `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.

J.13.2.5 `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.

J.13 OTF2_EvtWriter.h File Reference

J.13.2.6 `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.

J.13.2.7 `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 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.

J.13.2.8 `OTF2_ErrorCode OTF2_EvtWriter_MeasurementOnOff (OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, OTF2_MeasurementMode measurementMode)`

Records an MeasurementOnOff event.

APPENDIX J. FILE DOCUMENTATION

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.

J.13.2.9 `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.
<i>metricValues</i>	List of metric values.

J.13 OTF2_EvtWriter.h File Reference

Since

Version 1.0

Returns

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

J.13.2.10 **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.

J.13.2.11 **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.
---------------	----------------

APPENDIX J. FILE DOCUMENTATION

<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 <i>communicator</i> .
<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.

J.13.2.12 `OTF2_ErrorCode OTF2_EvtWriter.MpiIrecv (OTF2_EvtWriter *
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,
uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t
msgLength, uint64_t requestID)`

Records an MpiIrecv event.

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

Parameters

<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
<i>requestID</i>	ID of the related request

J.13 OTF2_EvtWriter.h File Reference

Since

Version 1.0

Returns

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

J.13.2.13 `OTF2_ErrorCode OTF2_EvtWriter_MpilrecvRequest (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.

J.13.2.14 `OTF2_ErrorCode OTF2_EvtWriter_Mpilsend (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

APPENDIX J. 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 <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
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

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

J.13.2.15 `OTF2_ErrorCode OTF2_EvtWriter_MpilsendComplete (OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, uint64_t requestID)`

Records an MpilsendComplete 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.

J.13 OTF2_EvtWriter.h File Reference

J.13.2.16 `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>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.

J.13.2.17 `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

Since

Version 1.0

Returns

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

J.13.2.18 **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.

J.13.2.19 **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

J.13 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 <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.

J.13.2.20 `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 record.

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.

Since

Version 1.0

Deprecated

In version 1.2

Returns

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

J.13.2.21 **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.

J.13.2.22 **OTF2_ErrorCode** OTF2_EvtWriter_OmpJoin (OTF2_EvtWriter * *writer*,
OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*)

Records an OmpJoin event.

J.13 OTF2_EvtWriter.h File Reference

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.

J.13.2.23 **OTF2_ErrorCode** OTF2_EvtWriter.OmpReleaseLock (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, uint32_t *lockID*, uint32_t *acquisitionOrder*)

Records an OmpReleaseLock event.

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

This event record is superseded by the [ThreadReleaseLock](#) event record and should not be used when the [ThreadReleaseLock](#) event record is in use.

Parameters

<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.

Since

Version 1.0

Deprecated

In version 1.2

Returns

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

J.13.2.24 **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.

J.13 OTF2_EvtWriter.h File Reference

J.13.2.25 **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.

J.13.2.26 **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.
---------------	----------------

APPENDIX J. FILE DOCUMENTATION

<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.

J.13.2.27 `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 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.

J.13 OTF2_EvtWriter.h File Reference

J.13.2.28 `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.

J.13.2.29 `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.

APPENDIX J. 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 1.0

Returns

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

J.13.2.30 `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.

J.13.2.31 `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.
---------------	----------------

J.13 OTF2_EvtWriter.h File Reference

<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.

J.13.2.32 `OTF2_StatusCode 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

Returns

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

J.13.2.33 **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.

J.13.2.34 **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.

J.13 OTF2_EvtWriter.h File Reference

<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.

J.13.2.35 `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 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 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.

APPENDIX J. FILE DOCUMENTATION

J.13.2.36 **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.

J.13.2.37 **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.

J.13 OTF2_EvtWriter.h File Reference

<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.

J.13.2.38 `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.

APPENDIX J. FILE DOCUMENTATION

J.13.2.39 **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.

J.13.2.40 **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.

J.13 OTF2_EvtWriter.h File Reference

Since

Version 1.2

Returns

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

J.13.2.41 `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.

J.13.2.42 `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.

APPENDIX J. FILE DOCUMENTATION

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.

J.13.2.43 `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

J.13 OTF2_EvtWriter.h File Reference

Returns

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

J.13.2.44 `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 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_SUCCESS*](#) if successful, an error code if an error occurs.

J.13.2.45 `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

APPENDIX J. FILE DOCUMENTATION

<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.

J.13.2.46 **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.

J.13 OTF2_EvtWriter.h File Reference

J.13.2.47 `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.

J.13.2.48 `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 destructured. 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

APPENDIX J. FILE DOCUMENTATION

Returns

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

J.13.2.49 `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.

J.13.2.50 `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 OTF2_EvtWriter_GetUserData .

Returns

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

J.13.2.51 `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.

J.13 OTF2_EvtWriter.h File Reference

Since

Version 1.1

Returns

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

J.13.2.52 **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>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.

J.13.2.53 **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.
---------------	----------------

APPENDIX J. FILE DOCUMENTATION

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>num-berOfRe-quest-edThreads</i>	Requested size of the team.

Since

Version 1.2

Returns

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

J.13.2.54 **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.

Since

Version 1.2

Returns

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

J.13.2.55 **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.

J.13 OTF2_EvtWriter.h File Reference

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.
<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.

J.13.2.56 `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 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.

J.13.2.57 **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. (This is redundant, remove?)
<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.

J.13.2.58 **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.

J.13 OTF2_EvtWriter.h File Reference

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.
<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.

J.13.2.59 **OTF2_ErrorCode** OTF2_EvtWriter_ThreadTeamBegin (OTF2_EvtWriter * *writer*, OTF2_AttributeList * *attributeList*, OTF2_TimeStamp *time*, OTF2_CommRef *threadTeam*)

Records an ThreadTeamBegin event.

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.

J.13.2.60 `OTF2_ErrorCode OTF2_EvtWriter_ThreadTeamEnd (OTF2_EvtWriter *
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,
OTF2_CommRef threadTeam)`

Records an ThreadTeamEnd event.

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.

J.14 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)`

J.14 OTF2_GeneralDefinitions.h File Reference

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_COMM](#) (([OTF2_CommRef](#))OTF2_UNDEFINED_UINT32)

The invalid value for a reference to a [Comm](#) definition.

- #define [OTF2_UNDEFINED_GROUP](#) (([OTF2_GroupRef](#))OTF2_UNDEFINED_UINT32)

The invalid value for a reference to a [Group](#) definition.

- #define [OTF2_UNDEFINED_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_TYPE OTF2_UNDEFINED_UINT8`

OTF2 library version.

- `#define OTF2_VERSION_MAJOR 1`
- `#define OTF2_VERSION_MINOR 2`
- `#define OTF2_VERSION_BUGFIX 1`
- `#define OTF2_VERSION_SUFFIX ""`
- `#define OTF2_VERSION "1.2.1"`

Standard undefined values for basic data types.

- `#define OTF2_UNDEFINED_UINT8 ((uint8_t)(~((uint8_t)0u)))`
- `#define OTF2_UNDEFINED_UINT16 ((uint16_t)(~((uint16_t)0u)))`
- `#define OTF2_UNDEFINED_UINT32 ((uint32_t)(~((uint32_t)0u)))`
- `#define OTF2_UNDEFINED_UINT64 ((uint64_t)(~((uint64_t)0u)))`

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_CommRef`
Type used to indicate a reference to a [Comm](#) definition.
- `typedef uint8_t OTF2_Compression`
Defines which compression is used. Please see [OTF2_Compression_enum](#) for a detailed description.
- `typedef struct OTF2_DefReader_struct OTF2_DefReader`
OTF2 local definition reader handle.
- `typedef struct OTF2_EvtReader_struct OTF2_EvtReader`
OTF2 local event reader handle.
- `typedef uint8_t OTF2_FileMode`
Defines how to interact with files. Please see [OTF2_FileMode_enum](#) for a detailed description.

J.14 OTF2_GeneralDefinitions.h File Reference

- typedef uint8_t [OTF2_FileSubstrate](#)
Defines which file substrate is used. Please see [OTF2_FileSubstrate_enum](#) for a detailed description.
- typedef uint8_t [OTF2_FileType](#)
Defines which file type is used. Please see [OTF2_FileType_enum](#) for a detailed description.
- 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](#) }

Return value to indicate that the record reading should be interrupted.

- enum [OTF2_Compression_enum](#) {
 [OTF2_COMPRESSION_UNDEFINED](#) = 0,
 [OTF2_COMPRESSION_NONE](#) = 1,
 [OTF2_COMPRESSION_ZLIB](#) = 2 }

Defines which compression is used.

- enum [OTF2_FileMode_enum](#) {
 [OTF2_FILEMODE_WRITE](#) = 0,
 [OTF2_FILEMODE_READ](#) = 1,
 [OTF2_FILEMODE_MODIFY](#) = 2 }

Defines how to interact with files.

- enum [OTF2_FileSubstrate_enum](#) {
 [OTF2_SUBSTRATE_UNDEFINED](#) = 0,
 [OTF2_SUBSTRATE_POSIX](#) = 1,
 [OTF2_SUBSTRATE_SION](#) = 2,
 [OTF2_SUBSTRATE_NONE](#) = 3 }

Defines which file substrate is used. Please note: At the moment only the posix and none interfaces are implemented.

- 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 }
 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 }
 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 basic data types.

J.14.1 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

Maintainer:

Michael Wagner <michael.wagner@zih.tu-dresden.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.14 OTF2_GeneralDefinitions.h File Reference

J.14.2 Define Documentation

J.14.2.1 `#define OTF2_UNDEFINED_TYPE OTF2_UNDEFINED_UINT8`

Undefined value for enums

J.14.3 Enumeration Type Documentation

J.14.3.1 `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.

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*.

J.14.3.2 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.

J.14.3.3 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.

J.14.3.4 enum OTF2_FileSubstrate_enum

Defines which file substrate is used. Please note: At the moment only the posix and none interfaces are implemented.

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.

J.14.3.5 enum OTF2_FileType_enum

Defines which file type is used.

J.14 OTF2_GeneralDefinitions.h File Reference

Enumerator:

- OTF2_FILETYPE_ANCHOR*** Represents the type for the anchor file (.otf2). Does has a undefined location ID.
- OTF2_FILETYPE_GLOBAL_DEFS*** Represents the type for the global definition file (.def). Does has a undefined location ID.
- 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).

J.14.3.6 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.

J.14.3.7 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 MPI communicator identifiers.

OTF2_MAPPING_PARAMETER Mapping of parameter identifiers.

OTF2_MAPPING_RMA_WIN Mapping of RMA window identifiers.

OTF2_MAPPING_MAX Max entry.

J.14.3.8 enum OTF2_Paradigm_enum

List of known paradigms.

Since

Version 1.1

Enumerator:

OTF2_PARADIGM_UNKNOWN An unknown paradigm.

OTF2_PARADIGM_USER Regions generated through user instrumentation.

OTF2_PARADIGM_COMPILER Regions generated through compiler instrumentation.

OTF2_PARADIGM_OPENMP Regions referring to OpenMP directives and API functions.

OTF2_PARADIGM_MPI Regions referring to MPI functions.

OTF2_PARADIGM_CUDA Regions referring to CUDA API functions.

OTF2_PARADIGM_MEASUREMENT_SYSTEM Regions used by the measurement software.

Since

Version 1.2.

J.14.3.9 enum OTF2_ThumbnailType_enum

Type of definitions used as metric in an thumbnail.

Since

Version 1.2

J.15 OTF2_GlobalDefReader.h File Reference

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](#).

J.14.3.10 enum OTF2_Type_enum

OTF2 basic data types.

Since

Version 1.0

Enumerator:

OTF2_TYPE_NONE Undefined type.

OTF2_TYPE_UINT8 Unsigned 8-bit integer.

OTF2_TYPE_UINT16 Unsigned 16-bit integer.

OTF2_TYPE_UINT32 Unsigned 32-bit integer.

OTF2_TYPE_UINT64 Unsigned 64-bit integer.

OTF2_TYPE_INT8 Signed 8-bit integer.

OTF2_TYPE_INT16 Signed 16-bit integer.

OTF2_TYPE_INT32 Signed 32-bit integer.

OTF2_TYPE_INT64 Signed 64-bit integer.

OTF2_TYPE_FLOAT 32-bit floating point value.

OTF2_TYPE_DOUBLE 64-bit floating point value.

J.15 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>
```

APPENDIX J. FILE DOCUMENTATION

Functions

- [OTF2_ErrorCode OTF2_GlobalDefReader_ReadDefinitions](#) ([OTF2_GlobalDefReader](#) *reader, uint64_t recordsToRead, uint64_t *recordsRead)
Reads the given number of records from the global definition reader.
- [OTF2_ErrorCode OTF2_GlobalDefReader_SetCallbacks](#) ([OTF2_GlobalDefReader](#) *reader, const [OTF2_GlobalDefReaderCallbacks](#) *callbacks, void *userData)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

J.15.1 Detailed Description

This is the definition reader.

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.15.2 Function Documentation

J.15.2.1 [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.

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

Returns

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

J.15.2.2 `OTF2_ErrorCode OTF2_GlobalDefReader_SetCallbacks (OTF2_GlobalDefReader * reader, const OTF2_GlobalDefReaderCallbacks * callbacks, void * userData)`

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

<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.

J.16 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_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.
- 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)
Function pointer definition for the callback which is triggered by a [Location-Group](#) 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)

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

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](#) 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)

APPENDIX J. FILE DOCUMENTATION

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.

- 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)

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

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

- [OTF2_ErrorCode](#) [OTF2_GlobalDefReaderCallbacks_SetClockPropertiesCallback](#) ([OTF2_GlobalDefReaderCallbacks](#) *globalDefReaderCallbacks, [OTF2_GlobalDefReaderCallback_ClockProperties](#) clockPropertiesCallback)

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

- [OTF2_ErrorCode](#) [OTF2_GlobalDefReaderCallbacks_SetCommCallback](#) ([OTF2_GlobalDefReaderCallbacks](#) *globalDefReaderCallbacks, [OTF2_GlobalDefReaderCallback_Comm](#) commCallback)

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

- [OTF2_ErrorCode](#) [OTF2_GlobalDefReaderCallbacks_SetGroupCallback](#) ([OTF2_GlobalDefReaderCallbacks](#) *globalDefReaderCallbacks, [OTF2_GlobalDefReaderCallback_Group](#) groupCallback)

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

- [OTF2_ErrorCode](#) [OTF2_GlobalDefReaderCallbacks_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_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.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_GlobalDefReaderCallbacks_SetRegionCallback](#) ([OTF2_GlobalDefReaderCallbacks](#) *[globalDefReaderCallbacks](#), [OTF2_GlobalDefReaderCallback_Region](#) [regionCallback](#))
Registers the callback for the [Region](#) definition.
- [OTF2_ErrorCode](#) [OTF2_GlobalDefReaderCallbacks_SetRmaWinCallback](#) ([OTF2_GlobalDefReaderCallbacks](#) *[globalDefReaderCallbacks](#), [OTF2_GlobalDefReaderCallback_RmaWin](#) [rmaWinCallback](#))
Registers the callback for the [RmaWin](#) definition.
- [OTF2_ErrorCode](#) [OTF2_GlobalDefReaderCallbacks_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.

J.16.1 Detailed Description

This defines the callbacks for the global definition reader.

Source Template:

templates/OTF2_GlobalDefReaderCallbacks.tmpl.h

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.16.2 Typedef Documentation

J.16.2.1 `typedef OTF2_CallbackCode(* OTF2_GlobalDefReaderCallback_ - Attribute)(void *userData, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_Type type)`

Function pointer definition for the callback which is triggered by a [Attribute](#) definition record.

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>type</i>	Type of the attribute value.

Since

Version 1.0

Returns

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

J.16.2.2 `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.

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>	References a Callpath definition.
<i>region</i>	References a Region definition.

Since

Version 1.0

Returns

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

J.16.2.3 `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.

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*](#).

J.16.2.4 `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*).

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

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](#).

J.16.2.5 `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.

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 <code>OTF2_GROUP_TYPE_MPI_GROUP</code> or <code>OTF2_GROUP_TYPE_MPI_COMM_SELF</code> . 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*.

J.16.2.6 `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.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_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*.

J.16.2.7 `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.

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

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](#).

J.16.2.8 `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.

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>sys- temTreePar- ent</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*](#).

J.16.2.9 `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

Returns

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

J.16.2.10 `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.

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_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](#).

J.16.2.11 `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

<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](#).

J.16.2.12 `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.
<i>valueType</i>	Type of the value: int64_t, uint64_t, or double.
<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.

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

Since

Version 1.0

Returns

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

J.16.2.13 `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.

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*](#).

J.16.2.14 `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.

Parameters

APPENDIX J. 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 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](#).

J.16.2.15 `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.

J.16 OTF2_GlobalDefReaderCallbacks.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](#).

J.16.2.16 `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.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_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](#).

J.16.2.17 `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.

Parameters

APPENDIX J. 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 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](#).

J.16.2.18 `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.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>systemTreeNode</i>	Parent SystemTreeNode definition to which this one is a supplementary definition. References a SystemTreeNode definition.

Since

Version 1.2

Returns

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

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

J.16.2.19 `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.

Parameters

<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
<i>systemTreeNode</i>	Parent SystemTreeNode definition to which this one is a supplementary definition. References a SystemTreeNode definition.
<i>name</i>	Name of the property. References a String definition.
<i>value</i>	Property value. References a String definition.

Since

Version 1.2

Returns

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

J.16.2.20 `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 OTF2_Reader_RegisterGlobalDefCallbacks or OTF2_GlobalDefReader_SetCallbacks .
-----------------	---

Returns

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

J.16.3 Function Documentation

J.16.3.1 `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 OTF2_- GlobalDefReaderCallbacks_New .
---	--

J.16.3.2 `void OTF2_GlobalDefReaderCallbacks_Delete (OTF2_-
GlobalDefReaderCallbacks * globalDefReaderCallbacks
)`

Deallocates a struct for the global definition callbacks.

Parameters

<i>globalDef- Reader- Callbacks</i>	Handle to a struct previously allocated with OTF2_- GlobalDefReaderCallbacks_New .
---	--

J.16.3.3 `OTF2_GlobalDefReaderCallbacks* OTF2_GlobalDefReaderCallbacks_New (
void)`

Allocates a new struct for the global definition callbacks.

Returns

A newly allocated struct of type [OTF2_GlobalDefReaderCallbacks](#).

J.16.3.4 `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetAttributeCallback
(OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks,
OTF2_GlobalDefReaderCallback_Attribute attributeCallback)`

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

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

Parameters

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.5 `OTF2_StatusCode 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 Callpath definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.6 `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetCallsiteCallback (OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Callsite callsiteCallback)`

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

Parameters

APPENDIX J. FILE DOCUMENTATION

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.7 `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetClockPropertiesCallback (OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_ ClockProperties clockPropertiesCallback)`

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

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>clockPropertiesCallback</i>	Function which should be called for all ClockProperties definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.8 `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetCommCallback (OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_ Comm commCallback)`

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

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>commCallback</i>	Function which should be called for all Comm definitions.

Returns

[**OTF2_SUCCESS**](#) if successful

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

J.16.3.9 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 Group definitions.

Returns

[**OTF2_SUCCESS**](#) if successful

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

J.16.3.10 OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetLocationCallback
(OTF2_GlobalDefReaderCallbacks * *globalDefReaderCallbacks*,
OTF2_GlobalDefReaderCallback_Location *locationCallback*)

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

Parameters

APPENDIX J. FILE DOCUMENTATION

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>location-Callback</i>	Function which should be called for all Location definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.11 **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 LocationGroup definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.12 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_SetMetricClassCallback** (**OTF2_GlobalDefReaderCallbacks** * *globalDefReaderCallbacks*, **OTF2_GlobalDefReaderCallback_MetricClass** *metricClassCallback*)

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

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

Parameters

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

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.13 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-Class-Recorder-Callback</i>	Function which should be called for all MetricClassRecorder definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

APPENDIX J. FILE DOCUMENTATION

J.16.3.14 **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 MetricInstance definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.15 **OTF2_ErrorCode** **OTF2_GlobalDefReaderCallbacks_-SetMetricMemberCallback** (**OTF2_GlobalDefReaderCallbacks *** *globalDefReaderCallbacks*, **OTF2_GlobalDefReaderCallback_-MetricMember** *metricMemberCallback*)

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

Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metricMemberCallback</i>	Function which should be called for all MetricMember definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

J.16.3.16 `OTF2_ErrorCode 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 Parameter definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.17 `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 Region definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

APPENDIX J. FILE DOCUMENTATION

J.16.3.18 **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 RmaWin definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.19 **OTF2_ErrorCode** **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 String definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16 OTF2_GlobalDefReaderCallbacks.h File Reference

J.16.3.20 **OTF2_ErrorCode** **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 SystemTreeNode definitions.

Returns

[OTF2_SUCCESS](#) if successful

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

J.16.3.21 **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>	Structs for all callbacks.
<i>systemTreeNodeDomainCallback</i>	Function which should be called for all SystemTreeNodeDomain definitions.

APPENDIX J. FILE DOCUMENTATION

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.16.3.22 **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 SystemTreeNodeProperty definitions.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.16.3.23 **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.

J.17 OTF2_GlobalDefWriter.h File Reference

Returns

[*OTF2_SUCCESS*](#) if successful

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

J.17 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_ArchiveHandler which is marked as OTF2_MASTER.

```
#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 \(OTF2_GlobalDefWriter *writerHandle, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_Type type\)*](#)
*Writes a [*Attribute*](#) definition record into the GlobalDefWriter.*
- [*OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallpath \(OTF2_GlobalDefWriter *writerHandle, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region\)*](#)

APPENDIX J. FILE DOCUMENTATION

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_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)

Writes a [LocationGroup](#) 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)

J.17 OTF2_GlobalDefWriter.h File Reference

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)

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.

J.17.1 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 def-

APPENDIX J. FILE DOCUMENTATION

initions 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_ArchiveHandler` which is marked as `OTF2_MASTER`.

Source Template:

templates/OTF2_GlobalDefWriter.templ.h

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.17.2 Function Documentation

J.17.2.1 `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.
out	<i>numberOfDefinitions</i>	Storage for the number of definitions.

Returns

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

J.17.2.2 `OTF2_ErrorCode OTF2_GlobalDefWriter_GetNumberOfLocations (OTF2_GlobalDefWriter * writerHandle, uint64_t * numberOfLocations)`

Returns the current number of written location definitions of a global definition writer.

Parameters

J.17 OTF2_GlobalDefWriter.h File Reference

	<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.

J.17.2.3 `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteAttribute (`
 `OTF2_GlobalDefWriter * writerHandle, OTF2_AttributeRef self,`
 `OTF2_StringRef name, OTF2_Type type)`

Writes a [Attribute](#) definition record into the GlobalDefWriter.

Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this Attribute definition.
<i>name</i>	Name of the attribute. References a String definition.
<i>type</i>	Type of the attribute value.

Since

Version 1.0

Returns

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

J.17.2.4 `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.

Parameters

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

APPENDIX J. FILE DOCUMENTATION

<i>self</i>	The unique identifier for this Callpath definition.
<i>parent</i>	References a Callpath definition.
<i>region</i>	References a Region definition.

Since

Version 1.0

Returns

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

J.17.2.5 `OTF2_StatusCode 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.

Parameters

<i>writerHandle</i>	The writer handle.
<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.

J.17.2.6 `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.

J.17 OTF2_GlobalDefWriter.h File Reference

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.

J.17.2.7 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.

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 <i>OTF2_GROUP_TYPE_MPI_GROUP</i> or <i>OTF2_GROUP_TYPE_MPI_COMM_SELF</i> . References a Group 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 Comm definition.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.0

Returns

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

J.17.2.8 `OTF2_StatusCode 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.

Parameters

<i>writerHandle</i>	The writer handle.
<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.

J.17.2.9 `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.

J.17 OTF2_GlobalDefWriter.h File Reference

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>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_SUCCESS](#) if successful, an error code if an error occurs.

J.17.2.10 `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.

Parameters

<i>writerHandle</i>	The writer handle.
<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

APPENDIX J. FILE DOCUMENTATION

Returns

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

J.17.2.11 `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.

J.17.2.12 `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricClassRecorder (`
`OTF2_GlobalDefWriter * writerHandle, OTF2_MetricRef metricClass,`
`OTF2_LocationRef recorder)`

Writes a [MetricClassRecorder](#) definition record into the GlobalDefWriter.

Parameters

J.17 OTF2_GlobalDefWriter.h File Reference

<i>writerHandle</i>	The writer handle.
<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.

J.17.2.13 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.

J.17.2.14 `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: int64_t, uint64_t, or double.
<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

J.17 OTF2_GlobalDefWriter.h File Reference

Returns

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

J.17.2.15 `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.

Parameters

<i>writerHandle</i>	The writer handle.
<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>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.

J.17.2.16 `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.

Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this Region definition.

APPENDIX J. FILE DOCUMENTATION

<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.

J.17.2.17 [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.

J.17 OTF2_GlobalDefWriter.h File Reference

Since

Version 1.2

Returns

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

J.17.2.18 **OTF2_ErrorCode** OTF2_GlobalDefWriter_WriteString (
OTF2_GlobalDefWriter * *writerHandle*, OTF2_StringRef *self*, const
char * *string*)

Writes a [String](#) definition record into the GlobalDefWriter.

Parameters

<i>writerHandle</i>	The writer handle.
<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.

J.17.2.19 **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.

Parameters

<i>writerHandle</i>	The writer handle.
<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 <i>OTF2_UNDEFINED_SYSTEM_TREE_NODE</i> to indicate that there is no parent. References a SystemTreeNode definition.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.0

Returns

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

J.17.2.20 **OTF2_ErrorCode** **OTF2_GlobalDefWriter_WriteSystemTreeNodeDomain** (
OTF2_GlobalDefWriter * *writerHandle*, OTF2_SystemTreeNodeRef
systemTreeNode, OTF2_SystemTreeDomain *systemTreeDomain*)

Writes a [SystemTreeNodeDomain](#) definition record into the GlobalDefWriter.

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.

Since

Version 1.2

Returns

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

J.17.2.21 **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.

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

J.18 OTF2_GlobalEvtReader.h File Reference

Since

Version 1.2

Returns

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

J.18 OTF2_GlobalEvtReader.h File Reference

This is the global event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_EvtReader.h>
#include <otf2/OTF2_GlobalEvtReaderCallbacks.h>
```

Functions

- [OTF2_ErrorCode OTF2_GlobalEvtReader_HasEvent](#) ([OTF2_GlobalEvtReader](#) *reader, int *flag)
Has more events.
- [OTF2_ErrorCode OTF2_GlobalEvtReader_ReadEvent](#) ([OTF2_GlobalEvtReader](#) *reader)
Triggers the callback for the next event record.
- [OTF2_ErrorCode OTF2_GlobalEvtReader_ReadEvents](#) ([OTF2_GlobalEvtReader](#) *reader, uint64_t recordsToRead, uint64_t *recordsRead)
Reads the given number of records from the global event reader.
- [OTF2_ErrorCode OTF2_GlobalEvtReader_SetCallbacks](#) ([OTF2_GlobalEvtReader](#) *reader, const [OTF2_GlobalEvtReaderCallbacks](#) *callbacks, void *userData)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

J.18.1 Detailed Description

This is the global event reader.

Maintainer:

Michael Wagner <michael.wagner@zih.tu-dresden.de>

APPENDIX J. FILE DOCUMENTATION

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

Used to read from multiple local event readers, and provide them in a timely ordered sequence.

J.18.2 Function Documentation

J.18.2.1 **OTF2_StatusCode 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.

J.18.2.2 **OTF2_StatusCode 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.
---------------	---

Returns

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

J.18 OTF2_GlobalEvtReader.h File Reference

J.18.2.3 `OTF2_StatusCode 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 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> .

Returns

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

J.18.2.4 `OTF2_StatusCode 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. <code>OTF2_GlobalEvtReaderCallbacks_New</code> .
<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.

J.19 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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

- 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.
- 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)

APPENDIX J. FILE DOCUMENTATION

Callback for the `MpiSend` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_OmpAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the `OmpAcquireLock` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_OmpFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t numberOfRequestedThreads)`

Callback for the `OmpFork` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_OmpJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList)`

Callback for the `OmpJoin` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_OmpReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the `OmpReleaseLock` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_OmpTaskComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)`

Callback for the `OmpTaskComplete` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)`

Callback for the `OmpTaskCreate` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_OmpTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)`

Callback for the `OmpTaskSwitch` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, int64_t value)`

Callback for the `ParameterInt` event record.

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ParameterString)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, OTF2_StringRef string)`

Callback for the `ParameterString` event record.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, uint64_t value)

Callback for the ParameterUnsignedInt event record.

- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)

Callback for the RmaAcquireLock event record.

- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaAtomic)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)

Callback for the RmaAtomic event record.

- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList)

Callback for the RmaCollectiveBegin event record.

- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)

Callback for the RmaCollectiveEnd event record.

- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaGet)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)

Callback for the RmaGet event record.

- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaGroupSync)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)

Callback for the RmaGroupSync event record.

- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaOpCompleteBlocking)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)

Callback for the RmaOpCompleteBlocking event record.

APPENDIX J. FILE DOCUMENTATION

- 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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

- 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_ThreadFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)
Callback for the ThreadFork event record.
- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model)
Callback for the ThreadJoin event record.
- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)
Callback for the ThreadReleaseLock event record.
- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadTaskComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)
Callback for the ThreadTaskComplete event record.
- typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)
Callback for the ThreadTaskCreate event record.

APPENDIX J. FILE DOCUMENTATION

- 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_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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_Leave](#) leaveCallback)
Registers the callback for the Leave event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMeasurementOnOffCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_MeasurementOnOff](#) measurementOnOffCallback)
Registers the callback for the MeasurementOnOff event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMetricCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_Metric](#) metricCallback)
Registers the callback for the Metric event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiCollectiveBeginCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_MpiCollectiveBegin](#) mpiCollectiveBeginCallback)
Registers the callback for the MpiCollectiveBegin event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiCollectiveEndCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_MpiCollectiveEnd](#) mpiCollectiveEndCallback)
Registers the callback for the MpiCollectiveEnd event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_MpiIrecv](#) mpiIrecvCallback)
Registers the callback for the MpiIrecv event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvRequestCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_MpiIrecvRequest](#) mpiIrecvRequestCallback)
Registers the callback for the MpiIrecvRequest event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_MpiIsend](#) mpiIsendCallback)
Registers the callback for the MpiIsend event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCompleteCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_MpiIsendComplete](#) mpiIsendCompleteCallback)
Registers the callback for the MpiIsendComplete event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiRecvCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_MpiRecv](#) mpiRecvCallback)
Registers the callback for the MpiRecv event.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiRequestCancelledCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-MpiRequestCancelled](#) mpiRequestCancelledCallback)
Registers the callback for the MpiRequestCancelled event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiRequestTestCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-MpiRequestTest](#) mpiRequestTestCallback)
Registers the callback for the MpiRequestTest event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiSendCallback](#) ([OTF2_-GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-MpiSend](#) mpiSendCallback)
Registers the callback for the MpiSend event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpAcquireLockCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-OmpAcquireLock](#) ompAcquireLockCallback)
Registers the callback for the OmpAcquireLock event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpForkCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-OmpFork](#) ompForkCallback)
Registers the callback for the OmpFork event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpJoinCallback](#) ([OTF2_-GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-OmpJoin](#) ompJoinCallback)
Registers the callback for the OmpJoin event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpReleaseLockCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-OmpReleaseLock](#) ompReleaseLockCallback)
Registers the callback for the OmpReleaseLock event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCompleteCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-OmpTaskComplete](#) ompTaskCompleteCallback)
Registers the callback for the OmpTaskComplete event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCreateCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-OmpTaskCreate](#) ompTaskCreateCallback)
Registers the callback for the OmpTaskCreate event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetOmpTaskSwitchCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-OmpTaskSwitch](#) ompTaskSwitchCallback)
Registers the callback for the OmpTaskSwitch event.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterIntCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-ParameterInt](#) parameterIntCallback)
Registers the callback for the ParameterInt event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterStringCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-ParameterString](#) parameterStringCallback)
Registers the callback for the ParameterString event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetParameterUnsignedIntCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-ParameterUnsignedInt](#) parameterUnsignedIntCallback)
Registers the callback for the ParameterUnsignedInt event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaAcquireLockCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-RmaAcquireLock](#) rmaAcquireLockCallback)
Registers the callback for the RmaAcquireLock event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaAtomicCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-RmaAtomic](#) rmaAtomicCallback)
Registers the callback for the RmaAtomic event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveBeginCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-RmaCollectiveBegin](#) rmaCollectiveBeginCallback)
Registers the callback for the RmaCollectiveBegin event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveEndCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-RmaCollectiveEnd](#) rmaCollectiveEndCallback)
Registers the callback for the RmaCollectiveEnd event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaGetCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-RmaGet](#) rmaGetCallback)
Registers the callback for the RmaGet event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaGroupSyncCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-RmaGroupSync](#) rmaGroupSyncCallback)
Registers the callback for the RmaGroupSync event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteBlockingCallback](#)
([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_-RmaOpCompleteBlocking](#) rmaOpCompleteBlockingCallback)
Registers the callback for the RmaOpCompleteBlocking event.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaOpCompleteNonBlocking](#) [rmaOpCompleteNonBlockingCallback](#))
Registers the callback for the RmaOpCompleteNonBlocking event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteRemoteCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaOpCompleteRemote](#) [rmaOpCompleteRemoteCallback](#))
Registers the callback for the RmaOpCompleteRemote event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpTestCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaOpTest](#) [rmaOpTestCallback](#))
Registers the callback for the RmaOpTest event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaPutCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaPut](#) [rmaPutCallback](#))
Registers the callback for the RmaPut event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaReleaseLockCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaReleaseLock](#) [rmaReleaseLockCallback](#))
Registers the callback for the RmaReleaseLock event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaRequestLockCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaRequestLock](#) [rmaRequestLockCallback](#))
Registers the callback for the RmaRequestLock event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaSyncCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaSync](#) [rmaSyncCallback](#))
Registers the callback for the RmaSync event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaTryLockCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaTryLock](#) [rmaTryLockCallback](#))
Registers the callback for the RmaTryLock event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaWaitChangeCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaWaitChange](#) [rmaWaitChangeCallback](#))
Registers the callback for the RmaWaitChange event.
- [OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaWinCreateCallback](#) ([OTF2_GlobalEvtReaderCallbacks *globalEvtReaderCallbacks](#), [OTF2_GlobalEvtReaderCallback_-RmaWinCreate](#) [rmaWinCreateCallback](#))
Registers the callback for the RmaWinCreate event.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

- [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_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.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_GlobalEvtReaderCallbacks_SetUnknownCallback](#) ([OTF2_GlobalEvtReaderCallbacks](#) *globalEvtReaderCallbacks, [OTF2_GlobalEvtReaderCallback_Unknown](#) unknownCallback)

Registers the callback for unknown events.

J.19.1 Detailed Description

This defines the callbacks for the global event reader.

Source Template:

templates/OTF2_GlobalEvtReaderCallbacks.tmpl.h

Maintainer:

Dominic Eschweiler <d.eschweiler@fz-juelich.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.19.2 Typedef Documentation

J.19.2.1 `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

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

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

J.19.2.2 `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*](#).

J.19.2.3 `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 .

APPENDIX J. FILE DOCUMENTATION

<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](#).

J.19.2.4 `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 OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>measure- mentMode</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](#).

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

J.19.2.5 `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 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.
<i>metricValues</i>	List of metric values.

Since

Version 1.0

Returns

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

J.19.2.6 `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.).

APPENDIX J. 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.

Since

Version 1.0

Returns

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

J.19.2.7 `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 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>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 <i>communicator</i> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.0

Returns

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

J.19.2.8 `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 <i>communicator</i> .
<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*](#).

APPENDIX J. FILE DOCUMENTATION

J.19.2.9 `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

<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](#).

J.19.2.10 `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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>receiver</i>	MPI rank of receiver 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
<i>requestID</i>	ID of the related request

Since

Version 1.0

Returns

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

J.19.2.11 `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 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

Returns

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

J.19.2.12 `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).

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 <i>communicator</i> .
<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_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

J.19.2.13 `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

J.19 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>requestID</i>	ID of the related request

Since

Version 1.0

Returns

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

J.19.2.14 `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 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

Returns

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

J.19.2.15 `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.
<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 <i>communicator</i> .
<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_CALLBACK_SUCCESS](#) or [OTF2_CALLBACK_INTERRUPT](#).

J.19.2.16 `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 record.

J.19 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.
<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](#).

J.19.2.17 `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 OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<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](#).

J.19.2.18 `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ -
OmpJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void
*userData, OTF2_AttributeList *attributeList)`

Callback for the OmpJoin event record.

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](#).

J.19.2.19 `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.

J.19 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.
<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](#).

J.19.2.20 `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 OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<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*](#).

J.19.2.21 `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.

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 <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 newly created task instance.

Since

Version 1.0

Returns

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

J.19.2.22 `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.

J.19 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.
<i>taskID</i>	Identifier of the now active task instance.

Since

Version 1.0

Returns

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

J.19.2.23 `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 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.

Since

Version 1.0

Returns

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

APPENDIX J. FILE DOCUMENTATION

J.19.2.24 `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.

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](#).

J.19.2.25 `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.
-------------------	---

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

<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.

Since

Version 1.0

Returns

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

J.19.2.26 `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 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*](#).

J.19.2.27 `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 <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 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*](#).

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

J.19.2.28 `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.

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](#).

J.19.2.29 `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.

APPENDIX J. FILE DOCUMENTATION

<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](#).

J.19.2.30 `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 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 received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

Since

Version 1.2

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

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

J.19.2.31 `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 <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<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 <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>group</i>	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.

Since

Version 1.2

Returns

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

J.19.2.32 `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.

APPENDIX J. FILE DOCUMENTATION

An `RmaOpCompleteBlocking` record denotes the local completion of a 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

Returns

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

J.19.2.33 `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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

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

J.19.2.34 `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 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*](#).

J.19.2.35 `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.

APPENDIX J. FILE DOCUMENTATION

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

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](#).

J.19.2.36 `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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

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

J.19.2.37 `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 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 released, if multiple locks are defined on a window.

Since

Version 1.2

Returns

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

J.19.2.38 `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 J. 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 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](#).

J.19.2.39 `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.

J.19 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*](#).

J.19.2.40 `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 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 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 J. FILE DOCUMENTATION

J.19.2.41 `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](#).

J.19.2.42 `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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

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

J.19.2.43 `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 OTF2_Reader_RegisterGlobalEvtCallbacks or OTF2_GlobalEvtReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window destroyed. 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*](#).

J.19.2.44 `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 J. 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>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](#).

J.19.2.45 `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>numberOfRequestedThreads</i>	Requested size of the team.

Since

Version 1.2

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

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

J.19.2.46 `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.

Since

Version 1.2

Returns

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

J.19.2.47 `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ - ThreadReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the ThreadReleaseLock event record.

An ThreadReleaseLock record marks that a thread releases an lock.

Parameters

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

APPENDIX J. FILE DOCUMENTATION

<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.

Since

Version 1.2

Returns

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

J.19.2.48 `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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Since

Version 1.2

Returns

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

J.19.2.49 `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 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. (This is redundant, remove?)
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

Since

Version 1.2

Returns

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

APPENDIX J. FILE DOCUMENTATION

J.19.2.50 `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](#).

J.19.2.51 `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.

Parameters

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

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

<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](#).

J.19.2.52 `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.

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 J. FILE DOCUMENTATION

J.19.2.53 `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.

Returns

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

J.19.3 Function Documentation

J.19.3.1 `void OTF2_GlobalEvtReaderCallbacks_Clear (OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks)`

Clears a struct for the global event callbacks.

Parameters

<i>globalEvtReaderCallbacks</i>	Handle to a struct previously allocated with OTF2_GlobalEvtReaderCallbacks_New .
---------------------------------	--

J.19.3.2 `void OTF2_GlobalEvtReaderCallbacks_Delete (OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks)`

Deallocates a struct for the global event callbacks.

Parameters

<i>globalEvtReaderCallbacks</i>	Handle to a struct previously allocated with OTF2_GlobalEvtReaderCallbacks_New .
---------------------------------	--

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

J.19.3.3 **OTF2_GlobalEvtReaderCallbacks*** **OTF2_GlobalEvtReaderCallbacks_New** (
 void)

Allocates a new struct for the event callbacks.

Returns

A newly allocated struct of type [OTF2_GlobalEvtReaderCallbacks](#).

J.19.3.4 **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.

Returns

[OTF2_SUCCESS](#) if successful

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

J.19.3.5 **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.

APPENDIX J. FILE DOCUMENTATION

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.6 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.7 **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.
<i>measurementOnOffCallback</i>	Function which should be called for all MeasurementOnOff events.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.8 OTF2_ErrorCode 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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.9 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.

APPENDIX J. FILE DOCUMENTATION

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.10 **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 <code>MpiCollectiveEnd</code> events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.11 **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.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.12 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks.-SetMpiIrecvRequestCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_-MpiIrecvRequest *mpiIrecvRequestCallback*)

Registers the callback for the `MpiIrecvRequest` event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all <code>MpiIrecvRequest</code> events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.13 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks.SetMpIsendCallback (OTF2_GlobalEvtReaderCallbacks * *globalEvtReaderCallbacks*, OTF2_GlobalEvtReaderCallback_MpIsend *mpIsendCallback*)

Registers the callback for the `MpiIsend` event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpIsendCallback</i>	Function which should be called for all <code>MpiIsend</code> events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.14 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetMpiSendCompleteCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_MpiSendComplete** *mpiSendCompleteCallback*)

Registers the callback for the `MpiSendComplete` event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSendCompleteCallback</i>	Function which should be called for all <code>MpiSendComplete</code> events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.15 **OTF2_StatusCode** **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 <code>MpiRecv</code> events.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.16 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.17 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetMpiRequestTestCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-MpiRequestTest** *mpiRequestTestCallback*)

Registers the callback for the `MpiRequestTest` event.

Parameters

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

APPENDIX J. FILE DOCUMENTATION

<i>mpiRequestTestCallback</i>	Function which should be called for all MpiRequestTest events.
-------------------------------	--

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.18 **OTF2_ErrorCode** **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 MpiSend events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.19 **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.
---------------------------------	---------------------------

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>ompAcquireLockCallback</i>	Function which should be called for all OmpAcquireLock events.
-------------------------------	--

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.20 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.21 **OTF2_ErrorCode** **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.
---------------------------------	---------------------------

APPENDIX J. FILE DOCUMENTATION

<i>ompJoin-Callback</i>	Function which should be called for all OmpJoin events.
-------------------------	---

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.22 **OTF2_ErrorCode** **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>ompReleaseLock-Callback</i>	Function which should be called for all OmpReleaseLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.23 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetOmpTaskCompleteCallback** (**OTF2_GlobalEvtReaderCallbacks** * **globalEvtReaderCallbacks**, **OTF2_GlobalEvtReaderCallback_-OmpTaskComplete** **ompTaskCompleteCallback**)

Registers the callback for the OmpTaskComplete event.

Parameters

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCompleteCallback</i>	Function which should be called for all OmpTaskComplete events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.24 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-SetOmpTaskCreateCallback (OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-OmpTaskCreate *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCreateCallback</i>	Function which should be called for all OmpTaskCreate events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.19.3.25 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.19.3.26 **OTF2_ErrorCode** **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks* argument

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

J.19.3.27 **OTF2_ErrorCode** **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.28 **OTF2_ErrorCode** **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 ParameterUnsignedInt events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`

argument

J.19.3.29 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks** .
SetRmaAcquireLockCallback (**OTF2_GlobalEvtReaderCallbacks**
* *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback** -
RmaAcquireLock *rmaAcquireLockCallback*)

Registers the callback for the RmaAcquireLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaAcquireLockCallback</i>	Function which should be called for all RmaAcquireLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks*
argument

J.19.3.30 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid *defReaderCallbacks*

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

argument

J.19.3.31 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks`
argument

J.19.3.32 **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.

APPENDIX J. FILE DOCUMENTATION

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.33 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.34 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetRmaGroupSyncCallback**
(**OTF2_GlobalEvtReaderCallbacks** *
globalEvtReaderCallbacks, **OTF2_GlobalEvtReaderCallback_-RmaGroupSync** *rmaGroupSyncCallback*)

Registers the callback for the RmaGroupSync event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaGroupSyncCallback</i>	Function which should be called for all RmaGroupSync events.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.35 **OTF2_ErrorCode** **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.36 **OTF2_ErrorCode** **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.
---------------------------------	---------------------------

APPENDIX J. FILE DOCUMENTATION

<i>rmaOp-CompleteNon-Blocking-Callback</i>	Function which should be called for all RmaOpCompleteNonBlocking events.
--	--

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.37 **OTF2_ErrorCode** **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>rmaOp-CompleteRemoteCallback</i>	Function which should be called for all RmaOpCompleteRemote events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.38 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetRmaOpTestCallback** (**OTF2_GlobalEvtReaderCallbacks** * **globalEvtReaderCallbacks**, **OTF2_GlobalEvtReaderCallback_RmaOpTest** *rmaOpTestCallback*)

Registers the callback for the RmaOpTest event.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpTest-Callback</i>	Function which should be called for all RmaOpTest events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.39 **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.
<i>rmaPut-Callback</i>	Function which should be called for all RmaPut events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.40 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-SetRmaReleaseLockCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-RmaReleaseLock** *rmaReleaseLockCallback*)

Registers the callback for the RmaReleaseLock event.

Parameters

APPENDIX J. FILE DOCUMENTATION

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaReleaseLockCallback</i>	Function which should be called for all RmaReleaseLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.41 **OTF2_StatusCode** **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.42 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_SetRmaSyncCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_RmaSync** *rmaSyncCallback*)

Registers the callback for the RmaSync event.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaSyncCallback</i>	Function which should be called for all RmaSync events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.43 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.44 **OTF2_StatusCode** **OTF2_GlobalEvtReaderCallbacks_-SetRmaWaitChangeCallback**
(**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-RmaWaitChange** *rmaWaitChangeCallback*)

Registers the callback for the RmaWaitChange event.

APPENDIX J. FILE DOCUMENTATION

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWaitChangeCallback</i>	Function which should be called for all RmaWaitChange events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.45 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.46 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_**
SetRmaWinDestroyCallback (**OTF2_GlobalEvtReaderCallbacks ***
globalEvtReaderCallbacks, **OTF2_GlobalEvtReaderCallback_**
RmaWinDestroy **rmaWinDestroyCallback)**

Registers the callback for the RmaWinDestroy event.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWinDestroyCallback</i>	Function which should be called for all RmaWinDestroy events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.47 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.48 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_SetThreadForkCallback** (**OTF2_GlobalEvtReaderCallbacks** * *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_ThreadFork** *threadForkCallback*)

Registers the callback for the ThreadFork event.

APPENDIX J. FILE DOCUMENTATION

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadForkCallback</i>	Function which should be called for all ThreadFork events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.49 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.50 **OTF2_ErrorCode** **OTF2_GlobalEvtReaderCallbacks_-**
SetThreadReleaseLockCallback (**OTF2_GlobalEvtReaderCallbacks**
* *globalEvtReaderCallbacks*, **OTF2_GlobalEvtReaderCallback_-**
ThreadReleaseLock *threadReleaseLockCallback*
)

Registers the callback for the ThreadReleaseLock event.

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadReleaseLockCallback</i>	Function which should be called for all ThreadReleaseLock events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

```
J.19.3.51  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.
<i>threadTaskCompleteCallback</i>	Function which should be called for all ThreadTaskComplete events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.19.3.52 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.53 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19 OTF2_GlobalEvtReaderCallbacks.h File Reference

J.19.3.54 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.19.3.55 **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.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.19.3.56 **OTF2_ErrorCode** **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>unknownCallback</i>	Function which should be called for all unknown events.

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.20 OTF2_GlobalSnapReader.h File Reference

This is the global snapshot event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_SnapReader.h>
#include <otf2/OTF2_GlobalSnapReaderCallbacks.h>
```

Functions

- **OTF2_ErrorCode** **OTF2_GlobalSnapReader_ReadSnapshots** (**OTF2_GlobalSnapReader** *reader, uint64_t recordsToRead, uint64_t *recordsRead)
Reads the given number of records from the global snap event reader.
- **OTF2_ErrorCode** **OTF2_GlobalSnapReader_SetCallbacks** (**OTF2_GlobalSnapReader** *reader, const **OTF2_GlobalSnapReaderCallbacks** *callbacks, void *userData)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

J.20 OTF2_GlobalSnapReader.h File Reference

J.20.1 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.

J.20.2 Function Documentation

J.20.2.1 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.
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.

Since

Version 1.2

Returns

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

J.20.2.2 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.

APPENDIX J. FILE DOCUMENTATION

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. OTF2_GlobalSnapReaderCallbacks_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.

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

This defines the callbacks for the global snap reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

Typedefs

- typedef [OTF2_CallbackCode](#)(* [OTF2_GlobalSnapReaderCallback_Enter](#))([OTF2_LocationRef](#) locationID, [OTF2_TimeStamp](#) snapTime, void *userData, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) origEventTime, [OTF2_RegionRef](#) region)
Callback for the Enter snap record.
- typedef [OTF2_CallbackCode](#)(* [OTF2_GlobalSnapReaderCallback_MeasurementOnOff](#))([OTF2_LocationRef](#) locationID, [OTF2_TimeStamp](#) snapTime, void *userData, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) origEventTime, [OTF2_MeasurementMode](#) measurementMode)
Callback for the MeasurementOnOff snap record.

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

- 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.
- 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,

APPENDIX J. FILE DOCUMENTATION

`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)`

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.

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

- 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)
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)

APPENDIX J. FILE DOCUMENTATION

Registers the callback for the MeasurementOnOff snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMetricCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_Metric metricCallback](#))

Registers the callback for the Metric snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveBeginCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin mpiCollectiveBeginCallback](#))

Registers the callback for the MpiCollectiveBegin snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveEndCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_MpiCollectiveEnd mpiCollectiveEndCallback](#))

Registers the callback for the MpiCollectiveEnd snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_MpiIrecv mpiIrecvCallback](#))

Registers the callback for the MpiIrecv snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvRequestCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_MpiIrecvRequest mpiIrecvRequestCallback](#))

Registers the callback for the MpiIrecvRequest snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIsendCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_MpiIsend mpiIsendCallback](#))

Registers the callback for the MpiIsend snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiIsendCompleteCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_MpiIsendComplete mpiIsendCompleteCallback](#))

Registers the callback for the MpiIsendComplete snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiRecvCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_MpiRecv mpiRecvCallback](#))

Registers the callback for the MpiRecv snap.

- [OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiSendCallback](#) ([OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks](#), [OTF2_GlobalSnapReaderCallback_MpiSend mpiSendCallback](#))

Registers the callback for the MpiSend snap.

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

- [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)
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.

J.21.1 Detailed Description

This defines the callbacks for the global snap reader.

Source Template:

templates/OTF2_GlobalSnapReaderCallbacks.tmpl.h

J.21.2 Typedef Documentation

J.21.2.1 `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 happened.
<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](#).

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

J.21.2.2 `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 happened.
<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](#).

J.21.2.3 `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 denotes an [OTF2_METRIC_VALUE_RELATIVE](#) mode the value indicates the accumulation of all previous metric values recorded.

APPENDIX J. 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>origEvent-Time</i>	The original time this event happened.
<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.
<i>metricValues</i>	List of metric values.

Since

Version 1.2

Returns

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

J.21.2.4 `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.

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

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

J.21.2.5 `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 *MpiCollectiveBeginSnaps* 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.
<i>userData</i>	User data as set by OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEventTime</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 <i>communicator</i> .
<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](#).

J.21.2.6 `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 [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>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 <i>communicator</i> .
<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](#).

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

J.21.2.7 `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 happended.
<i>requestID</i>	ID of the requested receive

Since

Version 1.2

Returns

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

J.21.2.8 `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

APPENDIX J. FILE DOCUMENTATION

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 <i>communicator</i> .
<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](#).

J.21.2.9 `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.

J.21 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>requestID</i>	ID of the related request

Since

Version 1.2

Returns

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

J.21.2.10 `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_
MpiRecv)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime,
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp
origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t
msgTag, uint64_t msgLength)`

Callback for the MpiRecv snap record.

This record exists for each [MpiRecv](#) event where the matching send message event did not occur on the remote location before the snapshot. This could either be an [MpiSend](#) or an [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>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>sender</i>	MPI rank of sender in <i>communicator</i> .
<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](#).

J.21.2.11 `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 [MpiSend](#) 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>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 <i>communicator</i> .
<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](#).

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

J.21.2.12 `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 happened.
<i>lockID</i>	ID of the lock.
<i>acqui-si-tionOrder</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](#).

J.21.2.13 `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.
-------------------	--

APPENDIX J. FILE DOCUMENTATION

<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>num-berOfRe-quest-edThreads</i>	Requested size of the team.

Since

Version 1.2

Returns

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

J.21.2.14 `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 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>taskID</i>	Identifier of the newly created task instance.

Since

Version 1.2

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

Returns

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

J.21.2.15 `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.
<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 now active task instance.

Since

Version 1.2

Returns

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

J.21.2.16 `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_
ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp
snapTime, void *userData, OTF2_AttributeList *attributeList,
OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter,
int64_t value)`

Callback for the ParameterInt snap record.

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

APPENDIX J. 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>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](#).

```
J.21.2.17  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 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.

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

<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](#).

J.21.2.18 `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](#).

APPENDIX J. FILE DOCUMENTATION

J.21.2.19 `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_ - SnapshotEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t contReadPos)`

Callback for the SnapshotEnd snap record.

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

Parameters

<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](#).

J.21.2.20 `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.

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

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>num-berOfRecord</i>	Number of snapshot event records in this snapshot. Excluding the <i>Snap-shotEnd</i> record.

Since

Version 1.2

Returns

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

J.21.2.21 `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 OTF2_Reader_RegisterGlobalSnapCallbacks or OTF2_GlobalSnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this snap.

Since

Version 1.2

Returns

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

J.21.2.22 `typedef struct OTF2_GlobalSnapReaderCallbacks_struct OTF2_GlobalSnapReaderCallbacks`

Opaque struct which holds all snap record callbacks.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.2

J.21.3 Function Documentation

**J.21.3.1 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

**J.21.3.2 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

**J.21.3.3 OTF2_GlobalSnapReaderCallbacks* OTF2_GlobalSnapReaderCallbacks_-
New (void)**

Allocates a new struct for the snap callbacks.

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

A newly allocated struct of type [OTF2_GlobalSnapReaderCallbacks](#).

J.21.3.4 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

J.21.3.5 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.
-----------------------------------	---------------------------

APPENDIX J. FILE DOCUMENTATION

<i>measurementOnOff-Callback</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

J.21.3.6 **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

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

**J.21.3.7 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_-
SetMpiCollectiveBeginCallback (OTF2_GlobalSnapReaderCallbacks
* *globalSnapReaderCallbacks*, OTF2_GlobalSnapReaderCallback_-
MpiCollectiveBegin *mpiCollectiveBeginCallback*
)**

Registers the callback for the MpiCollectiveBegin snap.

Parameters

<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

**J.21.3.8 OTF2_ErrorCode 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.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.21.3.9 **OTF2_StatusCode** **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>mpiIrecv-Callback</i>	Function which should be called for all <code>MpiIrecv</code> snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.21.3.10 **OTF2_StatusCode** **OTF2_GlobalSnapReaderCallbacks_SetMpiIrecvRequestCallback** (**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*, **OTF2_GlobalSnapReaderCallback_MpiIrecvRequest** *mpiIrecvRequestCallback*)

Registers the callback for the `MpiIrecvRequest` snap.

Parameters

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all MpiIrecvRequest snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.21.3.11 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

J.21.3.12 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_-SetMpiIsendCompleteCallback** (**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*, **OTF2_GlobalSnapReaderCallback_-MpiIsendComplete** *mpiIsendCompleteCallback*)

Registers the callback for the MpiIsendComplete snap.

Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<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

J.21.3.13 **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

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.21.3.14 **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

J.21.3.15 **OTF2_StatusCode** **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

J.21.3.16 **OTF2_StatusCode** **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

J.21.3.17 **OTF2_StatusCode** **OTF2_GlobalSnapReaderCallbacks_SetOmpTaskCreateCallback**
(**OTF2_GlobalSnapReaderCallbacks** *
globalSnapReaderCallbacks, **OTF2_GlobalSnapReaderCallback_OmpTaskCreate** *ompTaskCreateCallback*)

Registers the callback for the OmpTaskCreate snap.

Parameters

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

<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

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.21.3.18 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

APPENDIX J. FILE DOCUMENTATION

J.21.3.19 **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.
<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

J.21.3.20 **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

J.21 OTF2_GlobalSnapReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.21.3.21 **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 <code>ParameterUnsignedInt</code> snaps.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.21.3.22 **OTF2_ErrorCode** **OTF2_GlobalSnapReaderCallbacks_-SetSnapshotEndCallback** (**OTF2_GlobalSnapReaderCallbacks** * *globalSnapReaderCallbacks*, **OTF2_GlobalSnapReaderCallback_-SnapshotEnd** *snapshotEndCallback*)

Registers the callback for the `SnapshotEnd` snap.

Parameters

APPENDIX J. FILE DOCUMENTATION

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>snapshotEnd-Callback</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

J.21.3.23 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>snapshotStart-Callback</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

J.22 OTF2_IdMap.h File Reference

J.21.3.24 `OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetUnknownCallback
(OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks,
OTF2_GlobalSnapReaderCallback_Unknown unknownCallback)`

Registers the callback for unknown snaps.

Parameters

<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

J.22 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_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)

J.22.1 Detailed Description

Identifier mapping data structure, based on Scalasca's `epk_idmap.h`.

Maintainer:

Christian Rössel <c.roessel@fz-juelich.de>

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

J.22 OTF2_IdMap.h File Reference

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.

J.22.2 Typedef Documentation

J.22.2.1 typedef struct OTF2_IdMap_struct OTF2_IdMap

Opaque data structure representing an ID mapping table.

J.22.2.2 typedef uint8_t OTF2_IdMapMode

Wrapper around enum OTF2_IdMapMode_enum, so that it is guaranteed that it is a uint8_t

J.22.3 Enumeration Type Documentation

J.22.3.1 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

J.22.4 Function Documentation

J.22.4.1 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

APPENDIX J. FILE DOCUMENTATION

<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.

J.22.4.2 OTF2_StatusCode OTF2_IdMap_Clear (OTF2_IdMap * *instance*)

Removes all entries in the given mapping table *instance*. It can be used, e.g., to reuse an mapping table object for new input data.

Parameters

<i>instance</i>	Object to remove entries from.
-----------------	--------------------------------

Returns

OTF2_SUCCESS, or error code.

J.22.4.3 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 can not 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.

J.22.4.4 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*.

J.22 OTF2_IdMap.h File Reference

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.

J.22.4.5 OTF2_IdMap* OTF2_IdMap_CreateFromUint64Array (uint64_t *length*, const uint64_t * *mappings*, bool *optimizeSize*)

Creates and returns a new instance of *OTF2_IdMap* from the array given by *mappings*.

This creates always a DENSE mapping if *optimizeSize* is false. If it is true, it creates a SPARSE mapping, if the number of non-identitiy entries in the *mappings* array (ie. `mapping[i] != i`) is less than half the *length*.

Returns NULL when *optimizeSize* is true and the number of non-identitiy entries equals zero, ie. the given map is the identity map.

Parameters

<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.

J.22.4.6 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
-----------------	--------------------

APPENDIX J. FILE DOCUMENTATION

J.22.4.7 **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.

J.22.4.8 **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.

J.22.4.9 **OTF2_ErrorCode** **OTF2_IdMap_GetSize** (**const** **OTF2_IdMap** * *instance*, **uint64_t** * *size*)

Returns the actual number of entries stored in the given **OTF2_IdMap** *instance*.

Parameters

	<i>instance</i>	Queried object.
out	<i>size</i>	Number of entries.

J.23 OTF2_Marker.h File Reference

Returns

OTF2_SUCCESS, or error code.

J.22.4.10 `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.

J.23 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 Marker definition.

Typedefs

- `typedef uint32_t OTF2_MarkerRef`
Type used to indicate a reference to a Marker 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` }

J.23.1 Detailed Description

This file provides types and enums for markers.

J.23.2 Enumeration Type Documentation

J.23.2.1 enum `OTF2_MarkerScope_enum`

A user marker does have a scope of it validity.

Enumerator:

OTF2_MARKER_SCOPE_GLOBAL The user marker has a global scope (could also be NONE).

OTF2_MARKER_SCOPE_LOCATION The user marker has a scope of a location.

OTF2_MARKER_SCOPE_LOCATION_GROUP The user marker has a scope of a location group.

OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE The user marker has a scope of a system tree.

OTF2_MARKER_SCOPE_GROUP The user marker has a scope of a group.

OTF2_MARKER_SCOPE_COMM The user marker has a scope of a communicator.

J.24 OTF2_MarkerReader.h File Reference

J.23.2.2 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.

J.24 OTF2_MarkerReader.h File Reference

This file provides all routines that read marker records.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Marker.h>
#include <otf2/OTF2_MarkerReaderCallbacks.h>
```

Functions

- [OTF2_ErrorCode OTF2_MarkerReader_ReadMarkers](#) ([OTF2_MarkerReader](#) *reader, uint64_t recordsToRead, uint64_t *recordsRead)

After callback registration, the markers could be read with the following function. The user of this function tells the system how many markers it is able to handle (recordsToRead) and the function returns how many markers where in the stream (recordsRead). It should usually be the case that both values are the same. If this is not the case, then there where less records than requested in the stream.

- [OTF2_ErrorCode OTF2_MarkerReader_SetCallbacks](#) ([OTF2_MarkerReader](#) *reader, const [OTF2_MarkerReaderCallbacks](#) *callbacks, void *userData)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

J.24.1 Detailed Description

This file provides all routines that read marker records.

J.24.2 Function Documentation

J.24.2.1 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>record-sToRead</i>	How many records have to be read next.
<i>record-sRead</i>	How many records where read?

Since

Version 1.2

Returns

OTF2_ErrorCode with !=OTF2_SUCCESS if there was an error.

J.24.2.2 OTF2_ErrorCode OTF2_MarkerReader_SetCallbacks (
OTF2_MarkerReader * *reader*, const OTF2_MarkerReaderCallbacks
*** *callbacks*, void * *userData*)**

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

Parameters

<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.

J.25 OTF2_MarkerReaderCallbacks.h File Reference

Since

Version 1.2

Returns

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

J.25 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 Marker definition record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_MarkerReaderCallback_Marker](#))(void *userData, [OTF2_TimeStamp](#) timestamp, [OTF2_TimeStamp](#) duration, [OTF2_MarkerRef](#) marker, [OTF2_MarkerScope](#) scope, uint64_t scopeRef, const char *text)

Function pointer definition for the callback which is triggered by a Marker record.

- typedef [OTF2_CallbackCode](#)(* [OTF2_MarkerReaderCallback_Unknown](#))(void *userData)

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

- typedef struct OTF2_MarkerReaderCallbacks_struct [OTF2_MarkerReaderCallbacks](#)

Opaque struct which 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 Marker 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.

J.25.1 Detailed Description

This defines the callbacks for the marker reader.

J.25.2 Typedef Documentation

J.25.2.1 `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 Marker 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.

J.25 OTF2_MarkerReaderCallbacks.h File Reference

<i>marker-Group</i>	Group name, e.g., "MUST", ...
<i>markerCategory</i>	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_CALLBACK_SUCCESS*](#) or [*OTF2_CALLBACK_INTERRUPT*](#).

J.25.2.2 `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 <i>OTF2_Reader_RegisterMarkerCallbacks</i> or <i>OTF2_MarkerReader_SetCallbacks</i> .
<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 scope.
<i>text</i>	A textual description for this marker.

Since

Version 1.2

Returns

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

APPENDIX J. FILE DOCUMENTATION

J.25.2.3 `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 OTF2_Reader_RegisterMarkerCallbacks or OTF2_MarkerReader_SetCallbacks .
-----------------	---

Since

Version 1.2

Returns

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

J.25.3 Function Documentation

J.25.3.1 `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 OTF2_MarkerReaderCallbacks_New .
--------------------------------	---

J.25.3.2 `void OTF2_MarkerReaderCallbacks_Delete (OTF2_MarkerReaderCallbacks * markerReaderCallbacks)`

Deallocates a struct for the marker callbacks.

Since

Version 1.2

J.25 OTF2_MarkerReaderCallbacks.h File Reference

Parameters

<i>marker-Reader-Callbacks</i>	Handle to a struct previously allocated with OTF2_MarkerReaderCallbacks_New .
--------------------------------	---

J.25.3.3 `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](#).

J.25.3.4 `OTF2_ErrorCode OTF2_MarkerReaderCallbacks_SetDefMarkerCallback (OTF2_MarkerReaderCallbacks * markerReaderCallbacks, OTF2_MarkerReaderCallback_DefMarker defMarkerCallback)`

Registers the callback for the Marker definition.

Parameters

<i>marker-Reader-Callbacks</i>	Struct for all callbacks.
<i>defMarker-Callback</i>	Function which should be called for all Marker definitions.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful

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

APPENDIX J. FILE DOCUMENTATION

**J.25.3.5 `OTF2_ErrorCode` `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 Marker records.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful

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

**J.25.3.6 `OTF2_ErrorCode` `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.
<i>unknown-Callback</i>	Function which should be called for all unknown definitions.

Since

Version 1.2

Returns

[OTF2_SUCCESS](#) if successful

J.26 OTF2_MarkerWriter.h File Reference

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

J.26 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, uint64_t time, uint64_t duration, [*OTF2_MarkerRef*](#) marker, [*OTF2_MarkerScope*](#) scope, uint64_t scopeRef, const char *text)

Write a marker record.

J.26.1 Detailed Description

This file provides all routines that write marker records.

J.26.2 Function Documentation

J.26.2.1 `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>	Category name 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.

J.26.2.2 `OTF2_ErrorCode` `OTF2_MarkerWriter_WriteMarker` (`OTF2_MarkerWriter`
`* writerHandle`, `uint64_t time`, `uint64_t 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>time</i>	Time of the marker.
<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: <i>OTF2_MARKER_SCOPE_GLOBAL</i> , <i>OTF2_MARKER_SCOPE_LOCATION</i> , <i>OTF2_MARKER_SCOPE_LOCATION_GROUP</i> , <i>OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE</i> , <i>OTF2_MARKER_SCOPE_GROUP</i> , or <i>OTF2_MARKER_SCOPE_COMM</i> .

J.27 OTF2_Reader.h File Reference

<i>scopeRef</i>	The reference to an element of the scope of this marker. Depends on scope.
<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.

J.27 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)
Close a reader handle.
- [OTF2_ErrorCode OTF2_Reader_CloseDefReader](#) ([OTF2_Reader](#) *reader, [OTF2_DefReader](#) *defReader)
Close a local definition reader.
- [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)

APPENDIX J. FILE DOCUMENTATION

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_CloseSnapReader](#) ([OTF2_Reader](#) *reader, [OTF2_SnapReader](#) *snapReader)

Close a local snapshot reader.

- [OTF2_ErrorCode](#) [OTF2_Reader_CloseThumbReader](#) ([OTF2_Reader](#) *reader, [OTF2_ThumbReader](#) *thumbReader)

Close an opened thumbnail reader.

- [OTF2_ErrorCode](#) [OTF2_Reader_GetBoolProperty](#) ([OTF2_Reader](#) *reader, const char *name, bool *value)

Get the value of the named trace file property as boolean.

- [OTF2_ErrorCode](#) [OTF2_Reader_GetChunkSize](#) ([OTF2_Reader](#) *reader, uint64_t *chunkSizeEvents, uint64_t *chunkSizeDefinitions)

Get event and definition chunk sizes.

- [OTF2_ErrorCode](#) [OTF2_Reader_GetCompression](#) ([OTF2_Reader](#) *reader, [OTF2_Compression](#) *compression)

Get copression mode.

- [OTF2_ErrorCode](#) [OTF2_Reader_GetCreator](#) ([OTF2_Reader](#) *reader, char **creator)

Get creator name.

- [OTF2_DefReader](#) * [OTF2_Reader_GetDefReader](#) ([OTF2_Reader](#) *reader, [OTF2_LocationRef](#) location)

Get a local definition reader.

- [OTF2_ErrorCode](#) [OTF2_Reader_GetDescription](#) ([OTF2_Reader](#) *reader, char **description)

Get description.

- [OTF2_EvtReader](#) * [OTF2_Reader_GetEvtReader](#) ([OTF2_Reader](#) *reader, [OTF2_LocationRef](#) location)

Get a local event reader.

- [OTF2_ErrorCode](#) [OTF2_Reader_GetFileSubstrate](#) ([OTF2_Reader](#) *reader, [OTF2_FileSubstrate](#) *substrate)

Get file substrate information.

J.27 OTF2_Reader.h File Reference

- [OTF2_GlobalDefReader](#) * [OTF2_Reader_GetGlobalDefReader](#) ([OTF2_Reader](#) *reader)
Get a global definition reader.
- [OTF2_GlobalEvtReader](#) * [OTF2_Reader_GetGlobalEvtReader](#) ([OTF2_Reader](#) *reader)
Get a global event reader.
- [OTF2_GlobalSnapReader](#) * [OTF2_Reader_GetGlobalSnapReader](#) ([OTF2_Reader](#) *reader)
Get a global snap reader.
- [OTF2_ErrorCode](#) [OTF2_Reader_GetMachineName](#) ([OTF2_Reader](#) *reader, char **machineName)
Get machine name.
- [OTF2_MarkerReader](#) * [OTF2_Reader_GetMarkerReader](#) ([OTF2_Reader](#) *reader)
Get a marker reader.
- [OTF2_MarkerWriter](#) * [OTF2_Reader_GetMarkerWriter](#) ([OTF2_Reader](#) *reader)
Get a marker writer.
- [OTF2_ErrorCode](#) [OTF2_Reader_GetNumberOfGlobalDefinitions](#) ([OTF2_Reader](#) *reader, uint64_t *numberOfDefinitions)
Get number of global definitions.
- [OTF2_ErrorCode](#) [OTF2_Reader_GetNumberOfLocations](#) ([OTF2_Reader](#) *reader, uint64_t *numberOfLocations)
Get number of locations.
- [OTF2_ErrorCode](#) [OTF2_Reader_GetNumberOfSnapshots](#) ([OTF2_Reader](#) *reader, uint32_t *number)
Get number of snapshots.
- [OTF2_ErrorCode](#) [OTF2_Reader_GetNumberOfThumbnails](#) ([OTF2_Reader](#) *reader, uint32_t *number)
Get number of thumbs.
- [OTF2_ErrorCode](#) [OTF2_Reader_GetProperty](#) ([OTF2_Reader](#) *reader, const char *name, char **value)
Get the value of the named trace file property.
- [OTF2_ErrorCode](#) [OTF2_Reader_GetPropertyNames](#) ([OTF2_Reader](#) *reader, uint32_t *numberOfProperties, char ***names)
Get the names of all trace file properties.
- [OTF2_SnapReader](#) * [OTF2_Reader_GetSnapReader](#) ([OTF2_Reader](#) *reader, [OTF2_LocationRef](#) location)
Get a local snapshot reader.

APPENDIX J. FILE DOCUMENTATION

- `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_ReadAllGlobalDefinitions (OTF2_Reader *reader, OTF2_GlobalDefReader *defReader, uint64_t *definitionsRead)`
Read all definitions via a global definition reader.
- `OTF2_ErrorCode OTF2_Reader_ReadAllGlobalEvents (OTF2_Reader *reader, OTF2_GlobalEvtReader *evtReader, uint64_t *eventsRead)`
Read all events via a global event reader.
- `OTF2_ErrorCode OTF2_Reader_ReadAllGlobalSnapshots (OTF2_Reader *reader, OTF2_GlobalSnapReader *snapReader, uint64_t *recordsRead)`
Read all records via a global snapshot reader.
- `OTF2_ErrorCode OTF2_Reader_ReadAllLocalDefinitions (OTF2_Reader *reader, OTF2_DefReader *defReader, uint64_t *definitionsRead)`
Read all definitions via a local definition reader.
- `OTF2_ErrorCode OTF2_Reader_ReadAllLocalEvents (OTF2_Reader *reader, OTF2_EvtReader *evtReader, uint64_t *eventsRead)`
Read all events via a local event reader.
- `OTF2_ErrorCode OTF2_Reader_ReadAllLocalSnapshots (OTF2_Reader *reader, OTF2_SnapReader *snapReader, uint64_t *recordsRead)`
Read all records via a local snapshot reader.
- `OTF2_ErrorCode OTF2_Reader_ReadAllMarkers (OTF2_Reader *reader, OTF2_MarkerReader *markerReader, uint64_t *markersRead)`
Read all markers via a marker reader.
- `OTF2_ErrorCode OTF2_Reader_ReadGlobalDefinitions (OTF2_Reader *reader, OTF2_GlobalDefReader *defReader, uint64_t definitionsToRead, uint64_t *definitionsRead)`
Read a given number of definitions via a global definition reader.

J.27 OTF2_Reader.h File Reference

- [OTF2_ErrorCode OTF2_Reader_ReadGlobalEvent](#) ([OTF2_Reader](#) *reader, [OTF2_GlobalEvtReader](#) *evtReader)

Read an event via a global event reader.

- [OTF2_ErrorCode OTF2_Reader_ReadGlobalEvents](#) ([OTF2_Reader](#) *reader, [OTF2_GlobalEvtReader](#) *evtReader, [uint64_t](#) eventsToRead, [uint64_t](#) *eventsRead)

Read a given number of events via a global event reader.

- [OTF2_ErrorCode OTF2_Reader_ReadGlobalSnapshots](#) ([OTF2_Reader](#) *reader, [OTF2_GlobalSnapReader](#) *snapReader, [uint64_t](#) recordsToRead, [uint64_t](#) *recordsRead)

Read a given number of records via a global snapshot reader.

- [OTF2_ErrorCode OTF2_Reader_ReadLocalDefinitions](#) ([OTF2_Reader](#) *reader, [OTF2_DefReader](#) *defReader, [uint64_t](#) definitionsToRead, [uint64_t](#) *definitionsRead)

Read a given number of definitions via a local definition reader.

- [OTF2_ErrorCode OTF2_Reader_ReadLocalEvents](#) ([OTF2_Reader](#) *reader, [OTF2_EvtReader](#) *evtReader, [uint64_t](#) eventsToRead, [uint64_t](#) *eventsRead)

Read a given number of events via a local event reader.

- [OTF2_ErrorCode OTF2_Reader_ReadLocalEventsBackward](#) ([OTF2_Reader](#) *reader, [OTF2_EvtReader](#) *evtReader, [uint64_t](#) eventsToRead, [uint64_t](#) *eventsRead)

Read a given number of events via a local event reader backwards.

- [OTF2_ErrorCode OTF2_Reader_ReadLocalSnapshots](#) ([OTF2_Reader](#) *reader, [OTF2_SnapReader](#) *snapReader, [uint64_t](#) recordsToRead, [uint64_t](#) *recordsRead)

Read a given number of records via a local snapshot reader.

- [OTF2_ErrorCode OTF2_Reader_ReadMarkers](#) ([OTF2_Reader](#) *reader, [OTF2_MarkerReader](#) *markerReader, [uint64_t](#) markersToRead, [uint64_t](#) *markersRead)

Read a given number of markers via a marker reader.

- [OTF2_ErrorCode OTF2_Reader_RegisterDefCallbacks](#) ([OTF2_Reader](#) *reader, [OTF2_DefReader](#) *defReader, const [OTF2_DefReaderCallbacks](#) *callbacks, void *userData)

Register local definition reader callbacks.

- [OTF2_ErrorCode OTF2_Reader_RegisterEvtCallbacks](#) ([OTF2_Reader](#) *reader, [OTF2_EvtReader](#) *evtReader, const [OTF2_EvtReaderCallbacks](#) *callbacks, void *userData)

Register event reader callbacks.

APPENDIX J. FILE DOCUMENTATION

- [OTF2_ErrorCode](#) [OTF2_Reader_RegisterGlobalDefCallbacks](#) ([OTF2_Reader](#) *reader, [OTF2_GlobalDefReader](#) *defReader, const [OTF2_GlobalDefReaderCallbacks](#) *callbacks, void *userData)
Register global definition reader callbacks.
- [OTF2_ErrorCode](#) [OTF2_Reader_RegisterGlobalEvtCallbacks](#) ([OTF2_Reader](#) *reader, [OTF2_GlobalEvtReader](#) *evtReader, const [OTF2_GlobalEvtReaderCallbacks](#) *callbacks, void *userData)
Register global event reader callbacks.
- [OTF2_ErrorCode](#) [OTF2_Reader_RegisterGlobalSnapCallbacks](#) ([OTF2_Reader](#) *reader, [OTF2_GlobalSnapReader](#) *evtReader, const [OTF2_GlobalSnapReaderCallbacks](#) *callbacks, void *userData)
Register global event reader callbacks.
- [OTF2_ErrorCode](#) [OTF2_Reader_RegisterMarkerCallbacks](#) ([OTF2_Reader](#) *reader, [OTF2_MarkerReader](#) *markerReader, const [OTF2_MarkerReaderCallbacks](#) *callbacks, void *userData)
Register marker reader callbacks.
- [OTF2_ErrorCode](#) [OTF2_Reader_RegisterSnapCallbacks](#) ([OTF2_Reader](#) *reader, [OTF2_SnapReader](#) *snapReader, const [OTF2_SnapReaderCallbacks](#) *callbacks, void *userData)
Register snapshot event reader callbacks.
- [OTF2_ErrorCode](#) [OTF2_Reader_SetFileSionCallbacks](#) ([OTF2_Reader](#) *reader, const [OTF2_FileSionCallbacks](#) *fileSionCallbacks, void *fileSionData)
Register SION callbacks to the reader.

J.27.1 Detailed Description

Reading interface for OTF2 archives.

Maintainer:

Michael Wagner <michael.wagner@zih.tu-dresden.de>

Authors

Dominic Eschweiler <d.eschweiler@fz-juelich.de>, Michael Wagner <michael.wagner@zih.tu-dresden.de>

J.27.2 Function Documentation

J.27.2.1 [OTF2_ErrorCode](#) [OTF2_Reader_Close](#) ([OTF2_Reader](#) * reader)

Close a reader handle.

J.27 OTF2_Reader.h File Reference

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.

J.27.2.2 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.

J.27.2.3 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.

J.27.2.4 OTF2_ErrorCode OTF2_Reader_CloseGlobalDefReader (OTF2_Reader * *reader*, OTF2_GlobalDefReader * *globalDefReader*)

Closes the global definition reader.

APPENDIX J. FILE DOCUMENTATION

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.

J.27.2.5 `OTF2_StatusCode 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.

Returns

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

J.27.2.6 `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

J.27 OTF2_Reader.h File Reference

J.27.2.7 **OTF2_ErrorCode** **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.

J.27.2.8 **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.

J.27.2.9 **OTF2_ErrorCode** **OTF2_Reader_CloseSnapReader** (**OTF2_Reader** *
reader, **OTF2_SnapReader** * *snapReader*)

Close a local snapshot reader.

Parameters

APPENDIX J. FILE DOCUMENTATION

<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

J.27.2.10 **OTF2_StatusCode** **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.

J.27.2.11 **OTF2_StatusCode** **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.

J.27 OTF2_Reader.h File Reference

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

J.27.2.12 **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.

J.27.2.13 **OTF2_ErrorCode** OTF2_Reader_GetCompression (**OTF2_Reader** *
reader, **OTF2_Compression** * *compression*)

Get copression mode.

Parameters

	<i>reader</i>	Reader handle.
out	<i>compres-sion</i>	Returned compression mode.

Returns

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

APPENDIX J. FILE DOCUMENTATION

J.27.2.14 **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.

J.27.2.15 **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.

J.27.2.16 **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.

J.27 OTF2_Reader.h File Reference

J.27.2.17 **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.

J.27.2.18 **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.

J.27.2.19 **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.

APPENDIX J. FILE DOCUMENTATION

J.27.2.20 **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.

J.27.2.21 **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

J.27.2.22 **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.

J.27 OTF2_Reader.h File Reference

J.27.2.23 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.

J.27.2.24 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.

J.27.2.25 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.

APPENDIX J. FILE DOCUMENTATION

Returns

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

J.27.2.26 **OTF2_StatusCode** **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.

J.27.2.27 **OTF2_StatusCode** **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

J.27.2.28 **OTF2_StatusCode** **OTF2_Reader_GetNumberOfThumbnails** (**OTF2_Reader**
* *reader*, **uint32_t** * *number*)

Get number of thumbs.

J.27 OTF2_Reader.h File Reference

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

J.27.2.29 **OTF2_ErrorCode** 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

J.27.2.30 **OTF2_ErrorCode** 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> .

APPENDIX J. FILE DOCUMENTATION

Returns

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

J.27.2.31 **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

J.27.2.32 **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.

J.27 OTF2_Reader.h File Reference

J.27.2.33 `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.

J.27.2.34 `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.

J.27.2.35 `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.

APPENDIX J. FILE DOCUMENTATION

Returns

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

J.27.2.36 **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.

J.27.2.37 **OTF2_ErrorCode 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.

J.27.2.38 **OTF2_ErrorCode OTF2_Reader_ReadAllGlobalEvents (OTF2_Reader * *reader*, OTF2_GlobalEvtReader * *evtReader*, uint64_t * *eventsRead*)**

Read all events via a global event reader.

Parameters

J.27 OTF2_Reader.h File Reference

	<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.

J.27.2.39 **OTF2_ErrorCode** **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

J.27.2.40 **OTF2_ErrorCode** **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.
	<i>defReader</i>	Local definition reader handle.
out	<i>definitionsRead</i>	Return pointer to the number of definitions actually read.

APPENDIX J. FILE DOCUMENTATION

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

J.27.2.41 **OTF2_ErrorCode** **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.

J.27.2.42 **OTF2_ErrorCode** **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

Returns

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

Since

Version 1.2

J.27 OTF2_Reader.h File Reference

J.27.2.43 `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.

J.27.2.44 `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>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, an error code if an error occurs.

J.27.2.45 `OTF2_Reader_ReadGlobalEvent (OTF2_Reader * reader, OTF2_GlobalEvtReader * evtReader)`

Read an event via a global event reader.

APPENDIX J. FILE DOCUMENTATION

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.

J.27.2.46 **OTF2_ErrorCode** **OTF2_Reader_ReadGlobalEvents** (**OTF2_Reader** * *reader*, **OTF2_GlobalEvtReader** * *evtReader*, **uint64_t** *eventsToRead*, **uint64_t** * *eventsRead*)

Read a given number of events via a global event reader.

Parameters

	<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.

J.27.2.47 **OTF2_ErrorCode** **OTF2_Reader_ReadGlobalSnapshots** (**OTF2_Reader** * *reader*, **OTF2_GlobalSnapReader** * *snapReader*, **uint64_t** *recordsToRead*, **uint64_t** * *recordsRead*)

Read a given number of records via a global snapshot reader.

Parameters

	<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.

J.27 OTF2_Reader.h File Reference

Returns

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

Since

Version 1.2

J.27.2.48 `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

J.27.2.49 `OTF2_ErrorCode OTF2_Reader_ReadLocalEvents (OTF2_Reader *
reader, OTF2_EvtReader * evtReader, uint64_t eventsToRead, uint64_t *
eventsRead)`

Read a given number of events via a local event reader.

Parameters

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Local event reader handle.

APPENDIX J. FILE DOCUMENTATION

<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.

J.27.2.50 *OTF2_ErrorCode* *OTF2_Reader_ReadLocalEventsBackward* (*OTF2_Reader* * *reader*, *OTF2_EvtReader* * *evtReader*, *uint64_t* *eventsToRead*, *uint64_t* * *eventsRead*)

Read a given number of events via a local event reader backwards.

Parameters

	<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.

J.27.2.51 *OTF2_ErrorCode* *OTF2_Reader_ReadLocalSnapshots* (*OTF2_Reader* * *reader*, *OTF2_SnapReader* * *snapReader*, *uint64_t* *recordsToRead*, *uint64_t* * *recordsRead*)

Read a given number of records via a local snapshot reader.

Parameters

	<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.

J.27 OTF2_Reader.h File Reference

Returns

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

Since

Version 1.2

J.27.2.52 `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>marker-Reader</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.

J.27.2.53 `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.

J.27.2.54 **OTF2_StatusCode** **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.

J.27.2.55 **OTF2_StatusCode** **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.
<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.

J.27 OTF2_Reader.h File Reference

J.27.2.56 **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.

J.27.2.57 **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.

J.27.2.58 **OTF2_ErrorCode** **OTF2_Reader_RegisterMarkerCallbacks** (
 OTF2_Reader * *reader*, **OTF2_MarkerReader** * *markerReader*, **const**
 OTF2_MarkerReaderCallbacks * *callbacks*, **void** * *userData*)

Register marker reader callbacks.

APPENDIX J. FILE DOCUMENTATION

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.

J.27.2.59 **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

Returns

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

J.27.2.60 **OTF2_ErrorCode** **OTF2_Reader_SetFileSionCallbacks** (**OTF2_Reader** * *reader*, **const** **OTF2_FileSionCallbacks** * *fileSionCallbacks*, **void** * *fileSionData*)

Register SION callbacks to the reader.

It suffice to provide a function for *OTF2_FileSionGetRank*. The neccessary information for the rank mapping can be extracted from the global group definition

J.28 OTF2_SnapReader.h File Reference

of type *OTF2_GROUP_TYPE_MPI_LOCATIONS* or by the *locationGroupId* attribute of the Location definitions.

Parameters

<i>reader</i>	Reader handle.
<i>fileSion-Callbacks</i>	Struct holding the callbacks.
<i>fileSion-Data</i>	Pointer passed to the callbacks by the caller.

Returns

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

J.28 OTF2_SnapReader.h File Reference

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

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_SnapReaderCallbacks.h>
```

Functions

- [OTF2_ErrorCode OTF2_SnapReader_GetLocationID](#) (const [OTF2_SnapReader](#) *reader, [OTF2_LocationRef](#) *location)
Return the location ID of the reading related location.
- [OTF2_ErrorCode OTF2_SnapReader_ReadSnapshots](#) ([OTF2_SnapReader](#) *reader, uint64_t recordsToRead, uint64_t *recordsRead)
After callback registration, the local events could be read with the following function. Readn reads recordsToRead records. The reader indicates that it reached the end of the trace by just reading less records than requested.
- [OTF2_ErrorCode OTF2_SnapReader_Seek](#) ([OTF2_SnapReader](#) *reader, uint64_t req_time, bool *found)
Seek jumps to start of latest snapshot that was made before a given time 'req_time'.

APPENDIX J. FILE DOCUMENTATION

- [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.

J.28.1 Detailed Description

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

J.28.2 Function Documentation

J.28.2.1 [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.

J.28.2.2 [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

	<i>reader</i>	Reader object which reads the events from its buffer.
	<i>recordsToRead</i>	How many records can be read next.

J.28 OTF2_SnapReader.h File Reference

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.

J.28.2.3 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_Error_Code with !=OTF2_SUCCESS if there was an error.

J.28.2.4 OTF2_ErrorCode OTF2_SnapReader.SetCallbacks (OTF2_SnapReader * *reader*, const OTF2_SnapReaderCallbacks * *callbacks*, void * *userData*)

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

These callbacks are ignored, if the events are read by an global event reader.

Parameters

<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.

J.29 OTF2_SnapReaderCallbacks.h File Reference

This defines the callbacks for the snap reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

Typedefs

- typedef [OTF2_CallbackCode](#)(* [OTF2_SnapReaderCallback_Enter](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) snapTime, void *userData, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) origEventTime, [OTF2_RegionRef](#) region)
Callback for the Enter snap event record.
- typedef [OTF2_CallbackCode](#)(* [OTF2_SnapReaderCallback_MeasurementOnOff](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) snapTime, void *userData, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) origEventTime, [OTF2_MeasurementMode](#) measurementMode)
Callback for the MeasurementOnOff snap event record.
- typedef [OTF2_CallbackCode](#)(* [OTF2_SnapReaderCallback_Metric](#))([OTF2_LocationRef](#) location, [OTF2_TimeStamp](#) snapTime, void *userData, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) origEventTime, [OTF2_MetricRef](#) metric, uint8_t numberOfMetrics, const [OTF2_Type](#) *typeIDs, const [OTF2_MetricValue](#) *metricValues)
Callback for the Metric snap event record.

J.29 OTF2_SnapReaderCallbacks.h File Reference

- 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 receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)

APPENDIX J. FILE DOCUMENTATION

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)`

Callback for the `SnapshotEnd` snap event record.

J.29 OTF2_SnapReaderCallbacks.h File Reference

- 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.
- OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_SnapReaderCallbacks *snapReaderCallbacks, OTF2_SnapReaderCallback_MpiCollectiveEnd mpiCollectiveEndCallback)

Registers the callback for the MpiCollectiveEnd snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetMpiIrecvCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_MpiIrecv](#) mpiIrecvCallback)

Registers the callback for the MpiIrecv snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetMpiIrecvRequestCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_MpiIrecvRequest](#) mpiIrecvRequestCallback)

Registers the callback for the MpiIrecvRequest snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetMpiIsendCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_MpiIsend](#) mpiIsendCallback)

Registers the callback for the MpiIsend snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_MpiIsendComplete](#) mpiIsendCompleteCallback)

Registers the callback for the MpiIsendComplete snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetMpiRecvCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_MpiRecv](#) mpiRecvCallback)

Registers the callback for the MpiRecv snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetMpiSendCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_MpiSend](#) mpiSendCallback)

Registers the callback for the MpiSend snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetOmpAcquireLockCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_OmpAcquireLock](#) ompAcquireLockCallback)

Registers the callback for the OmpAcquireLock snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetOmpForkCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_OmpFork](#) ompForkCallback)

Registers the callback for the OmpFork snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetOmpTaskCreateCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_OmpTaskCreate](#) ompTaskCreateCallback)

Registers the callback for the OmpTaskCreate snap event.

- [OTF2_ErrorCode](#) [OTF2_SnapReaderCallbacks_SetOmpTaskSwitchCallback](#) ([OTF2_SnapReaderCallbacks](#) *snapReaderCallbacks, [OTF2_SnapReaderCallback_OmpTaskSwitch](#) ompTaskSwitchCallback)

Registers the callback for the OmpTaskSwitch snap event.

J.29 OTF2_SnapReaderCallbacks.h File Reference

- [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.

J.29.1 Detailed Description

This defines the callbacks for the snap reader.

Source Template:

templates/OTF2_SnapReaderCallbacks.tmpl.h

J.29.2 Typedef Documentation

- J.29.2.1** `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.

APPENDIX J. FILE DOCUMENTATION

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 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>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](#).

J.29.2.2 `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

<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>measurementMode</i>	Is the measurement turned on (OTF2_MEASUREMENT_ON) or off (OTF2_MEASUREMENT_OFF)?

J.29 OTF2_SnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

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

J.29.2.3 `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 <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>metric</i>	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.
<i>numberOf-Metrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types.
<i>metricVal-ues</i>	List of metric values.

Since

Version 1.2

Returns

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

J.29.2.4 `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 <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.

Since

Version 1.2

Returns

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

J.29.2.5 `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 *MpiCollectiveBeginSnaps* record is still in the snapshot though.

J.29 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>origEventTime</i>	The original time this event happended.
<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 <i>communicator</i> .
<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](#).

J.29.2.6 `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.
-----------------	--

APPENDIX J. FILE DOCUMENTATION

<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 <i>communicator</i> .
<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](#).

J.29.2.7 `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_ -
MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp
snapTime, void *userData, OTF2_AttributeList *attributeList,
OTF2_TimeStamp origEventTime, uint64_t requestID)`

Callback for the MpiIrecvRequest snap event record.

This record exists for each [MpiIrecvRequest](#) event where an corresponding [Mpi-Irecv](#) or [MpiRequestCancelled](#) event did not occur on this location before the 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 OTF2_Reader_RegisterSnapCallbacks or OTF2_SnapReader_SetCallbacks .
<i>attributeList</i>	Additional attributes for this event.

J.29 OTF2_SnapReaderCallbacks.h File Reference

<i>origEvent-Time</i>	The original time this event happended.
<i>requestID</i>	ID of the requested receive

Since

Version 1.2

Returns

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

J.29.2.8 `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 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 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
<i>requestID</i>	ID of the related request

Since

Version 1.2

Returns

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

J.29.2.9 `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.
<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 related request

Since

Version 1.2

Returns

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

J.29 OTF2_SnapReaderCallbacks.h File Reference

J.29.2.10 `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 happended.
<i>sender</i>	MPI rank of sender in <i>communicator</i> .
<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](#).

APPENDIX J. FILE DOCUMENTATION

J.29.2.11 `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 *MpiIrecv* 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 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>receiver</i>	MPI rank of receiver in <i>communicator</i> .
<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](#).

J.29.2.12 `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.

J.29 OTF2_SnapReaderCallbacks.h File Reference

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 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>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](#).

J.29.2.13 `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.
<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.

APPENDIX J. FILE DOCUMENTATION

<i>num-berOfRe-quest-edThreads</i>	Requested size of the team.
------------------------------------	-----------------------------

Since

Version 1.2

Returns

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

J.29.2.14 `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 <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>taskID</i>	Identifier of the newly created task instance.

Since

Version 1.2

Returns

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

J.29 OTF2_SnapReaderCallbacks.h File Reference

J.29.2.15 `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 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 now active task instance.

Since

Version 1.2

Returns

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

J.29.2.16 `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_
ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp
snapTime, void *userData, OTF2_AttributeList *attributeList,
OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter,
int64_t value)`

Callback for the ParameterInt snap event record.

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

Parameters

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

APPENDIX J. FILE DOCUMENTATION

<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](#).

J.29.2.17 `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.

J.29 OTF2_SnapReaderCallbacks.h File Reference

<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](#).

J.29.2.18 `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](#).

APPENDIX J. FILE DOCUMENTATION

J.29.2.19 `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](#).

J.29.2.20 `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.

J.29 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>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](#).

J.29.2.21 `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](#).

J.29.2.22 `typedef struct OTF2_SnapReaderCallbacks_struct OTF2_SnapReaderCallbacks`

Opaque struct which holds all snap event record callbacks.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.2

J.29.3 Function Documentation

J.29.3.1 `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

J.29.3.2 `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

J.29.3.3 `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](#).

J.29 OTF2_SnapReaderCallbacks.h File Reference

J.29.3.4 **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

J.29.3.5 **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

APPENDIX J. FILE DOCUMENTATION

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.29.3.6 **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

J.29.3.7 **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.

J.29 OTF2_SnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.29.3.8 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

J.29.3.9 OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiIrecvCallback
(OTF2_SnapReaderCallbacks * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_MpiIrecv *mpilrecvCallback*)

Registers the callback for the MpiIrecv snap event.

Parameters

APPENDIX J. FILE DOCUMENTATION

<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

J.29.3.10 **OTF2_ErrorCode** **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

J.29 OTF2_SnapReaderCallbacks.h File Reference

J.29.3.11 **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

J.29.3.12 **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

Returns

OTF2_SUCCESS if successful

APPENDIX J. FILE DOCUMENTATION

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.29.3.13 **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

J.29.3.14 **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

J.29 OTF2_SnapReaderCallbacks.h File Reference

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.29.3.15 **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

J.29.3.16 **OTF2_ErrorCode** **OTF2_SnapReaderCallbacks_SetOmpForkCallback**
(**OTF2_SnapReaderCallbacks** * *snapReaderCallbacks*,
OTF2_SnapReaderCallback_OmpFork *ompForkCallback*)

Registers the callback for the OmpFork snap event.

Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompForkCallback</i>	Function which should be called for all OmpFork events.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

```
J.29.3.17  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

```
J.29.3.18  OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpTaskSwitchCallback
            ( OTF2_SnapReaderCallbacks * snapReaderCallbacks,
              OTF2_SnapReaderCallback_OmpTaskSwitch ompTaskSwitchCallback
            )
```

Registers the callback for the OmpTaskSwitch snap event.

Parameters

J.29 OTF2_SnapReaderCallbacks.h File Reference

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskSwitch-Callback</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

J.29.3.19 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>parameter-IntCallback</i>	Function which should be called for all ParameterInt events.

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

APPENDIX J. FILE DOCUMENTATION

J.29.3.20 **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

J.29.3.21 **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.

J.29 OTF2_SnapReaderCallbacks.h File Reference

Since

Version 1.2

Returns

OTF2_SUCCESS if successful

OTF2_ERROR_INVALID_ARGUMENT for an invalid `defReaderCallbacks` argument

J.29.3.22 **OTF2_ErrorCode** **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

J.29.3.23 **OTF2_ErrorCode** **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.
----------------------------	---------------------------

APPENDIX J. FILE DOCUMENTATION

<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

J.29.3.24 OTF2_StatusCode 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

J.30 OTF2_SnapWriter.h File Reference

This lowest user-visible layer provides write routines to write snapshot records for a single location.

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

J.30 OTF2_SnapWriter.h File Reference

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_AttributeList.h>
```

Typedefs

- typedef struct OTF2_SnapWriter_struct [OTF2_SnapWriter](#)
Keeps all necessary information about the snap writer. See OTF2_SnapWriter_-struct for detailed information.

Functions

- [OTF2_ErrorCode OTF2_SnapWriter_Enter](#) ([OTF2_SnapWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) snapTime, [OTF2_TimeStamp](#) origEventTime, [OTF2_RegionRef](#) region)
Records an Enter snapshot record.
- [OTF2_ErrorCode OTF2_SnapWriter_GetLocationID](#) (const [OTF2_SnapWriter](#) *writer, [OTF2_LocationRef](#) *locationID)
Function to get the location ID of a snap writer object.
- [OTF2_ErrorCode OTF2_SnapWriter_MeasurementOnOff](#) ([OTF2_SnapWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) snapTime, [OTF2_TimeStamp](#) origEventTime, [OTF2_MeasurementMode](#) measurementMode)
Records an MeasurementOnOff snapshot record.
- [OTF2_ErrorCode OTF2_SnapWriter_Metric](#) ([OTF2_SnapWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) snapTime, [OTF2_TimeStamp](#) origEventTime, [OTF2_MetricRef](#) metric, [uint8_t](#) numberOfMetrics, const [OTF2_Type](#) *typeIDs, const [OTF2_MetricValue](#) *metricValues)
Records an Metric snapshot record.
- [OTF2_ErrorCode OTF2_SnapWriter_MpiCollectiveBegin](#) ([OTF2_SnapWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) snapTime, [OTF2_TimeStamp](#) origEventTime)
Records an MpiCollectiveBegin snapshot record.
- [OTF2_ErrorCode OTF2_SnapWriter_MpiCollectiveEnd](#) ([OTF2_SnapWriter](#) *writer, [OTF2_AttributeList](#) *attributeList, [OTF2_TimeStamp](#) snapTime, [OTF2_TimeStamp](#) origEventTime, [OTF2_CollectiveOp](#) collectiveOp, [OTF2_CommRef](#) communicator, [uint32_t](#) root, [uint64_t](#) sizeSent, [uint64_t](#) sizeReceived)
Records an MpiCollectiveEnd snapshot record.

APPENDIX J. FILE DOCUMENTATION

- [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.

J.30 OTF2_SnapWriter.h File Reference

- [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.

J.30.1 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

J.30.2 Typedef Documentation

J.30.2.1 typedef struct OTF2_SnapWriter struct OTF2_SnapWriter

Keeps all necessary information about the snap writer. See [OTF2_SnapWriter_struct](#) for detailed information.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.2

J.30.3 Function Documentation

J.30.3.1 `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>origEvent-Time</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.

J.30.3.2 `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

J.30 OTF2_SnapWriter.h File Reference

Since

Version 1.2

Returns

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

J.30.3.3 `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>origEvent-Time</i>	The original time this event happened.
<i>measure-mentMode</i>	Is the measurement turned on (<i>OTF2_MEASUREMENT_ON</i>) or off (<i>OTF2_MEASUREMENT_OFF</i>)?

Since

Version 1.2

Returns

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

J.30.3.4 `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 * typeIdDs, 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.

APPENDIX J. FILE DOCUMENTATION

As an exception for metric classes where the metric mode denotes 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 happened.
<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.
<i>metricValues</i>	List of metric values.

Since

Version 1.2

Returns

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

J.30.3.5 OTF2_StatusCode 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 happened.

J.30 OTF2_SnapWriter.h File Reference

Since

Version 1.2

Returns

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

J.30.3.6 `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 *MpiCollectiveBeginSnaps* 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 <i>communicator</i> .
<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.

APPENDIX J. FILE DOCUMENTATION

J.30.3.7 `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 <i>communicator</i> .
<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.

J.30.3.8 `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 [Mpi-Irecv](#) or [MpiRequestCancelled](#) event did not occur on this location before the snapshot. Or the corresponding [MpiIrecv](#) did occurred (the [MpiIrecvSnap](#) record exists

J.30 OTF2_SnapWriter.h File Reference

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.

J.30.3.9 `OTF2_ErrorCode 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 <i>communicator</i> .
<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.

APPENDIX J. FILE DOCUMENTATION

<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.

J.30.3.10 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.

J.30 OTF2_SnapWriter.h File Reference

J.30.3.11 `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 <i>communicator</i> .
<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.

J.30.3.12 `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

APPENDIX J. FILE DOCUMENTATION

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 <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_SUCCESS if successful, an error code if an error occurs.

J.30.3.13 *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.

J.30 OTF2_SnapWriter.h File Reference

<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.

J.30.3.14 `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.

APPENDIX J. FILE DOCUMENTATION

J.30.3.15 **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.

J.30.3.16 **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.

J.30 OTF2_SnapWriter.h File Reference

Since

Version 1.2

Returns

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

J.30.3.17 `OTF2_StatusCode 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>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_SUCCESS](#) if successful, an error code if an error occurs.

J.30.3.18 `OTF2_StatusCode 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.

APPENDIX J. FILE DOCUMENTATION

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.

J.30.3.19 `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.

J.30 OTF2_SnapWriter.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 1.2

Returns

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

J.30.3.20 `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.

J.30.3.21 `OTF2_ErrorCode OTF2_SnapWriter.SnapshotStart (OTF2_SnapWriter
* writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp
snapTime, uint64_t numberOfRecords)`

Records an SnapshotStart snapshot record.

APPENDIX J. FILE DOCUMENTATION

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>num- berOfRecord</i>	Number of snapshot event records in this snapshot. Excluding the <i>SnapshotEnd</i> record.

Since

Version 1.2

Returns

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

J.31 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.

J.31 OTF2_Thumbnail.h File Reference

Functions

- [OTF2_ErrorCode OTF2_ThumbReader_GetHeader](#) ([OTF2_ThumbReader](#) *reader, char **const name, char **const description, [OTF2_ThumbnailType](#) *type, uint32_t *numberOfSamples, uint32_t *numberOfMetrics, uint64_t **refsToDefs)

Reads a thumbnail header.

- [OTF2_ErrorCode OTF2_ThumbReader_ReadSample](#) ([OTF2_ThumbReader](#) *reader, uint64_t *baseline, uint32_t numberOfMetrics, uint64_t *metricSamples)

Reads a thumbnail sample.

- [OTF2_ErrorCode OTF2_ThumbWriter_WriteSample](#) ([OTF2_ThumbWriter](#) *writer, uint64_t baseline, uint32_t numberOfMetrics, const uint64_t *metricSamples)

Writes a thumbnail sample.

J.31.1 Detailed Description

This lowest user-visible layer provides write routines to read and write thumbnail data.

J.31.2 Function Documentation

J.31.2.1 [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.
<i>name</i>	Name of thumbnail.
<i>description</i>	Description of thumbnail.
<i>type</i>	Type of thumbnail.
<i>numberOfSamples</i>	Number of samples.
<i>numberOfMetrics</i>	Number of metrics.

APPENDIX J. FILE DOCUMENTATION

Since

Version 1.2

Returns

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

J.31.2.2 **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.
<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.
<i>metricSamples</i>	Metric sample values.

Since

Version 1.2

Returns

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

J.31.2.3 **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.

J.31 OTF2_Thumbnail.h File Reference

<i>metricSamples</i>	Metric sample values.
----------------------	-----------------------

Since

Version 1.2

Returns

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

Index

otf2.h, [95](#)
OTF2_ABORT
 OTF2_ErrorCodes.h, [221](#)
OTF2_Archive.h
 OTF2_MASTER, [102](#)
 OTF2_SLAVE, [102](#)
OTF2_BASE_BINARY
 OTF2_Definitions.h, [163](#)
OTF2_BASE_DECIMAL
 OTF2_Definitions.h, [163](#)
OTF2_CALLBACK_INTERRUPT
 OTF2_GeneralDefinitions.h, [363](#)
OTF2_CALLBACK_SUCCESS
 OTF2_GeneralDefinitions.h, [363](#)
OTF2_COLLECTIVE_OP_ALLOCATE
 OTF2_Events.h, [228](#)
OTF2_COLLECTIVE_OP_CREATE_HANDLE
 OTF2_Events.h, [228](#)
OTF2_COLLECTIVE_OP_CREATE_HANDLE_
 AND_ALLOCATE
 OTF2_Events.h, [228](#)
OTF2_COLLECTIVE_OP_DEALLOCATE
 OTF2_Events.h, [228](#)
OTF2_COLLECTIVE_OP_DESTROY_
 HANDLE
 OTF2_Events.h, [228](#)
OTF2_COLLECTIVE_OP_DESTROY_
 HANDLE_AND_DEALLOCATE
 OTF2_Events.h, [228](#)
OTF2_COMPRESSION_NONE
 OTF2_GeneralDefinitions.h, [364](#)
OTF2_COMPRESSION_UNDEFINED
 OTF2_GeneralDefinitions.h, [364](#)
OTF2_COMPRESSION_ZLIB
 OTF2_GeneralDefinitions.h, [364](#)
OTF2_Definitions.h
OTF2_BASE_BINARY, [163](#)
OTF2_BASE_DECIMAL, [163](#)
OTF2_GROUP_FLAG_GLOBAL_
 MEMBERS, [161](#)
OTF2_GROUP_FLAG_NONE, [161](#)
OTF2_GROUP_TYPE_COMM_GROUP,
 [162](#)
OTF2_GROUP_TYPE_COMM_LOCATIONS,
 [162](#)
OTF2_GROUP_TYPE_COMM_SELF,
 [162](#)
OTF2_GROUP_TYPE_LOCATIONS,
 [161](#)
OTF2_GROUP_TYPE_METRIC, [161](#)
OTF2_GROUP_TYPE_REGIONS,
 [161](#)
OTF2_GROUP_TYPE_UNKNOWN,
 [161](#)
OTF2_LOCATION_GROUP_TYPE_
 PROCESS, [162](#)
OTF2_LOCATION_GROUP_TYPE_
 UNKNOWN, [162](#)
OTF2_LOCATION_TYPE_CPU_THREAD,
 [162](#)
OTF2_LOCATION_TYPE_GPU, [163](#)
OTF2_LOCATION_TYPE_METRIC,
 [163](#)
OTF2_LOCATION_TYPE_UNKNOWN,
 [162](#)
OTF2_METRIC_ABSOLUTE_LAST,
 [163](#)
OTF2_METRIC_ABSOLUTE_NEXT,
 [163](#)
OTF2_METRIC_ABSOLUTE_POINT,
 [163](#)

INDEX

OTF2_METRIC_ACCUMULATED_- LAST, 163	OTF2_PARAMETER_TYPE_UINT64, 166
OTF2_METRIC_ACCUMULATED_- NEXT, 163	OTF2_RECORDER_KIND_ABSTRACT, 166
OTF2_METRIC_ACCUMULATED_- POINT, 163	OTF2_RECORDER_KIND_CPU, 166
OTF2_METRIC_ACCUMULATED_- START, 163	OTF2_RECORDER_KIND_GPU, 166
OTF2_METRIC_ASYNCHRONOUS, 164	OTF2_RECORDER_KIND_UNKNOWN, 166
OTF2_METRIC_RELATIVE_LAST, 163	OTF2_REGION_FLAG_DYNAMIC, 167
OTF2_METRIC_RELATIVE_NEXT, 163	OTF2_REGION_FLAG_NONE, 167
OTF2_METRIC_RELATIVE_POINT, 163	OTF2_REGION_FLAG_PHASE, 167
OTF2_METRIC_SYNCHRONOUS, 164	OTF2_REGION_ROLE_ARTIFICIAL, 168
OTF2_METRIC_SYNCHRONOUS_- STRICT, 164	OTF2_REGION_ROLE_ATOMIC, 168
OTF2_METRIC_TIMING_LAST, 165	OTF2_REGION_ROLE_BARRIER, 168
OTF2_METRIC_TIMING_MASK, 165	OTF2_REGION_ROLE_CODE, 167
OTF2_METRIC_TIMING_NEXT, 165	OTF2_REGION_ROLE_COLL_ALL2ALL, 168
OTF2_METRIC_TIMING_POINT, 164	OTF2_REGION_ROLE_COLL_ALL2ONE, 168
OTF2_METRIC_TIMING_START, 164	OTF2_REGION_ROLE_COLL_ONE2ALL, 168
OTF2_METRIC_TYPE_OTHER, 165	OTF2_REGION_ROLE_COLL_OTHER, 168
OTF2_METRIC_TYPE_PAPI, 165	OTF2_REGION_ROLE_CRITICAL, 168
OTF2_METRIC_TYPE_RUSAGE, 165	OTF2_REGION_ROLE_CRITICAL_- SBLOCK, 168
OTF2_METRIC_TYPE_USER, 165	OTF2_REGION_ROLE_DATA_TRANSFER, 168
OTF2_METRIC_VALUE_ABSOLUTE, 165	OTF2_REGION_ROLE_FILE_IO, 168
OTF2_METRIC_VALUE_ACCUMULATED, 165	OTF2_REGION_ROLE_FLUSH, 168
OTF2_METRIC_VALUE_MASK, 166	OTF2_REGION_ROLE_FUNCTION, 167
OTF2_METRIC_VALUE_RELATIVE, 165	OTF2_REGION_ROLE_IMPLICIT_- BARRIER, 168
OTF2_PARAMETER_TYPE_INT64, 166	OTF2_REGION_ROLE_LOOP, 167
OTF2_PARAMETER_TYPE_STRING, 166	OTF2_REGION_ROLE_MASTER, 168
	OTF2_REGION_ROLE_ORDERED, 168

OTF2_REGION_ROLE_ORDERED_- SBLOCK, 168
 OTF2_REGION_ROLE_PARALLEL, 167
 OTF2_REGION_ROLE_POINT2POINT, 168
 OTF2_REGION_ROLE_RMA, 168
 OTF2_REGION_ROLE_SECTION, 167
 OTF2_REGION_ROLE_SECTIONS, 167
 OTF2_REGION_ROLE_SINGLE, 167
 OTF2_REGION_ROLE_SINGLE_- SBLOCK, 167
 OTF2_REGION_ROLE_TASK, 168
 OTF2_REGION_ROLE_TASK_CREATE, 168
 OTF2_REGION_ROLE_TASK_WAIT, 168
 OTF2_REGION_ROLE_UNKNOWN, 167
 OTF2_REGION_ROLE_WORKSHARE, 167
 OTF2_REGION_ROLE_WRAPPER, 167
 OTF2_SCOPE_GROUP, 164
 OTF2_SCOPE_LOCATION, 164
 OTF2_SCOPE_LOCATION_GROUP, 164
 OTF2_SCOPE_SYSTEM_TREE_- NODE, 164
 OTF2_SYSTEM_TREE_DOMAIN_- CACHE, 169
 OTF2_SYSTEM_TREE_DOMAIN_- CORE, 169
 OTF2_SYSTEM_TREE_DOMAIN_- MACHINE, 169
 OTF2_SYSTEM_TREE_DOMAIN_- NUMA, 169
 OTF2_SYSTEM_TREE_DOMAIN_- PU, 169
 OTF2_SYSTEM_TREE_DOMAIN_- SHARED_MEMORY, 169
 OTF2_SYSTEM_TREE_DOMAIN_- SOCKET, 169
 OTF2_DEPRECATED, 167
 OTF2_ERROR_CODES.h, 221
 OTF2_ERROR_DUPLICATE_MAPPING_- TABLE, 224
 OTF2_ERROR_E2BIG, 221
 OTF2_ERROR_EACCES, 221
 OTF2_ERROR_EADDRNOTAVAIL, 221
 OTF2_ERROR_EAFNOSUPPORT, 221
 OTF2_ERROR_EAGAIN, 222
 OTF2_ERROR_EALREADY, 222
 OTF2_ERROR_EBADF, 222
 OTF2_ERROR_EBADMSG, 222
 OTF2_ERROR_EBUSY, 222
 OTF2_ERROR_ECANCELED, 222
 OTF2_ERROR_ECHILD, 222
 OTF2_ERROR_ECONNREFUSED, 222
 OTF2_ERROR_ECONNRESET, 222
 OTF2_ERROR_EDEADLK, 222
 OTF2_ERROR_EDESTADDRREQ, 222
 OTF2_ERROR_EDOM, 222
 OTF2_ERROR_EDQUOT, 222
 OTF2_ERROR_EEXIST, 222
 OTF2_ERROR_EFAULT, 222

INDEX

OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EFBIG
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EINPROGRESS
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EINTR
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EINVAL
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EIO
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EISCONN
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EISDIR
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_ELOOP
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EMFILE
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EMLINK
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_MSGSIZE
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_EMULTIHOP
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_ENAMETOOLONG
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_END_OF_BUFFER
OTF2_ErrorCodes.h, [224](#)
OTF2_ERROR_END_OF_FUNCTION
OTF2_ErrorCodes.h, [224](#)
OTF2_ERROR_ENETDOWN
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_ENETRESET
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_ENETUNREACH
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_ENFILE
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_ENOBUFS
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_ENODATA
OTF2_ErrorCodes.h, [222](#)
OTF2_ERROR_ENODEV
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOENT
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOEXEC
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOLCK
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOLINK
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOMEM
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOMSG
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOPROTOOPT
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOSPC
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOSR
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOSTR
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOSYS
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOTCONN
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOTDIR
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOTEMPTY
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOTSOCK
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOTSUP
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENOTTY
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_ENXIO
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_EOPNOTSUPP
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_EOVERFLOW
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_EPERM
OTF2_ErrorCodes.h, [223](#)
OTF2_ERROR_EPIPE

OTF2_ErrorCodes.h, 223	OTF2_ErrorCodes.h, 224
OTF2_ERROR_EPROTO	OTF2_ERROR_INVALID_CALL
OTF2_ErrorCodes.h, 223	OTF2_ErrorCodes.h, 224
OTF2_ERROR_EPROTONOSUPPORT	OTF2_ERROR_INVALID_DATA
OTF2_ErrorCodes.h, 223	OTF2_ErrorCodes.h, 224
OTF2_ERROR_EPROTOTYPE	OTF2_ERROR_INVALID_FILE_MODE_-
OTF2_ErrorCodes.h, 223	TRANSITION
OTF2_ERROR_ERANGE	OTF2_ErrorCodes.h, 224
OTF2_ErrorCodes.h, 223	OTF2_ERROR_INVALID_LINENO
OTF2_ERROR_EROFS	OTF2_ErrorCodes.h, 224
OTF2_ErrorCodes.h, 223	OTF2_ERROR_INVALID_RECORD
OTF2_ERROR_ESPIPE	OTF2_ErrorCodes.h, 224
OTF2_ErrorCodes.h, 223	OTF2_ERROR_INVALID_SIZE_GIVEN
OTF2_ERROR_ESRCH	OTF2_ErrorCodes.h, 224
OTF2_ErrorCodes.h, 223	OTF2_ERROR_MEM_ALLOC_FAILED
OTF2_ERROR_ESTALE	OTF2_ErrorCodes.h, 224
OTF2_ErrorCodes.h, 223	OTF2_ERROR_MEM_FAULT
OTF2_ERROR_ETIME	OTF2_ErrorCodes.h, 224
OTF2_ErrorCodes.h, 223	OTF2_ERROR_PROCESSED_WITH_-
OTF2_ERROR_ETIMEDOUT	FAULTS
OTF2_ErrorCodes.h, 223	OTF2_ErrorCodes.h, 224
OTF2_ERROR_ETXTBSY	OTF2_ERROR_PROPERTY_EXISTS
OTF2_ErrorCodes.h, 223	OTF2_ErrorCodes.h, 224
OTF2_ERROR_EWOULDBLOCK	OTF2_ERROR_PROPERTY_NAME_INVALID
OTF2_ErrorCodes.h, 224	OTF2_ErrorCodes.h, 224
OTF2_ERROR_EXDEV	OTF2_ERROR_PROPERTY_NOT_FOUND
OTF2_ErrorCodes.h, 224	OTF2_ErrorCodes.h, 224
OTF2_ERROR_FILE_CAN_NOT_OPEN	OTF2_ERROR_PROPERTY_VALUE_-
OTF2_ErrorCodes.h, 224	INVALID
OTF2_ERROR_FILE_COMPRESSION_-	OTF2_ErrorCodes.h, 224
NOT_SUPPORTED	OTF2_ERROR_UNKNOWN_TYPE
OTF2_ErrorCodes.h, 224	OTF2_ErrorCodes.h, 224
OTF2_ERROR_FILE_INTERACTION	OTF2_ErrorCodes.h
OTF2_ErrorCodes.h, 224	OTF2_ABORT, 221
OTF2_ERROR_INDEX_OUT_OF_BOUNDS	OTF2_DEPRECATED, 221
OTF2_ErrorCodes.h, 224	OTF2_ERROR_DUPLICATE_MAPPING_-
OTF2_ERROR_INTEGRITY_FAULT	TABLE, 224
OTF2_ErrorCodes.h, 224	OTF2_ERROR_E2BIG, 221
OTF2_ERROR_INTERRUPTED_BY_-	OTF2_ERROR_EACCES, 221
CALLBACK	OTF2_ERROR_EADDRNOTAVAIL,
OTF2_ErrorCodes.h, 224	221
OTF2_ERROR_INVALID	OTF2_ERROR_EAFNOSUPPORT,
OTF2_ErrorCodes.h, 221	221
OTF2_ERROR_INVALID_ARGUMENT	OTF2_ERROR_EAGAIN, 222

INDEX

OTF2_ERROR_EALREADY, [222](#)
OTF2_ERROR_EBADF, [222](#)
OTF2_ERROR_EBADMSG, [222](#)
OTF2_ERROR_EBUSY, [222](#)
OTF2_ERROR_ECANCELED, [222](#)
OTF2_ERROR_ECHILD, [222](#)
OTF2_ERROR_ECONNREFUSED, [222](#)
OTF2_ERROR_ECONNRESET, [222](#)
OTF2_ERROR_EDEADLK, [222](#)
OTF2_ERROR_EDESTADDRREQ, [222](#)
OTF2_ERROR_EDOM, [222](#)
OTF2_ERROR_EDQUOT, [222](#)
OTF2_ERROR_EEXIST, [222](#)
OTF2_ERROR_EFAULT, [222](#)
OTF2_ERROR_EFBIG, [222](#)
OTF2_ERROR_EINPROGRESS, [222](#)
OTF2_ERROR_EINTR, [222](#)
OTF2_ERROR_EINVAL, [222](#)
OTF2_ERROR_EIO, [222](#)
OTF2_ERROR_EISCONN, [222](#)
OTF2_ERROR_EISDIR, [222](#)
OTF2_ERROR_ELOOP, [222](#)
OTF2_ERROR_EMFILE, [222](#)
OTF2_ERROR_EMLINK, [222](#)
OTF2_ERROR EMSGSIZE, [222](#)
OTF2_ERROR_EMULTIHOP, [222](#)
OTF2_ERROR_ENAMETOOLONG, [222](#)
OTF2_ERROR_END_OF_BUFFER, [224](#)
OTF2_ERROR_END_OF_FUNCTION, [224](#)
OTF2_ERROR_ENETDOWN, [222](#)
OTF2_ERROR_ENETRESET, [222](#)
OTF2_ERROR_ENETUNREACH, [222](#)
OTF2_ERROR_ENFILE, [222](#)
OTF2_ERROR_ENOBUFS, [222](#)
OTF2_ERROR_ENODATA, [222](#)
OTF2_ERROR_ENODEV, [223](#)
OTF2_ERROR_ENOENT, [223](#)
OTF2_ERROR_ENOEXEC, [223](#)
OTF2_ERROR_ENOLCK, [223](#)
OTF2_ERROR_ENOLINK, [223](#)
OTF2_ERROR_ENOMEM, [223](#)
OTF2_ERROR_ENOMSG, [223](#)
OTF2_ERROR_ENOPROTOOPT, [223](#)
OTF2_ERROR_ENOSPC, [223](#)
OTF2_ERROR_ENOSR, [223](#)
OTF2_ERROR_ENOSTR, [223](#)
OTF2_ERROR_ENOSYS, [223](#)
OTF2_ERROR_ENOTCONN, [223](#)
OTF2_ERROR_ENOTDIR, [223](#)
OTF2_ERROR_ENOTEMPTY, [223](#)
OTF2_ERROR_ENOTSOCK, [223](#)
OTF2_ERROR_ENOTSUP, [223](#)
OTF2_ERROR_ENOTTY, [223](#)
OTF2_ERROR_ENXIO, [223](#)
OTF2_ERROR_EOPNOTSUPP, [223](#)
OTF2_ERROR_EOVERFLOW, [223](#)
OTF2_ERROR_EPERM, [223](#)
OTF2_ERROR_EPIPE, [223](#)
OTF2_ERROR_EPROTO, [223](#)
OTF2_ERROR_EPROTONOSUPPORT, [223](#)
OTF2_ERROR_EPROTOTYPE, [223](#)
OTF2_ERROR_ERANGE, [223](#)
OTF2_ERROR_EROFS, [223](#)
OTF2_ERROR_ESPIPE, [223](#)
OTF2_ERROR_ESRCH, [223](#)
OTF2_ERROR_ESTALE, [223](#)
OTF2_ERROR_ETIME, [223](#)
OTF2_ERROR_ETIMEDOUT, [223](#)
OTF2_ERROR_ETXTBSY, [223](#)
OTF2_ERROR_EWOULDBLOCK, [224](#)
OTF2_ERROR_EXDEV, [224](#)
OTF2_ERROR_FILE_CAN_NOT_OPEN, [224](#)
OTF2_ERROR_FILE_COMPRESSION_NOT_SUPPORTED, [224](#)
OTF2_ERROR_FILE_INTERACTION, [224](#)
OTF2_ERROR_INDEX_OUT_OF_BOUNDS, [224](#)

644

INDEX

- OTF2_CALLBACK_INTERRUPT, [363](#)
- OTF2_CALLBACK_SUCCESS, [363](#)
- OTF2_COMPRESSION_NONE, [364](#)
- OTF2_COMPRESSION_UNDEFINED, [364](#)
- OTF2_COMPRESSION_ZLIB, [364](#)
- OTF2_FILEMODE_MODIFY, [364](#)
- OTF2_FILEMODE_READ, [364](#)
- OTF2_FILEMODE_WRITE, [364](#)
- OTF2_FILETYPE_ANCHOR, [365](#)
- OTF2_FILETYPE_EVENTS, [365](#)
- OTF2_FILETYPE_GLOBAL_DEFS, [365](#)
- OTF2_FILETYPE_LOCAL_DEFS, [365](#)
- OTF2_FILETYPE_MARKER, [365](#)
- OTF2_FILETYPE_SNAPSHOTS, [365](#)
- OTF2_FILETYPE_THUMBNAIL, [365](#)
- OTF2_FLUSH, [365](#)
- OTF2_MAPPING_ATTRIBUTE, [365](#)
- OTF2_MAPPING_COMM, [366](#)
- OTF2_MAPPING_GROUP, [365](#)
- OTF2_MAPPING_LOCATION, [365](#)
- OTF2_MAPPING_MAX, [366](#)
- OTF2_MAPPING_METRIC, [365](#)
- OTF2_MAPPING_PARAMETER, [366](#)
- OTF2_MAPPING_REGION, [365](#)
- OTF2_MAPPING_RMA_WIN, [366](#)
- OTF2_MAPPING_STRING, [365](#)
- OTF2_NO_FLUSH, [365](#)
- OTF2_PARADIGM_COMPILER, [366](#)
- OTF2_PARADIGM_CUDA, [366](#)
- OTF2_PARADIGM_MEASUREMENT_SYSTEM, [366](#)
- OTF2_PARADIGM_MPI, [366](#)
- OTF2_PARADIGM_OPENMP, [366](#)
- OTF2_PARADIGM_UNKNOWN, [366](#)
- OTF2_PARADIGM_USER, [366](#)
- OTF2_SUBSTRATE_NONE, [364](#)
- OTF2_SUBSTRATE_POSIX, [364](#)
- OTF2_SUBSTRATE_SION, [364](#)
- OTF2_SUBSTRATE_UNDEFINED, [364](#)
- OTF2_THUMBNAIL_TYPE_ATTRIBUTES, [367](#)
- OTF2_THUMBNAIL_TYPE_METRIC, [367](#)
- OTF2_THUMBNAIL_TYPE_REGION, [367](#)
- OTF2_TYPE_DOUBLE, [367](#)
- OTF2_TYPE_FLOAT, [367](#)
- OTF2_TYPE_INT16, [367](#)
- OTF2_TYPE_INT32, [367](#)
- OTF2_TYPE_INT64, [367](#)
- OTF2_TYPE_INT8, [367](#)
- OTF2_TYPE_NONE, [367](#)
- OTF2_TYPE_UINT16, [367](#)
- OTF2_TYPE_UINT32, [367](#)
- OTF2_TYPE_UINT64, [367](#)
- OTF2_TYPE_UINT8, [367](#)
- OTF2_GROUP_FLAG_GLOBAL_MEMBERS
OTF2_Definitions.h, [161](#)
- OTF2_GROUP_FLAG_NONE
OTF2_Definitions.h, [161](#)
- OTF2_GROUP_TYPE_COMM_GROUP
OTF2_Definitions.h, [162](#)
- OTF2_GROUP_TYPE_COMM_LOCATIONS
OTF2_Definitions.h, [162](#)
- OTF2_GROUP_TYPE_COMM_SELF
OTF2_Definitions.h, [162](#)
- OTF2_GROUP_TYPE_LOCATIONS
OTF2_Definitions.h, [161](#)
- OTF2_GROUP_TYPE_METRIC
OTF2_Definitions.h, [161](#)
- OTF2_GROUP_TYPE_REGIONS
OTF2_Definitions.h, [161](#)
- OTF2_GROUP_TYPE_UNKNOWN
OTF2_Definitions.h, [161](#)
- OTF2_ID_MAP_DENSE
OTF2_IdMap.h, [533](#)
- OTF2_ID_MAP_SPARSE
OTF2_IdMap.h, [533](#)
- OTF2_IdMap.h
- OTF2_ID_MAP_DENSE, [533](#)
- OTF2_ID_MAP_SPARSE, [533](#)

OTF2_LOCATION_GROUP_TYPE_PROCESSOR, 538
 OTF2_Definitions.h, 162
 OTF2_LOCATION_GROUP_TYPE_UNKNOWN, 538
 OTF2_Definitions.h, 162
 OTF2_LOCATION_TYPE_CPU_THREAD, 538
 OTF2_Definitions.h, 162
 OTF2_LOCATION_TYPE_GPU, 539
 OTF2_Definitions.h, 163
 OTF2_LOCATION_TYPE_METRIC, 539
 OTF2_Definitions.h, 163
 OTF2_LOCATION_TYPE_UNKNOWN, 538
 OTF2_Definitions.h, 162
 OTF2_LOCK_EXCLUSIVE, 228
 OTF2_Events.h, 228
 OTF2_LOCK_SHARED, 228
 OTF2_Events.h, 228
 OTF2_MAPPING_ATTRIBUTE, 365
 OTF2_GeneralDefinitions.h, 365
 OTF2_MAPPING_COMM, 366
 OTF2_GeneralDefinitions.h, 366
 OTF2_MAPPING_GROUP, 365
 OTF2_GeneralDefinitions.h, 365
 OTF2_MAPPING_LOCATION, 365
 OTF2_GeneralDefinitions.h, 365
 OTF2_MAPPING_MAX, 366
 OTF2_GeneralDefinitions.h, 366
 OTF2_MAPPING_METRIC, 365
 OTF2_GeneralDefinitions.h, 365
 OTF2_MAPPING_PARAMETER, 366
 OTF2_GeneralDefinitions.h, 366
 OTF2_MAPPING_REGION, 365
 OTF2_GeneralDefinitions.h, 365
 OTF2_MAPPING_RMA_WIN, 366
 OTF2_GeneralDefinitions.h, 366
 OTF2_MAPPING_STRING, 365
 OTF2_GeneralDefinitions.h, 365
 OTF2_Marker.h, 538
 OTF2_MARKER_SCOPE_COMM, 538
 OTF2_MARKER_SCOPE_GLOBAL, 538
 OTF2_MARKER_SCOPE_GROUP, 538
 OTF2_MARKER_SCOPE_LOCATION, 538
 OTF2_MARKER_SCOPE_LOCATION_GROUP, 538
 OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE, 538
 OTF2_MASTER, 102
 OTF2_Archive.h, 102
 OTF2_MEASUREMENT_OFF, 229
 OTF2_Events.h, 229
 OTF2_MEASUREMENT_ON, 229
 OTF2_Events.h, 229
 OTF2_METRIC_ABSOLUTE_LAST, 163
 OTF2_Definitions.h, 163
 OTF2_METRIC_ABSOLUTE_NEXT, 163
 OTF2_Definitions.h, 163
 OTF2_METRIC_ABSOLUTE_POINT, 163
 OTF2_Definitions.h, 163
 OTF2_METRIC_ACCUMULATED_LAST, 163
 OTF2_Definitions.h, 163
 OTF2_METRIC_ACCUMULATED_NEXT, 163
 OTF2_Definitions.h, 163
 OTF2_METRIC_ACCUMULATED_POINT, 163
 OTF2_Definitions.h, 163
 OTF2_METRIC_ACCUMULATED_START, 163
 OTF2_Definitions.h, 163

INDEX

OTF2_METRIC_ASYNCHRONOUS
OTF2_Definitions.h, [164](#)

OTF2_METRIC_RELATIVE_LAST
OTF2_Definitions.h, [163](#)

OTF2_METRIC_RELATIVE_NEXT
OTF2_Definitions.h, [163](#)

OTF2_METRIC_RELATIVE_POINT
OTF2_Definitions.h, [163](#)

OTF2_METRIC_SYNCHRONOUS
OTF2_Definitions.h, [164](#)

OTF2_METRIC_SYNCHRONOUS_STRICT
OTF2_Definitions.h, [164](#)

OTF2_METRIC_TIMING_LAST
OTF2_Definitions.h, [165](#)

OTF2_METRIC_TIMING_MASK
OTF2_Definitions.h, [165](#)

OTF2_METRIC_TIMING_NEXT
OTF2_Definitions.h, [165](#)

OTF2_METRIC_TIMING_POINT
OTF2_Definitions.h, [164](#)

OTF2_METRIC_TIMING_START
OTF2_Definitions.h, [164](#)

OTF2_METRIC_TYPE_OTHER
OTF2_Definitions.h, [165](#)

OTF2_METRIC_TYPE_PAPI
OTF2_Definitions.h, [165](#)

OTF2_METRIC_TYPE_RUSAGE
OTF2_Definitions.h, [165](#)

OTF2_METRIC_TYPE_USER
OTF2_Definitions.h, [165](#)

OTF2_METRIC_VALUE_ABSOLUTE
OTF2_Definitions.h, [165](#)

OTF2_METRIC_VALUE_ACCUMULATED
OTF2_Definitions.h, [165](#)

OTF2_METRIC_VALUE_MASK
OTF2_Definitions.h, [166](#)

OTF2_METRIC_VALUE_RELATIVE
OTF2_Definitions.h, [165](#)

OTF2_NO_FLUSH
OTF2_GeneralDefinitions.h, [365](#)

OTF2_PARADIGM_COMPILER
OTF2_GeneralDefinitions.h, [366](#)

OTF2_PARADIGM_CUDA
OTF2_GeneralDefinitions.h, [366](#)

OTF2_PARADIGM_MEASUREMENT_-\nSYSTEM
OTF2_GeneralDefinitions.h, [366](#)

OTF2_PARADIGM_MPI
OTF2_GeneralDefinitions.h, [366](#)

OTF2_PARADIGM_OPENMP
OTF2_GeneralDefinitions.h, [366](#)

OTF2_PARADIGM_UNKNOWN
OTF2_GeneralDefinitions.h, [366](#)

OTF2_PARADIGM_USER
OTF2_GeneralDefinitions.h, [366](#)

OTF2_PARAMETER_TYPE_INT64
OTF2_Definitions.h, [166](#)

OTF2_PARAMETER_TYPE_STRING
OTF2_Definitions.h, [166](#)

OTF2_PARAMETER_TYPE_UINT64
OTF2_Definitions.h, [166](#)

OTF2_RECORDER_KIND_ABSTRACT
OTF2_Definitions.h, [166](#)

OTF2_RECORDER_KIND_CPU
OTF2_Definitions.h, [166](#)

OTF2_RECORDER_KIND_GPU
OTF2_Definitions.h, [166](#)

OTF2_RECORDER_KIND_UNKNOWN
OTF2_Definitions.h, [166](#)

OTF2_REGION_FLAG_DYNAMIC
OTF2_Definitions.h, [167](#)

OTF2_REGION_FLAG_NONE
OTF2_Definitions.h, [167](#)

OTF2_REGION_FLAG_PHASE
OTF2_Definitions.h, [167](#)

OTF2_REGION_ROLE_ARTIFICIAL
OTF2_Definitions.h, [168](#)

OTF2_REGION_ROLE_ATOMIC
OTF2_Definitions.h, [168](#)

OTF2_REGION_ROLE_BARRIER
OTF2_Definitions.h, [168](#)

OTF2_REGION_ROLE_CODE
OTF2_Definitions.h, [167](#)

OTF2_REGION_ROLE_COLL_ALL2ALL
OTF2_Definitions.h, [168](#)

OTF2_REGION_ROLE_COLL_ALL2ONE
OTF2_Definitions.h, [168](#)

OTF2_REGION_ROLE_COLL_ONE2ALL

OTF2_Definitions.h, 168	OTF2_Definitions.h, 168
OTF2_REGION_ROLE_COLL_OTHER	OTF2_REGION_ROLE_UNKNOWN
OTF2_Definitions.h, 168	OTF2_Definitions.h, 167
OTF2_REGION_ROLE_CRITICAL	OTF2_REGION_ROLE_WORKSHARE
OTF2_Definitions.h, 168	OTF2_Definitions.h, 167
OTF2_REGION_ROLE_CRITICAL_SBLOCK	OTF2_REGION_ROLE_WRAPPER
OTF2_Definitions.h, 168	OTF2_Definitions.h, 167
OTF2_REGION_ROLE_DATA_TRANSFER	OTF2_RMA_SYNC_LEVEL_MEMORY
OTF2_Definitions.h, 168	OTF2_Events.h, 229
OTF2_REGION_ROLE_FILE_IO	OTF2_RMA_SYNC_LEVEL_NONE
OTF2_Definitions.h, 168	OTF2_Events.h, 229
OTF2_REGION_ROLE_FLUSH	OTF2_RMA_SYNC_LEVEL_PROCESS
OTF2_Definitions.h, 168	OTF2_Events.h, 229
OTF2_REGION_ROLE_FUNCTION	OTF2_RMA_SYNC_TYPE_MEMORY
OTF2_Definitions.h, 167	OTF2_Events.h, 230
OTF2_REGION_ROLE_IMPLICIT_BARRIER	OTF2_RMA_SYNC_TYPE_NOTIFY_IN
OTF2_Definitions.h, 168	OTF2_Events.h, 230
OTF2_REGION_ROLE_LOOP	OTF2_RMA_SYNC_TYPE_NOTIFY_OUT
OTF2_Definitions.h, 167	OTF2_Events.h, 230
OTF2_REGION_ROLE_MASTER	OTF2_SCOPE_GROUP
OTF2_Definitions.h, 168	OTF2_Definitions.h, 164
OTF2_REGION_ROLE_ORDERED	OTF2_SCOPE_LOCATION
OTF2_Definitions.h, 168	OTF2_Definitions.h, 164
OTF2_REGION_ROLE_ORDERED_SBLOCK	OTF2_SCOPE_LOCATION_GROUP
OTF2_Definitions.h, 167	OTF2_Definitions.h, 164
OTF2_REGION_ROLE_PARALLEL	OTF2_SCOPE_SYSTEM_TREE_NODE
OTF2_Definitions.h, 168	OTF2_Definitions.h, 164
OTF2_REGION_ROLE_POINT2POINT	OTF2_SEVERITY_HIGH
OTF2_Definitions.h, 168	OTF2_Marker.h, 539
OTF2_REGION_ROLE_RMA	OTF2_SEVERITY_LOW
OTF2_Definitions.h, 168	OTF2_Marker.h, 539
OTF2_REGION_ROLE_SECTION	OTF2_SEVERITY_MEDIUM
OTF2_Definitions.h, 167	OTF2_Marker.h, 539
OTF2_REGION_ROLE_SECTIONS	OTF2_SEVERITY_NONE
OTF2_Definitions.h, 167	OTF2_Marker.h, 539
OTF2_REGION_ROLE_SINGLE	OTF2_SLAVE
OTF2_Definitions.h, 167	OTF2_Archive.h, 102
OTF2_REGION_ROLE_SINGLE_SBLOCK	OTF2_SUBSTRATE_NONE
OTF2_Definitions.h, 167	OTF2_GeneralDefinitions.h, 364
OTF2_REGION_ROLE_TASK	OTF2_SUBSTRATE_POSIX
OTF2_Definitions.h, 168	OTF2_GeneralDefinitions.h, 364
OTF2_REGION_ROLE_TASK_CREATE	OTF2_SUBSTRATE_SION
OTF2_Definitions.h, 168	
OTF2_REGION_ROLE_TASK_WAIT	

INDEX

OTF2_GeneralDefinitions.h, [364](#)
OTF2_SUBSTRATE_UNDEFINED
OTF2_GeneralDefinitions.h, [364](#)
OTF2_SUCCESS
OTF2_ErrorCodes.h, [221](#)
OTF2_SYSTEM_TREE_DOMAIN_CACHE
OTF2_Definitions.h, [169](#)
OTF2_SYSTEM_TREE_DOMAIN_CORE
OTF2_Definitions.h, [169](#)
OTF2_SYSTEM_TREE_DOMAIN_MACHINE
OTF2_Definitions.h, [169](#)
OTF2_SYSTEM_TREE_DOMAIN_NUMA
OTF2_Definitions.h, [169](#)
OTF2_SYSTEM_TREE_DOMAIN_PU
OTF2_Definitions.h, [169](#)
OTF2_SYSTEM_TREE_DOMAIN_SHARED_MEMORY
OTF2_Definitions.h, [169](#)
OTF2_SYSTEM_TREE_DOMAIN_SOCKET
OTF2_Definitions.h, [169](#)
OTF2_THUMBNAIL_TYPE_ATTRIBUTES
OTF2_GeneralDefinitions.h, [367](#)
OTF2_THUMBNAIL_TYPE_METRIC
OTF2_GeneralDefinitions.h, [367](#)
OTF2_THUMBNAIL_TYPE_REGION
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_DOUBLE
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_FLOAT
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_INT16
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_INT32
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_INT64
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_INT8
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_NONE
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_UINT16
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_UINT32
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_UINT64
OTF2_GeneralDefinitions.h, [367](#)
OTF2_TYPE_UINT8
OTF2_GeneralDefinitions.h, [367](#)
OTF2_WARNING
OTF2_ErrorCodes.h, [221](#)
OTF2_Archive
OTF2_Archive.h, [101](#)
OTF2_Archive.h, [95](#)
OTF2_Archive, [101](#)
OTF2_Archive_Close, [102](#)
OTF2_Archive_CloseDefReader, [102](#)
OTF2_Archive_CloseDefWriter, [103](#)
OTF2_Archive_CloseEvtReader, [103](#)
OTF2_Archive_CloseEvtWriter, [103](#)
OTF2_Archive_CloseGlobalDefReader,
[104](#)
OTF2_Archive_CloseGlobalEvtReader,
[104](#)
OTF2_Archive_CloseGlobalSnapReader,
[105](#)
OTF2_Archive_CloseMarkerReader,
[105](#)
OTF2_Archive_CloseMarkerWriter,
[105](#)
OTF2_Archive_CloseSnapReader, [106](#)
OTF2_Archive_CloseSnapWriter, [106](#)
OTF2_Archive_CloseThumbReader,
[107](#)
OTF2_Archive_GetChunkSize, [107](#)
OTF2_Archive_GetCompression, [107](#)
OTF2_Archive_GetCreator, [108](#)
OTF2_Archive_GetDefReader, [108](#)
OTF2_Archive_GetDefWriter, [108](#)
OTF2_Archive_GetDescription, [109](#)
OTF2_Archive_GetEvtReader, [109](#)
OTF2_Archive_GetEvtWriter, [109](#)
OTF2_Archive_GetFileSubstrate, [110](#)
OTF2_Archive_GetGlobalDefReader,
[110](#)
OTF2_Archive_GetGlobalDefWriter,
[110](#)
OTF2_Archive_GetGlobalEvtReader,
[111](#)

OTF2_Archive_GetGlobalSnapReader, 111
 OTF2_Archive_GetMachineName, 111
 OTF2_Archive_GetMarkerReader, 112
 OTF2_Archive_GetMarkerWriter, 112
 OTF2_Archive_GetMasterSlaveMode, 113
 OTF2_Archive_GetNumberOfGlobalDefinitions, 113
 OTF2_Archive_GetNumberOfLocations, 113
 OTF2_Archive_GetNumberOfSnapshots, 114
 OTF2_Archive_GetNumberOfThumbnails, 114
 OTF2_Archive_GetProperty, 114
 OTF2_Archive_GetPropertyNames, 115
 OTF2_Archive_GetSnapReader, 115
 OTF2_Archive_GetSnapWriter, 116
 OTF2_Archive_GetThumbReader, 116
 OTF2_Archive_GetThumbWriter, 116
 OTF2_Archive_GetTraceId, 117
 OTF2_Archive_GetVersion, 117
 OTF2_Archive_Open, 118
 OTF2_Archive_SetBoolProperty, 119
 OTF2_Archive_SetCreator, 120
 OTF2_Archive_SetDescription, 120
 OTF2_Archive_SetFileSionCallbacks, 121
 OTF2_Archive_SetFlushCallbacks, 121
 OTF2_Archive_SetMachineName, 121
 OTF2_Archive_SetMasterSlaveMode, 122
 OTF2_Archive_SetMemoryCallbacks, 122
 OTF2_Archive_SetNumberOfSnapshots, 123
 OTF2_Archive_SetProperty, 123
 OTF2_Archive_SwitchFileMode, 124
 OTF2_CHUNK_SIZE_DEFINITIONS, 101
 OTF2_CHUNK_SIZE_EVENTS_DEFAULT, 101
 OTF2_MasterSlaveMode, 102
 OTF2_MasterSlaveMode_enum, 102
 OTF2_Archive_Close
 OTF2_Archive.h, 102
 OTF2_Archive_CloseDefReader
 OTF2_Archive.h, 102
 OTF2_Archive_CloseDefWriter
 OTF2_Archive.h, 103
 OTF2_Archive_CloseEvtReader
 OTF2_Archive.h, 103
 OTF2_Archive_CloseEvtWriter
 OTF2_Archive.h, 103
 OTF2_Archive_CloseGlobalDefReader
 OTF2_Archive.h, 104
 OTF2_Archive_CloseGlobalEvtReader
 OTF2_Archive.h, 104
 OTF2_Archive_CloseGlobalSnapReader
 OTF2_Archive.h, 105
 OTF2_Archive_CloseMarkerReader
 OTF2_Archive.h, 105
 OTF2_Archive_CloseMarkerWriter
 OTF2_Archive.h, 105
 OTF2_Archive_CloseSnapReader
 OTF2_Archive.h, 106
 OTF2_Archive_CloseSnapWriter
 OTF2_Archive.h, 106
 OTF2_Archive_CloseThumbReader
 OTF2_Archive.h, 107
 OTF2_Archive_GetChunkSize
 OTF2_Archive.h, 107
 OTF2_Archive_GetCompression
 OTF2_Archive.h, 107
 OTF2_Archive_GetCreator
 OTF2_Archive.h, 108
 OTF2_Archive_GetDefReader
 OTF2_Archive.h, 108
 OTF2_Archive_GetDefWriter
 OTF2_Archive.h, 108
 OTF2_Archive_GetDescription
 OTF2_Archive.h, 109
 OTF2_Archive_GetEvtReader
 OTF2_Archive.h, 109

INDEX

- | | |
|---|--|
| OTF2_Archive_GetEvtWriter | OTF2_Archive_Open |
| OTF2_Archive.h, 109 | OTF2_Archive.h, 118 |
| OTF2_Archive_GetFileSubstrate | OTF2_Archive_SetBoolProperty |
| OTF2_Archive.h, 110 | OTF2_Archive.h, 119 |
| OTF2_Archive_GetGlobalDefReader | OTF2_Archive_SetCreator |
| OTF2_Archive.h, 110 | OTF2_Archive.h, 120 |
| OTF2_Archive_GetGlobalDefWriter | OTF2_Archive_SetDescription |
| OTF2_Archive.h, 110 | OTF2_Archive.h, 120 |
| OTF2_Archive_GetGlobalEvtReader | OTF2_Archive_SetFileSionCallbacks |
| OTF2_Archive.h, 111 | OTF2_Archive.h, 121 |
| OTF2_Archive_GetGlobalSnapReader | OTF2_Archive_SetFlushCallbacks |
| OTF2_Archive.h, 111 | OTF2_Archive.h, 121 |
| OTF2_Archive_GetMachineName | OTF2_Archive_SetMachineName |
| OTF2_Archive.h, 111 | OTF2_Archive.h, 121 |
| OTF2_Archive_GetMarkerReader | OTF2_Archive_SetMasterSlaveMode |
| OTF2_Archive.h, 112 | OTF2_Archive.h, 122 |
| OTF2_Archive_GetMarkerWriter | OTF2_Archive_SetMemoryCallbacks |
| OTF2_Archive.h, 112 | OTF2_Archive.h, 122 |
| OTF2_Archive_GetMasterSlaveMode | OTF2_Archive_SetNumberOfSnapshots |
| OTF2_Archive.h, 113 | OTF2_Archive.h, 123 |
| OTF2_Archive_GetNumberOfGlobalDefinitions | OTF2_Archive_SetProperty |
| OTF2_Archive.h, 113 | OTF2_Archive.h, 123 |
| OTF2_Archive_GetNumberOfLocations | OTF2_Archive_SwitchFileMode |
| OTF2_Archive.h, 113 | OTF2_Archive.h, 124 |
| OTF2_Archive_GetNumberOfSnapshots | OTF2_AttributeList.h, 124 |
| OTF2_Archive.h, 114 | OTF2_AttributeList_AddAttribute, 130 |
| OTF2_Archive_GetNumberOfThumbnails | OTF2_AttributeList_AddAttributeRef, |
| OTF2_Archive.h, 114 | 131 |
| OTF2_Archive_GetProperty | OTF2_AttributeList_AddCommRef, |
| OTF2_Archive.h, 114 | 131 |
| OTF2_Archive_GetPropertyNames | OTF2_AttributeList_AddDouble, 131 |
| OTF2_Archive.h, 115 | OTF2_AttributeList_AddFloat, 132 |
| OTF2_Archive_GetSnapReader | OTF2_AttributeList_AddGroupRef, |
| OTF2_Archive.h, 115 | 132 |
| OTF2_Archive_GetSnapWriter | OTF2_AttributeList_AddInt16, 133 |
| OTF2_Archive.h, 116 | OTF2_AttributeList_AddInt32, 133 |
| OTF2_Archive_GetThumbReader | OTF2_AttributeList_AddInt64, 133 |
| OTF2_Archive.h, 116 | OTF2_AttributeList_AddInt8, 134 |
| OTF2_Archive_GetThumbWriter | OTF2_AttributeList_AddLocationRef, |
| OTF2_Archive.h, 116 | 134 |
| OTF2_Archive_GetTraceId | OTF2_AttributeList_AddMetricRef, |
| OTF2_Archive.h, 117 | 134 |
| OTF2_Archive_GetVersion | OTF2_AttributeList_AddParameterRef, |
| OTF2_Archive.h, 117 | 135 |

OTF2_AttributeList_AddRegionRef, 135
 OTF2_AttributeList_AddRmaWinRef, 136
 OTF2_AttributeList_AddString, 136
 OTF2_AttributeList_AddStringRef, 137
 OTF2_AttributeList_AddUInt16, 137
 OTF2_AttributeList_AddUInt32, 137
 OTF2_AttributeList_AddUInt64, 138
 OTF2_AttributeList_AddUInt8, 138
 OTF2_AttributeList_Delete, 138
 OTF2_AttributeList_GetAttributeByID, 139
 OTF2_AttributeList_GetAttributeByIndex, 139
 OTF2_AttributeList_GetAttributeRef, 140
 OTF2_AttributeList_GetCommRef, 140
 OTF2_AttributeList_GetDouble, 140
 OTF2_AttributeList_GetFloat, 141
 OTF2_AttributeList_GetGroupRef, 141
 OTF2_AttributeList_GetInt16, 142
 OTF2_AttributeList_GetInt32, 142
 OTF2_AttributeList_GetInt64, 142
 OTF2_AttributeList_GetInt8, 143
 OTF2_AttributeList_GetLocationRef, 143
 OTF2_AttributeList_GetMetricRef, 144
 OTF2_AttributeList_GetNumberOfElements, 144
 OTF2_AttributeList_GetParameterRef, 144
 OTF2_AttributeList_GetRegionRef, 145
 OTF2_AttributeList_GetRmaWinRef, 145
 OTF2_AttributeList_GetString, 146
 OTF2_AttributeList_GetStringRef, 146
 OTF2_AttributeList_GetUInt16, 147
 OTF2_AttributeList_GetUInt32, 147
 OTF2_AttributeList_GetUInt64, 147
 OTF2_AttributeList_GetUInt8, 148
 OTF2_AttributeList_New, 148
 OTF2_AttributeList_PopAttribute, 148
 OTF2_AttributeList_RemoveAllAttributes, 149
 OTF2_AttributeList_RemoveAttribute, 149
 OTF2_AttributeList_TestAttributeByID, 150
 OTF2_AttributeList_AddAttribute, 130
 OTF2_AttributeList.h, 130
 OTF2_AttributeList_AddAttributeRef, 131
 OTF2_AttributeList.h, 131
 OTF2_AttributeList_AddCommRef, 131
 OTF2_AttributeList.h, 131
 OTF2_AttributeList_AddDouble, 131
 OTF2_AttributeList.h, 131
 OTF2_AttributeList_AddFloat, 132
 OTF2_AttributeList.h, 132
 OTF2_AttributeList_AddGroupRef, 132
 OTF2_AttributeList.h, 132
 OTF2_AttributeList_AddInt16, 133
 OTF2_AttributeList.h, 133
 OTF2_AttributeList_AddInt32, 133
 OTF2_AttributeList.h, 133
 OTF2_AttributeList_AddInt64, 133
 OTF2_AttributeList.h, 133
 OTF2_AttributeList_AddInt8, 134
 OTF2_AttributeList.h, 134
 OTF2_AttributeList_AddLocationRef, 134
 OTF2_AttributeList.h, 134
 OTF2_AttributeList_AddMetricRef, 134
 OTF2_AttributeList.h, 134
 OTF2_AttributeList_AddParameterRef, 135
 OTF2_AttributeList.h, 135
 OTF2_AttributeList_AddRegionRef, 135
 OTF2_AttributeList.h, 135
 OTF2_AttributeList_AddRmaWinRef, 136
 OTF2_AttributeList.h, 136
 OTF2_AttributeList_AddString, 136
 OTF2_AttributeList.h, 136
 OTF2_AttributeList_AddStringRef, 137
 OTF2_AttributeList.h, 137

INDEX

- OTF2_AttributeList_AddUInt16
OTF2_AttributeList.h, [137](#)
- OTF2_AttributeList_AddUInt32
OTF2_AttributeList.h, [137](#)
- OTF2_AttributeList_AddUInt64
OTF2_AttributeList.h, [138](#)
- OTF2_AttributeList_AddUInt8
OTF2_AttributeList.h, [138](#)
- OTF2_AttributeList_Delete
OTF2_AttributeList.h, [138](#)
- OTF2_AttributeList_GetAttributeByID
OTF2_AttributeList.h, [139](#)
- OTF2_AttributeList_GetAttributeByIndex
OTF2_AttributeList.h, [139](#)
- OTF2_AttributeList_GetAttributeRef
OTF2_AttributeList.h, [140](#)
- OTF2_AttributeList_GetCommRef
OTF2_AttributeList.h, [140](#)
- OTF2_AttributeList_GetDouble
OTF2_AttributeList.h, [140](#)
- OTF2_AttributeList_GetFloat
OTF2_AttributeList.h, [141](#)
- OTF2_AttributeList_GetGroupRef
OTF2_AttributeList.h, [141](#)
- OTF2_AttributeList_GetInt16
OTF2_AttributeList.h, [142](#)
- OTF2_AttributeList_GetInt32
OTF2_AttributeList.h, [142](#)
- OTF2_AttributeList_GetInt64
OTF2_AttributeList.h, [142](#)
- OTF2_AttributeList_GetInt8
OTF2_AttributeList.h, [143](#)
- OTF2_AttributeList_GetLocationRef
OTF2_AttributeList.h, [143](#)
- OTF2_AttributeList_GetMetricRef
OTF2_AttributeList.h, [144](#)
- OTF2_AttributeList_GetNumberOfElements
OTF2_AttributeList.h, [144](#)
- OTF2_AttributeList_GetParameterRef
OTF2_AttributeList.h, [144](#)
- OTF2_AttributeList_GetRegionRef
OTF2_AttributeList.h, [145](#)
- OTF2_AttributeList_GetRmaWinRef
OTF2_AttributeList.h, [145](#)
- OTF2_AttributeList_GetString
OTF2_AttributeList.h, [146](#)
- OTF2_AttributeList_GetStringRef
OTF2_AttributeList.h, [146](#)
- OTF2_AttributeList_GetUInt16
OTF2_AttributeList.h, [147](#)
- OTF2_AttributeList_GetUInt32
OTF2_AttributeList.h, [147](#)
- OTF2_AttributeList_GetUInt64
OTF2_AttributeList.h, [147](#)
- OTF2_AttributeList_GetUInt8
OTF2_AttributeList.h, [148](#)
- OTF2_AttributeList_New
OTF2_AttributeList.h, [148](#)
- OTF2_AttributeList_PopAttribute
OTF2_AttributeList.h, [148](#)
- OTF2_AttributeList_RemoveAllAttributes
OTF2_AttributeList.h, [149](#)
- OTF2_AttributeList_RemoveAttribute
OTF2_AttributeList.h, [149](#)
- OTF2_AttributeList_TestAttributeByID
OTF2_AttributeList.h, [150](#)
- OTF2_AttributeValue, [89](#)
- OTF2_CallbackCode
OTF2_GeneralDefinitions.h, [363](#)
- OTF2_Callbacks.h, [150](#)
- OTF2_FileSionClose, [152](#)
- OTF2_FileSionGetRank, [152](#)
- OTF2_FileSionOpen, [152](#)
- OTF2_MemoryAllocate, [153](#)
- OTF2_MemoryFreeAll, [154](#)
- OTF2_PostFlushCallback, [154](#)
- OTF2_PreFlushCallback, [155](#)
- OTF2_CHUNK_SIZE_DEFINITIONS_
DEFAULT
OTF2_Archive.h, [101](#)
- OTF2_CHUNK_SIZE_EVENTS_DEFAULT
OTF2_Archive.h, [101](#)
- OTF2_CollectiveOp_enum
OTF2_Events.h, [228](#)
- OTF2_Compression_enum
OTF2_GeneralDefinitions.h, [363](#)
- OTF2_Definitions.h, [155](#)
- OTF2_GroupFlag_enum, [161](#)

OTF2_GroupType_enum, 161
 OTF2_LocationGroupType_enum, 160
 OTF2_LocationType_enum, 162
 OTF2_MetricBase_enum, 163
 OTF2_MetricMode_enum, 163
 OTF2_MetricOccurrence_enum, 163
 OTF2_MetricScope_enum, 164
 OTF2_MetricTiming_enum, 164
 OTF2_MetricType_enum, 165
 OTF2_MetricValueProperty_enum, 165
 OTF2_ParameterType_enum, 166
 OTF2_RecorderKind_enum, 166
 OTF2_RegionFlag_enum, 166
 OTF2_RegionRole_enum, 167
 OTF2_SystemTreeDomain_enum, 168
 OTF2_DefReader.h, 169
 OTF2_DefReader_GetLocationID, 170
 OTF2_DefReader_ReadDefinitions, 170
 OTF2_DefReader_SetCallbacks, 171
 OTF2_DefReader_GetLocationID
 OTF2_DefReader.h, 170
 OTF2_DefReader_ReadDefinitions
 OTF2_DefReader.h, 170
 OTF2_DefReader_SetCallbacks
 OTF2_DefReader.h, 171
 OTF2_DefReaderCallback_Attribute
 OTF2_DefReaderCallbacks.h, 178
 OTF2_DefReaderCallback_Callpath
 OTF2_DefReaderCallbacks.h, 178
 OTF2_DefReaderCallback_Callsite
 OTF2_DefReaderCallbacks.h, 179
 OTF2_DefReaderCallback_ClockOffset
 OTF2_DefReaderCallbacks.h, 179
 OTF2_DefReaderCallback_Comm
 OTF2_DefReaderCallbacks.h, 180
 OTF2_DefReaderCallback_Group
 OTF2_DefReaderCallbacks.h, 180
 OTF2_DefReaderCallback_Location
 OTF2_DefReaderCallbacks.h, 181
 OTF2_DefReaderCallback_LocationGroup
 OTF2_DefReaderCallbacks.h, 182
 OTF2_DefReaderCallback_MappingTable
 OTF2_DefReaderCallbacks.h, 182
 OTF2_DefReaderCallback_MetricClass
 OTF2_DefReaderCallbacks.h, 183
 OTF2_DefReaderCallback_MetricClassRecorder
 OTF2_DefReaderCallbacks.h, 184
 OTF2_DefReaderCallback_MetricInstance
 OTF2_DefReaderCallbacks.h, 184
 OTF2_DefReaderCallback_MetricMember
 OTF2_DefReaderCallbacks.h, 185
 OTF2_DefReaderCallback_Parameter
 OTF2_DefReaderCallbacks.h, 186
 OTF2_DefReaderCallback_Region
 OTF2_DefReaderCallbacks.h, 187
 OTF2_DefReaderCallback_RmaWin
 OTF2_DefReaderCallbacks.h, 188
 OTF2_DefReaderCallback_String
 OTF2_DefReaderCallbacks.h, 188
 OTF2_DefReaderCallback_SystemTreeNode
 OTF2_DefReaderCallbacks.h, 189
 OTF2_DefReaderCallback_SystemTreeNodeDomain
 OTF2_DefReaderCallbacks.h, 189
 OTF2_DefReaderCallback_SystemTreeNodeProperty
 OTF2_DefReaderCallbacks.h, 190
 OTF2_DefReaderCallback_Unknown
 OTF2_DefReaderCallbacks.h, 191
 OTF2_DefReaderCallbacks.h, 172
 OTF2_DefReaderCallback_Attribute,
 178
 OTF2_DefReaderCallback_Callpath,
 178
 OTF2_DefReaderCallback_Callsite,
 179
 OTF2_DefReaderCallback_ClockOffset,
 179
 OTF2_DefReaderCallback_Comm,
 180
 OTF2_DefReaderCallback_Group, 180
 OTF2_DefReaderCallback_Location,
 181
 OTF2_DefReaderCallback_LocationGroup,
 182
 OTF2_DefReaderCallback_MappingTable,
 182

INDEX

OTF2_DefReaderCallback_MetricClass, 183
OTF2_DefReaderCallback_MetricClassRecorder, 184
OTF2_DefReaderCallback_MetricInstance, 184
OTF2_DefReaderCallback_MetricMember, 185
OTF2_DefReaderCallback_Parameter, 186
OTF2_DefReaderCallback_Region, 187
OTF2_DefReaderCallback_RmaWin, 188
OTF2_DefReaderCallback_String, 188
OTF2_DefReaderCallback_SystemTreeNode, 189
OTF2_DefReaderCallback_SystemTreeNodeDomain, 189
OTF2_DefReaderCallback_SystemTreeNodeProperty, 190
OTF2_DefReaderCallback_Unknown, 191
OTF2_DefReaderCallbacks_Clear, 191
OTF2_DefReaderCallbacks_Delete, 191
OTF2_DefReaderCallbacks_New, 191
OTF2_DefReaderCallbacks_SetAttributeCallback, 192
OTF2_DefReaderCallbacks_SetCallpathCallback, 192
OTF2_DefReaderCallbacks_SetCallsiteCallback, 193
OTF2_DefReaderCallbacks_SetClockOffsetCallback, 193
OTF2_DefReaderCallbacks_SetCommCallback, 193
OTF2_DefReaderCallbacks_SetGroupCallback, 194
OTF2_DefReaderCallbacks_SetLocationCallback, 194
OTF2_DefReaderCallbacks_SetLocationCallback, 195
OTF2_DefReaderCallbacks_SetMappingTableCallback, 195
OTF2_DefReaderCallbacks_SetMetricClassCallback, 196
OTF2_DefReaderCallbacks_SetMetricClassRecorderCallback, 196
OTF2_DefReaderCallbacks_SetMetricInstanceCallback, 197
OTF2_DefReaderCallbacks_SetMetricMemberCallback, 197
OTF2_DefReaderCallbacks_SetParameterCallback, 198
OTF2_DefReaderCallbacks_SetRegionCallback, 198
OTF2_DefReaderCallbacks_SetRmaWinCallback, 199
OTF2_DefReaderCallbacks_SetStringCallback, 199
OTF2_DefReaderCallbacks_SetSystemTreeNodeDomainCallback, 200
OTF2_DefReaderCallbacks_SetSystemTreeNodePropertyCallback, 200
OTF2_DefReaderCallbacks_SetUnknownCallback, 201
OTF2_DefReaderCallbacks_Clear, 191
OTF2_DefReaderCallbacks.h, 191
OTF2_DefReaderCallbacks_Delete, 191
OTF2_DefReaderCallbacks.h, 191
OTF2_DefReaderCallbacks_New, 191
OTF2_DefReaderCallbacks.h, 191
OTF2_DefReaderCallbacks_SetAttributeCallback, 192
OTF2_DefReaderCallbacks.h, 192
OTF2_DefReaderCallbacks_SetCallpathCallback, 192
OTF2_DefReaderCallbacks.h, 192
OTF2_DefReaderCallbacks_SetCallsiteCallback, 193
OTF2_DefReaderCallbacks.h, 193
OTF2_DefReaderCallbacks_SetClockOffsetCallback, 193
OTF2_DefReaderCallbacks.h, 193
OTF2_DefReaderCallbacks_SetCommCallback, 193
OTF2_DefReaderCallbacks.h, 193
OTF2_DefReaderCallbacks_SetGroupCallback, 194
OTF2_DefReaderCallbacks.h, 194
OTF2_DefReaderCallbacks_SetLocationCallback, 194
OTF2_DefReaderCallbacks.h, 194
OTF2_DefReaderCallbacks_SetLocationCallback, 195
OTF2_DefReaderCallbacks.h, 195

OTF2_DefReaderCallbacks_SetLocationCallback, [OTF2_DefWriter_WriteMetricClass](#),
 OTF2_DefReaderCallbacks.h, [194](#) [210](#)
 OTF2_DefReaderCallbacks_SetLocationGroupCallback, [OTF2_DefWriter_WriteMetricClassRecorder](#),
 OTF2_DefReaderCallbacks.h, [195](#) [210](#)
 OTF2_DefReaderCallbacks_SetMappingTableCallback, [OTF2_DefWriter_WriteMetricInstance](#),
 OTF2_DefReaderCallbacks.h, [195](#) [211](#)
 OTF2_DefReaderCallbacks_SetMetricClassCallback, [OTF2_DefWriter_WriteMetricMember](#),
 OTF2_DefReaderCallbacks.h, [196](#) [211](#)
 OTF2_DefReaderCallbacks_SetMetricClassRecorderCallback, [OTF2_DefWriter_WriteParameter](#), [212](#)
 OTF2_DefReaderCallbacks.h, [196](#) [OTF2_DefWriter_WriteRegion](#), [213](#)
 OTF2_DefReaderCallbacks_SetMetricInstanceCallback, [OTF2_DefWriter_WriteRmaWin](#), [214](#)
 OTF2_DefReaderCallbacks.h, [197](#) [OTF2_DefWriter_WriteString](#), [214](#)
 OTF2_DefReaderCallbacks_SetMetricMemberCallback, [OTF2_DefWriter_WriteSystemTreeNode](#),
 OTF2_DefReaderCallbacks.h, [197](#) [215](#)
 OTF2_DefReaderCallbacks_SetParameterCallback, [OTF2_DefWriter_WriteSystemTreeNodeDomain](#),
 OTF2_DefReaderCallbacks.h, [198](#) [215](#)
 OTF2_DefReaderCallbacks_SetRegionCallback, [OTF2_DefWriter_WriteSystemTreeNodeProperty](#),
 OTF2_DefReaderCallbacks.h, [198](#) [216](#)
 OTF2_DefReaderCallbacks_SetRmaWinCallback, [OTF2_DefWriter_GetLocationID](#)
 OTF2_DefReaderCallbacks.h, [199](#) [OTF2_DefWriter.h](#), [204](#)
 OTF2_DefReaderCallbacks_SetStringCallback, [OTF2_DefWriter_WriteAttribute](#)
 OTF2_DefReaderCallbacks.h, [199](#) [OTF2_DefWriter.h](#), [205](#)
 OTF2_DefReaderCallbacks_SetSystemTreeNodeCallback, [OTF2_DefWriter_WriteCallpath](#)
 OTF2_DefReaderCallbacks.h, [200](#) [OTF2_DefWriter.h](#), [205](#)
 OTF2_DefReaderCallbacks_SetSystemTreeNodeDomainCallback, [OTF2_DefWriter_WriteCallsite](#)
 OTF2_DefReaderCallbacks.h, [200](#) [OTF2_DefWriter.h](#), [206](#)
 OTF2_DefReaderCallbacks_SetSystemTreeNodePropertyCallback, [OTF2_DefWriter_WriteClockOffset](#)
 OTF2_DefReaderCallbacks.h, [201](#) [OTF2_DefWriter.h](#), [206](#)
 OTF2_DefReaderCallbacks_SetUnknownCallback, [OTF2_DefWriter_WriteComm](#)
 OTF2_DefReaderCallbacks.h, [201](#) [OTF2_DefWriter.h](#), [206](#)
 OTF2_DefWriter.h, [202](#) [OTF2_DefWriter_WriteGroup](#)
 OTF2_DefWriter_GetLocationID, [204](#) [OTF2_DefWriter.h](#), [207](#)
 OTF2_DefWriter_WriteAttribute, [205](#) [OTF2_DefWriter_WriteLocation](#)
 OTF2_DefWriter_WriteCallpath, [205](#) [OTF2_DefWriter.h](#), [208](#)
 OTF2_DefWriter_WriteCallsite, [206](#) [OTF2_DefWriter_WriteLocationGroup](#)
 OTF2_DefWriter_WriteClockOffset, [206](#) [OTF2_DefWriter.h](#), [208](#)
 OTF2_DefWriter_WriteMappingTable
 OTF2_DefWriter_WriteComm, [206](#) [OTF2_DefWriter.h](#), [209](#)
 OTF2_DefWriter_WriteGroup, [207](#) [OTF2_DefWriter_WriteMetricClass](#)
 OTF2_DefWriter_WriteLocation, [208](#) [OTF2_DefWriter.h](#), [210](#)
 OTF2_DefWriter_WriteLocationGroup, [208](#) [OTF2_DefWriter_WriteMetricClassRecorder](#)
 OTF2_DefWriter.h, [210](#)
 OTF2_DefWriter_WriteMappingTable, [209](#) [OTF2_DefWriter_WriteMetricInstance](#)
 OTF2_DefWriter.h, [211](#)

INDEX

- OTF2_DefWriter_WriteMetricMember
 - OTF2_DefWriter.h, [211](#)
- OTF2_DefWriter_WriteParameter
 - OTF2_DefWriter.h, [212](#)
- OTF2_DefWriter_WriteRegion
 - OTF2_DefWriter.h, [213](#)
- OTF2_DefWriter_WriteRmaWin
 - OTF2_DefWriter.h, [214](#)
- OTF2_DefWriter_WriteString
 - OTF2_DefWriter.h, [214](#)
- OTF2_DefWriter_WriteSystemTreeNode
 - OTF2_DefWriter.h, [215](#)
- OTF2_DefWriter_WriteSystemTreeNodeDomain
 - OTF2_DefWriter.h, [215](#)
- OTF2_DefWriter_WriteSystemTreeNodeProperty
 - OTF2_DefWriter.h, [216](#)
- OTF2_Error_GetDescription
 - OTF2_ErrorCodes.h, [225](#)
- OTF2_Error_GetName
 - OTF2_ErrorCodes.h, [225](#)
- OTF2_Error_RegisterCallback
 - OTF2_ErrorCodes.h, [225](#)
- OTF2_ErrorCallback
 - OTF2_ErrorCodes.h, [221](#)
- OTF2_ErrorCode
 - OTF2_ErrorCodes.h, [221](#)
- OTF2_ErrorCodes.h, [217](#)
 - OTF2_Error_GetDescription, [225](#)
 - OTF2_Error_GetName, [225](#)
 - OTF2_Error_RegisterCallback, [225](#)
 - OTF2_ErrorCallback, [221](#)
 - OTF2_ErrorCode, [221](#)
- OTF2_Events.h, [226](#)
 - OTF2_CollectiveOp_enum, [228](#)
 - OTF2_LockType_enum, [228](#)
 - OTF2_MeasurementMode_enum, [228](#)
 - OTF2_RmaAtomicType_enum, [229](#)
 - OTF2_RmaSyncLevel_enum, [229](#)
 - OTF2_RmaSyncType_enum, [229](#)
- OTF2_EvtReader.h, [230](#)
 - OTF2_EvtReader_ApplyClockOffsets, [232](#)
 - OTF2_EvtReader_ApplyMappingTables, [232](#)
 - OTF2_EvtReader_GetLocationID, [232](#)
 - OTF2_EvtReader_GetPos, [233](#)
 - OTF2_EvtReader_ReadEvents, [233](#)
 - OTF2_EvtReader_ReadEventsBackward, [233](#)
 - OTF2_EvtReader_Seek, [234](#)
 - OTF2_EvtReader_SetCallbacks, [234](#)
 - OTF2_EvtReader_TimeStampRewrite, [235](#)
 - OTF2_EvtReader_ApplyClockOffsets
 - OTF2_EvtReader.h, [232](#)
 - OTF2_EvtReader_ApplyMappingTables
 - OTF2_EvtReader.h, [232](#)
 - OTF2_EvtReader_GetLocationID
 - OTF2_EvtReader.h, [232](#)
 - OTF2_EvtReader_GetPos
 - OTF2_EvtReader.h, [233](#)
 - OTF2_EvtReader_ReadEvents
 - OTF2_EvtReader.h, [233](#)
 - OTF2_EvtReader_ReadEventsBackward
 - OTF2_EvtReader.h, [233](#)
 - OTF2_EvtReader_Seek
 - OTF2_EvtReader.h, [234](#)
 - OTF2_EvtReader_SetCallbacks
 - OTF2_EvtReader.h, [234](#)
 - OTF2_EvtReader_TimeStampRewrite
 - OTF2_EvtReader.h, [235](#)
 - OTF2_EvtReaderCallback_BufferFlush
 - OTF2_EvtReaderCallbacks.h, [248](#)
 - OTF2_EvtReaderCallback_Enter
 - OTF2_EvtReaderCallbacks.h, [248](#)
 - OTF2_EvtReaderCallback_Leave
 - OTF2_EvtReaderCallbacks.h, [249](#)
 - OTF2_EvtReaderCallback_MeasurementOnOff
 - OTF2_EvtReaderCallbacks.h, [250](#)
 - OTF2_EvtReaderCallback_Metric
 - OTF2_EvtReaderCallbacks.h, [250](#)
 - OTF2_EvtReaderCallback_MpiCollectiveBegin
 - OTF2_EvtReaderCallbacks.h, [251](#)
 - OTF2_EvtReaderCallback_MpiCollectiveEnd
 - OTF2_EvtReaderCallbacks.h, [252](#)
 - OTF2_EvtReaderCallback_MpiIrecv
 - OTF2_EvtReaderCallbacks.h, [253](#)
 - OTF2_EvtReaderCallback_MpiIrecvRequest
 - OTF2_EvtReaderCallbacks.h, [253](#)

OTF2_EvtReaderCallbacks.h, [254](#) OTF2_EvtReaderCallbacks.h, [270](#)
 OTF2_EvtReaderCallback_MpiIsend OTF2_EvtReaderCallback_RmaOpCompleteBlocking
 OTF2_EvtReaderCallbacks.h, [254](#) OTF2_EvtReaderCallbacks.h, [271](#)
 OTF2_EvtReaderCallback_MpiIsendComplete OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking
 OTF2_EvtReaderCallbacks.h, [255](#) OTF2_EvtReaderCallbacks.h, [271](#)
 OTF2_EvtReaderCallback_MpiRecv OTF2_EvtReaderCallback_RmaOpCompleteRemote
 OTF2_EvtReaderCallbacks.h, [256](#) OTF2_EvtReaderCallbacks.h, [272](#)
 OTF2_EvtReaderCallback_MpiRequestCancel OTF2_EvtReaderCallback_RmaOpTest
 OTF2_EvtReaderCallbacks.h, [257](#) OTF2_EvtReaderCallbacks.h, [273](#)
 OTF2_EvtReaderCallback_MpiRequestTest OTF2_EvtReaderCallback_RmaPut
 OTF2_EvtReaderCallbacks.h, [257](#) OTF2_EvtReaderCallbacks.h, [274](#)
 OTF2_EvtReaderCallback_MpiSend OTF2_EvtReaderCallback_RmaReleaseLock
 OTF2_EvtReaderCallbacks.h, [258](#) OTF2_EvtReaderCallbacks.h, [274](#)
 OTF2_EvtReaderCallback_OmpAcquireLock OTF2_EvtReaderCallback_RmaRequestLock
 OTF2_EvtReaderCallbacks.h, [259](#) OTF2_EvtReaderCallbacks.h, [275](#)
 OTF2_EvtReaderCallback_OmpFork OTF2_EvtReaderCallback_RmaSync
 OTF2_EvtReaderCallbacks.h, [259](#) OTF2_EvtReaderCallbacks.h, [276](#)
 OTF2_EvtReaderCallback_OmpJoin OTF2_EvtReaderCallback_RmaTryLock
 OTF2_EvtReaderCallbacks.h, [260](#) OTF2_EvtReaderCallbacks.h, [277](#)
 OTF2_EvtReaderCallback_OmpReleaseLock OTF2_EvtReaderCallback_RmaWaitChange
 OTF2_EvtReaderCallbacks.h, [261](#) OTF2_EvtReaderCallbacks.h, [278](#)
 OTF2_EvtReaderCallback_OmpTaskComplete OTF2_EvtReaderCallback_RmaWinCreate
 OTF2_EvtReaderCallbacks.h, [262](#) OTF2_EvtReaderCallbacks.h, [278](#)
 OTF2_EvtReaderCallback_OmpTaskCreate OTF2_EvtReaderCallback_RmaWinDestroy
 OTF2_EvtReaderCallbacks.h, [262](#) OTF2_EvtReaderCallbacks.h, [279](#)
 OTF2_EvtReaderCallback_OmpTaskSwitch OTF2_EvtReaderCallback_ThreadAcquireLock
 OTF2_EvtReaderCallbacks.h, [263](#) OTF2_EvtReaderCallbacks.h, [280](#)
 OTF2_EvtReaderCallback_ParameterInt OTF2_EvtReaderCallback_ThreadFork
 OTF2_EvtReaderCallbacks.h, [264](#) OTF2_EvtReaderCallbacks.h, [280](#)
 OTF2_EvtReaderCallback_ParameterString OTF2_EvtReaderCallback_ThreadJoin
 OTF2_EvtReaderCallbacks.h, [264](#) OTF2_EvtReaderCallbacks.h, [281](#)
 OTF2_EvtReaderCallback_ParameterUnsignedInt OTF2_EvtReaderCallback_ThreadReleaseLock
 OTF2_EvtReaderCallbacks.h, [265](#) OTF2_EvtReaderCallbacks.h, [282](#)
 OTF2_EvtReaderCallback_RmaAcquireLock OTF2_EvtReaderCallback_ThreadTaskComplete
 OTF2_EvtReaderCallbacks.h, [266](#) OTF2_EvtReaderCallbacks.h, [282](#)
 OTF2_EvtReaderCallback_RmaAtomic OTF2_EvtReaderCallback_ThreadTaskCreate
 OTF2_EvtReaderCallbacks.h, [267](#) OTF2_EvtReaderCallbacks.h, [283](#)
 OTF2_EvtReaderCallback_RmaCollective OTF2_EvtReaderCallback_ThreadTaskSwitch
 OTF2_EvtReaderCallbacks.h, [268](#) OTF2_EvtReaderCallbacks.h, [284](#)
 OTF2_EvtReaderCallback_RmaCollectiveWait OTF2_EvtReaderCallback_ThreadTeamBegin
 OTF2_EvtReaderCallbacks.h, [268](#) OTF2_EvtReaderCallbacks.h, [285](#)
 OTF2_EvtReaderCallback_RmaGet OTF2_EvtReaderCallback_ThreadTeamEnd
 OTF2_EvtReaderCallbacks.h, [269](#) OTF2_EvtReaderCallbacks.h, [285](#)
 OTF2_EvtReaderCallback_RmaGroupSync OTF2_EvtReaderCallback_Unknown

INDEX

OTF2_EvtReaderCallbacks.h, 286	OTF2_EvtReaderCallback_ParameterInt, 264
OTF2_EvtReaderCallbacks.h, 235	OTF2_EvtReaderCallback_ParameterString, 264
OTF2_EvtReaderCallback_BufferFlush, 248	OTF2_EvtReaderCallback_ParameterUnsignedInt, 265
OTF2_EvtReaderCallback_Enter, 248	OTF2_EvtReaderCallback_RmaAcquireLock, 266
OTF2_EvtReaderCallback_Leave, 249	OTF2_EvtReaderCallback_RmaAtomic, 267
OTF2_EvtReaderCallback_MeasurementOnOff, 250	OTF2_EvtReaderCallback_RmaCollectiveBegin, 268
OTF2_EvtReaderCallback_Metric, 250	OTF2_EvtReaderCallback_RmaCollectiveEnd, 268
OTF2_EvtReaderCallback_MpiCollectiveBegin, 251	OTF2_EvtReaderCallback_RmaCollectiveEnd, 268
OTF2_EvtReaderCallback_MpiCollectiveEnd, 252	OTF2_EvtReaderCallback_RmaGet, 269
OTF2_EvtReaderCallback_MpiIrecv, 253	OTF2_EvtReaderCallback_RmaGroupSync, 270
OTF2_EvtReaderCallback_MpiIrecvRequest, 254	OTF2_EvtReaderCallback_RmaOpCompleteBlocking, 271
OTF2_EvtReaderCallback_MpiIsend, 254	OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking, 271
OTF2_EvtReaderCallback_MpiIsendComplete, 255	OTF2_EvtReaderCallback_RmaOpCompleteRemote, 272
OTF2_EvtReaderCallback_MpiRecv, 256	OTF2_EvtReaderCallback_RmaOpTest, 273
OTF2_EvtReaderCallback_MpiRequestCancelled, 257	OTF2_EvtReaderCallback_RmaPut, 274
OTF2_EvtReaderCallback_MpiRequestTest, 257	OTF2_EvtReaderCallback_RmaReleaseLock, 274
OTF2_EvtReaderCallback_MpiSend, 258	OTF2_EvtReaderCallback_RmaRequestLock, 275
OTF2_EvtReaderCallback_OmpAcquireLock, 259	OTF2_EvtReaderCallback_RmaSync, 276
OTF2_EvtReaderCallback_OmpFork, 259	OTF2_EvtReaderCallback_RmaTryLock, 277
OTF2_EvtReaderCallback_OmpJoin, 260	OTF2_EvtReaderCallback_RmaWaitChange, 278
OTF2_EvtReaderCallback_OmpReleaseLock, 261	OTF2_EvtReaderCallback_RmaWinCreate, 278
OTF2_EvtReaderCallback_OmpTaskComplete, 262	OTF2_EvtReaderCallback_RmaWinDestroy, 279
OTF2_EvtReaderCallback_OmpTaskCreate, 262	OTF2_EvtReaderCallback_ThreadAcquireLock, 280
OTF2_EvtReaderCallback_OmpTaskSwitch, 263	

INDEX

OTF2_EvtReaderCallback_ThreadFork, [280](#) OTF2_EvtReaderCallbacks_SetMpiRecvCallback, [292](#)
 OTF2_EvtReaderCallback_ThreadJoin, [281](#) OTF2_EvtReaderCallbacks_SetMpiRequestCancelledCallback, [293](#)
 OTF2_EvtReaderCallback_ThreadRelease, [282](#) OTF2_EvtReaderCallbacks_SetMpiRequestTestCallback, [293](#)
 OTF2_EvtReaderCallback_ThreadTaskComplete, [282](#) OTF2_EvtReaderCallbacks_SetMpiSendCallback, [294](#)
 OTF2_EvtReaderCallback_ThreadTaskCreate, [283](#) OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback, [294](#)
 OTF2_EvtReaderCallback_ThreadTaskSwitch, [284](#) OTF2_EvtReaderCallbacks_SetOmpForkCallback, [295](#)
 OTF2_EvtReaderCallback_ThreadTeamBegin, [285](#) OTF2_EvtReaderCallbacks_SetOmpJoinCallback, [295](#)
 OTF2_EvtReaderCallback_ThreadTeamEnd, [285](#) OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback, [296](#)
 OTF2_EvtReaderCallback_Unknown, [286](#) OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback, [296](#)
 OTF2_EvtReaderCallbacks_Clear, [287](#) OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback, [297](#)
 OTF2_EvtReaderCallbacks_Delete, [287](#) OTF2_EvtReaderCallbacks_SetOmpTaskSwitchCallback, [297](#)
 OTF2_EvtReaderCallbacks_New, [287](#) OTF2_EvtReaderCallbacks_SetParameterIntCallback, [298](#)
 OTF2_EvtReaderCallbacks_SetBufferFlushCallback, [287](#) OTF2_EvtReaderCallbacks_SetParameterStringCallback, [298](#)
 OTF2_EvtReaderCallbacks_SetEnterCallback, [288](#) OTF2_EvtReaderCallbacks_SetParameterUnsignedIntCallback, [299](#)
 OTF2_EvtReaderCallbacks_SetLeaveCallback, [288](#) OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback, [299](#)
 OTF2_EvtReaderCallbacks_SetMeasurementCallback, [289](#) OTF2_EvtReaderCallbacks_SetRmaAtomicCallback, [300](#)
 OTF2_EvtReaderCallbacks_SetMetricCallback, [289](#) OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback, [300](#)
 OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback, [290](#) OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback, [301](#)
 OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback, [290](#) OTF2_EvtReaderCallbacks_SetRmaGetCallback, [301](#)
 OTF2_EvtReaderCallbacks_SetMpiIrecvCallback, [291](#) OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback, [302](#)
 OTF2_EvtReaderCallbacks_SetMpiIrecvRequestCallback, [291](#) OTF2_EvtReaderCallbacks_SetRmaOpCompleteBlockingCallback, [302](#)
 OTF2_EvtReaderCallbacks_SetMpiIsendCallback, [292](#) OTF2_EvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback, [303](#)
 OTF2_EvtReaderCallbacks_SetMpiIsendRequestCallback, [292](#)

INDEX

OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_New	OTF2_EvtReaderCallbacks.h, 287
304	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetBufferFlushCallback	OTF2_EvtReaderCallbacks.h, 287
304	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetEnterCallback	OTF2_EvtReaderCallbacks.h, 288
305	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetLeaveCallback	OTF2_EvtReaderCallbacks.h, 288
305	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetMeasurementOnOffCallback	OTF2_EvtReaderCallbacks.h, 289
305	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetMetricCallback	OTF2_EvtReaderCallbacks.h, 289
306	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback	OTF2_EvtReaderCallbacks.h, 290
306	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback	OTF2_EvtReaderCallbacks.h, 290
307	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetMpiIrecvCallback	OTF2_EvtReaderCallbacks.h, 291
307	
OTF2_EvtReaderCallbacks_SetRmaOpOTF2_EvtReaderCallbacks_SetMpiIrecvRequestCallback	OTF2_EvtReaderCallbacks.h, 291
308	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetMpiIsendCallback	OTF2_EvtReaderCallbacks.h, 292
308	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetMpiIsendCompleteCallback	OTF2_EvtReaderCallbacks.h, 292
309	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetMpiRecvCallback	OTF2_EvtReaderCallbacks.h, 292
309	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetMpiRequestCancelledCallback	OTF2_EvtReaderCallbacks.h, 293
310	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetMpiRequestTestCallback	OTF2_EvtReaderCallbacks.h, 293
310	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetMpiSendCallback	OTF2_EvtReaderCallbacks.h, 294
311	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback	OTF2_EvtReaderCallbacks.h, 294
311	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetOmpForkCallback	OTF2_EvtReaderCallbacks.h, 295
312	
OTF2_EvtReaderCallbacks_SetThreadOTF2_EvtReaderCallbacks_SetOmpJoinCallback	OTF2_EvtReaderCallbacks.h, 295
312	
OTF2_EvtReaderCallbacks_SetUnknownOTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback	OTF2_EvtReaderCallbacks.h, 296
313	
OTF2_EvtReaderCallbacks_Clear	OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback
OTF2_EvtReaderCallbacks.h, 287	OTF2_EvtReaderCallbacks.h, 296
OTF2_EvtReaderCallbacks_Delete	OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback
OTF2_EvtReaderCallbacks.h, 287	OTF2_EvtReaderCallbacks.h, 297

OTF2_EvtReaderCallbacks_SetOmpTaskSetCallback, 297
 OTF2_EvtReaderCallbacks_SetThreadAcquireLockCallback, 308
 OTF2_EvtReaderCallbacks_SetParameterCallback, 298
 OTF2_EvtReaderCallbacks_SetThreadForkCallback, 309
 OTF2_EvtReaderCallbacks_SetParameterCallback, 298
 OTF2_EvtReaderCallbacks_SetThreadJoinCallback, 309
 OTF2_EvtReaderCallbacks_SetParameterCallback, 299
 OTF2_EvtReaderCallbacks_SetThreadReleaseLockCallback, 310
 OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback, 299
 OTF2_EvtReaderCallbacks_SetThreadTaskCompleteCallback, 310
 OTF2_EvtReaderCallbacks_SetRmaAtomicLockCallback, 300
 OTF2_EvtReaderCallbacks_SetThreadTaskCreateCallback, 311
 OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback, 300
 OTF2_EvtReaderCallbacks_SetThreadTaskSwitchCallback, 311
 OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback, 301
 OTF2_EvtReaderCallbacks_SetThreadTeamBeginCallback, 312
 OTF2_EvtReaderCallbacks_SetRmaGetCallback, 301
 OTF2_EvtReaderCallbacks_SetThreadTeamEndCallback, 312
 OTF2_EvtReaderCallbacks_SetRmaGroupCallback, 302
 OTF2_EvtReaderCallbacks_SetUnknownCallback, 313
 OTF2_EvtReaderCallbacks_SetRmaOpCompleteCallback, 302
 OTF2_EvtWriter_BufferFlush, 320
 OTF2_EvtReaderCallbacks_SetRmaOpCompleteCallback, 303
 OTF2_EvtWriter_RewindPoint, 321
 OTF2_EvtReaderCallbacks_SetRmaOpCompleteCallback, 304
 OTF2_EvtWriter_Enter, 321
 OTF2_EvtReaderCallbacks_SetRmaOpTestCallback, 304
 OTF2_EvtWriter_GetLocationID, 322
 OTF2_EvtReaderCallbacks_SetRmaPutCallback, 305
 OTF2_EvtWriter_GetUserData, 323
 OTF2_EvtReaderCallbacks_SetRmaReleaseLockCallback, 305
 OTF2_EvtWriter_Leave, 323
 OTF2_EvtReaderCallbacks_SetRmaRequestLockCallback, 305
 OTF2_EvtWriter_MeasurementOnOff, 323
 OTF2_EvtReaderCallbacks_SetRmaSyncCallback, 306
 OTF2_EvtWriter_Metric, 324
 OTF2_EvtReaderCallbacks_SetRmaTryLockCallback, 306
 OTF2_EvtWriter_MpiCollectiveBegin, 325
 OTF2_EvtReaderCallbacks_SetRmaWaitChangeCallback, 307
 OTF2_EvtWriter_MpiCollectiveEnd, 325
 OTF2_EvtReaderCallbacks_SetRmaWinCreateCallback, 307
 OTF2_EvtWriter_MpiIrecv, 326
 OTF2_EvtReaderCallbacks_SetRmaWinDestroyCallback, 308
 OTF2_EvtWriter_MpiIrecvRequest, 327
 OTF2_EvtReaderCallbacks_SetRmaWinCreateCallback, 307
 OTF2_EvtWriter_MpiIsend, 327
 OTF2_EvtReaderCallbacks_SetRmaWinDestroyCallback, 308
 OTF2_EvtWriter_MpiIsendComplete, 328
 OTF2_EvtReaderCallbacks_SetRmaWinDestroyCallback, 308
 OTF2_EvtWriter_MpiRecv, 328

INDEX

OTF2_EvtWriter_MpiRequestCancelled, [329](#)
OTF2_EvtWriter_MpiRequestTest, [330](#)
OTF2_EvtWriter_MpiSend, [330](#)
OTF2_EvtWriter_OmpAcquireLock, [331](#)
OTF2_EvtWriter_OmpFork, [332](#)
OTF2_EvtWriter_OmpJoin, [332](#)
OTF2_EvtWriter_OmpReleaseLock, [333](#)
OTF2_EvtWriter_OmpTaskComplete, [334](#)
OTF2_EvtWriter_OmpTaskCreate, [334](#)
OTF2_EvtWriter_OmpTaskSwitch, [335](#)
OTF2_EvtWriter_ParameterInt, [336](#)
OTF2_EvtWriter_ParameterString, [336](#)
OTF2_EvtWriter_ParameterUnsignedInt, [337](#)
OTF2_EvtWriter_Rewind, [338](#)
OTF2_EvtWriter_RmaAcquireLock, [338](#)
OTF2_EvtWriter_RmaAtomic, [339](#)
OTF2_EvtWriter_RmaCollectiveBegin, [340](#)
OTF2_EvtWriter_RmaCollectiveEnd, [340](#)
OTF2_EvtWriter_RmaGet, [341](#)
OTF2_EvtWriter_RmaGroupSync, [341](#)
OTF2_EvtWriter_RmaOpCompleteBlocking, [342](#)
OTF2_EvtWriter_RmaOpCompleteNonBlocking, [343](#)
OTF2_EvtWriter_RmaOpCompleteRemote, [343](#)
OTF2_EvtWriter_RmaOpTest, [344](#)
OTF2_EvtWriter_RmaPut, [345](#)
OTF2_EvtWriter_RmaReleaseLock, [345](#)
OTF2_EvtWriter_RmaRequestLock, [346](#)
OTF2_EvtWriter_RmaSync, [347](#)
OTF2_EvtWriter_RmaTryLock, [347](#)
OTF2_EvtWriter_RmaWaitChange, [348](#)
OTF2_EvtWriter_RmaWinCreate, [348](#)
OTF2_EvtWriter_RmaWinDestroy, [349](#)
OTF2_EvtWriter_SetLocationID, [350](#)
OTF2_EvtWriter_SetUserData, [350](#)
OTF2_EvtWriter_StoreRewindPoint, [350](#)
OTF2_EvtWriter_ThreadAcquireLock, [351](#)
OTF2_EvtWriter_ThreadFork, [351](#)
OTF2_EvtWriter_ThreadJoin, [352](#)
OTF2_EvtWriter_ThreadReleaseLock, [352](#)
OTF2_EvtWriter_ThreadTaskComplete, [353](#)
OTF2_EvtWriter_ThreadTaskCreate, [354](#)
OTF2_EvtWriter_ThreadTaskSwitch, [354](#)
OTF2_EvtWriter_ThreadTeamBegin, [355](#)
OTF2_EvtWriter_ThreadTeamEnd, [356](#)
OTF2_EvtWriter_BufferFlush
OTF2_EvtWriter.h, [320](#)
OTF2_EvtWriter_ClearRewindPoint
OTF2_EvtWriter.h, [321](#)
OTF2_EvtWriter_Enter
OTF2_EvtWriter.h, [321](#)
OTF2_EvtWriter_GetLocationID
OTF2_EvtWriter.h, [322](#)
OTF2_EvtWriter_GetNumberOfEvents
OTF2_EvtWriter.h, [322](#)
OTF2_EvtWriter_GetUserData
OTF2_EvtWriter.h, [323](#)
OTF2_EvtWriter_Leave
OTF2_EvtWriter.h, [323](#)
OTF2_EvtWriter_MeasurementOnOff
OTF2_EvtWriter.h, [323](#)
OTF2_EvtWriter_Metric
OTF2_EvtWriter.h, [324](#)
OTF2_EvtWriter_MpiCollectiveBegin

OTF2_EvtWriter.h, 325	OTF2_EvtWriter.h, 339
OTF2_EvtWriter_MpiCollectiveEnd	OTF2_EvtWriter_RmaCollectiveBegin
OTF2_EvtWriter.h, 325	OTF2_EvtWriter.h, 340
OTF2_EvtWriter_MpiIrecv	OTF2_EvtWriter_RmaCollectiveEnd
OTF2_EvtWriter.h, 326	OTF2_EvtWriter.h, 340
OTF2_EvtWriter_MpiIrecvRequest	OTF2_EvtWriter_RmaGet
OTF2_EvtWriter.h, 327	OTF2_EvtWriter.h, 341
OTF2_EvtWriter_MpiIsend	OTF2_EvtWriter_RmaGroupSync
OTF2_EvtWriter.h, 327	OTF2_EvtWriter.h, 341
OTF2_EvtWriter_MpiIsendComplete	OTF2_EvtWriter_RmaOpCompleteBlocking
OTF2_EvtWriter.h, 328	OTF2_EvtWriter.h, 342
OTF2_EvtWriter_MpiRecv	OTF2_EvtWriter_RmaOpCompleteNonBlocking
OTF2_EvtWriter.h, 328	OTF2_EvtWriter.h, 343
OTF2_EvtWriter_MpiRequestCancelled	OTF2_EvtWriter_RmaOpCompleteRemote
OTF2_EvtWriter.h, 329	OTF2_EvtWriter.h, 343
OTF2_EvtWriter_MpiRequestTest	OTF2_EvtWriter_RmaOpTest
OTF2_EvtWriter.h, 330	OTF2_EvtWriter.h, 344
OTF2_EvtWriter_MpiSend	OTF2_EvtWriter_RmaPut
OTF2_EvtWriter.h, 330	OTF2_EvtWriter.h, 345
OTF2_EvtWriter_OmpAcquireLock	OTF2_EvtWriter_RmaReleaseLock
OTF2_EvtWriter.h, 331	OTF2_EvtWriter.h, 345
OTF2_EvtWriter_OmpFork	OTF2_EvtWriter_RmaRequestLock
OTF2_EvtWriter.h, 332	OTF2_EvtWriter.h, 346
OTF2_EvtWriter_OmpJoin	OTF2_EvtWriter_RmaSync
OTF2_EvtWriter.h, 332	OTF2_EvtWriter.h, 347
OTF2_EvtWriter_OmpReleaseLock	OTF2_EvtWriter_RmaTryLock
OTF2_EvtWriter.h, 333	OTF2_EvtWriter.h, 347
OTF2_EvtWriter_OmpTaskComplete	OTF2_EvtWriter_RmaWaitChange
OTF2_EvtWriter.h, 334	OTF2_EvtWriter.h, 348
OTF2_EvtWriter_OmpTaskCreate	OTF2_EvtWriter_RmaWinCreate
OTF2_EvtWriter.h, 334	OTF2_EvtWriter.h, 348
OTF2_EvtWriter_OmpTaskSwitch	OTF2_EvtWriter_RmaWinDestroy
OTF2_EvtWriter.h, 335	OTF2_EvtWriter.h, 349
OTF2_EvtWriter_ParameterInt	OTF2_EvtWriter_SetLocationID
OTF2_EvtWriter.h, 336	OTF2_EvtWriter.h, 350
OTF2_EvtWriter_ParameterString	OTF2_EvtWriter_SetUserData
OTF2_EvtWriter.h, 336	OTF2_EvtWriter.h, 350
OTF2_EvtWriter_ParameterUnsignedInt	OTF2_EvtWriter_StoreRewindPoint
OTF2_EvtWriter.h, 337	OTF2_EvtWriter.h, 350
OTF2_EvtWriter_Rewind	OTF2_EvtWriter_ThreadAcquireLock
OTF2_EvtWriter.h, 338	OTF2_EvtWriter.h, 351
OTF2_EvtWriter_RmaAcquireLock	OTF2_EvtWriter_ThreadFork
OTF2_EvtWriter.h, 338	OTF2_EvtWriter.h, 351
OTF2_EvtWriter_RmaAtomic	OTF2_EvtWriter_ThreadJoin

INDEX

- OTF2_EvtWriter.h, [352](#)
- OTF2_EvtWriter_ThreadReleaseLock
 - OTF2_EvtWriter.h, [352](#)
- OTF2_EvtWriter_ThreadTaskComplete
 - OTF2_EvtWriter.h, [353](#)
- OTF2_EvtWriter_ThreadTaskCreate
 - OTF2_EvtWriter.h, [354](#)
- OTF2_EvtWriter_ThreadTaskSwitch
 - OTF2_EvtWriter.h, [354](#)
- OTF2_EvtWriter_ThreadTeamBegin
 - OTF2_EvtWriter.h, [355](#)
- OTF2_EvtWriter_ThreadTeamEnd
 - OTF2_EvtWriter.h, [356](#)
- OTF2_FileMode_enum
 - OTF2_GeneralDefinitions.h, [364](#)
- OTF2_FileSionCallbacks, [91](#)
- OTF2_FileSionClose
 - OTF2_Callbacks.h, [152](#)
- OTF2_FileSionGetRank
 - OTF2_Callbacks.h, [152](#)
- OTF2_FileSionOpen
 - OTF2_Callbacks.h, [152](#)
- OTF2_FileSubstrate_enum
 - OTF2_GeneralDefinitions.h, [364](#)
- OTF2_FileType_enum
 - OTF2_GeneralDefinitions.h, [364](#)
- OTF2_FlushCallbacks, [91](#)
- OTF2_FlushType_enum
 - OTF2_GeneralDefinitions.h, [365](#)
- OTF2_GeneralDefinitions.h, [356](#)
 - OTF2_CallbackCode, [363](#)
 - OTF2_Compression_enum, [363](#)
 - OTF2_FileMode_enum, [364](#)
 - OTF2_FileSubstrate_enum, [364](#)
 - OTF2_FileType_enum, [364](#)
 - OTF2_FlushType_enum, [365](#)
 - OTF2_MappingType_enum, [365](#)
 - OTF2_Paradigm_enum, [366](#)
 - OTF2_ThumbnailType_enum, [366](#)
 - OTF2_Type_enum, [367](#)
 - OTF2_UNDEFINED_TYPE, [363](#)
- OTF2_GlobalDefReader.h, [367](#)
 - OTF2_GlobalDefReader_ReadDefinitions, [368](#)
 - OTF2_GlobalDefReader_SetCallbacks, [369](#)
- OTF2_GlobalDefReader_ReadDefinitions
 - OTF2_GlobalDefReader.h, [368](#)
- OTF2_GlobalDefReader_SetCallbacks
 - OTF2_GlobalDefReader.h, [369](#)
- OTF2_GlobalDefReaderCallback_Attribute
 - OTF2_GlobalDefReaderCallbacks.h, [375](#)
- OTF2_GlobalDefReaderCallback_Callpath
 - OTF2_GlobalDefReaderCallbacks.h, [375](#)
- OTF2_GlobalDefReaderCallback_Callsite
 - OTF2_GlobalDefReaderCallbacks.h, [376](#)
- OTF2_GlobalDefReaderCallback_ClockProperties
 - OTF2_GlobalDefReaderCallbacks.h, [376](#)
- OTF2_GlobalDefReaderCallback_Comm
 - OTF2_GlobalDefReaderCallbacks.h, [377](#)
- OTF2_GlobalDefReaderCallback_Group
 - OTF2_GlobalDefReaderCallbacks.h, [378](#)
- OTF2_GlobalDefReaderCallback_Location
 - OTF2_GlobalDefReaderCallbacks.h, [378](#)
- OTF2_GlobalDefReaderCallback_LocationGroup
 - OTF2_GlobalDefReaderCallbacks.h, [379](#)
- OTF2_GlobalDefReaderCallback_MetricClass
 - OTF2_GlobalDefReaderCallbacks.h, [380](#)
- OTF2_GlobalDefReaderCallback_MetricClassRecorder
 - OTF2_GlobalDefReaderCallbacks.h, [380](#)
- OTF2_GlobalDefReaderCallback_MetricInstance
 - OTF2_GlobalDefReaderCallbacks.h, [381](#)
- OTF2_GlobalDefReaderCallback_MetricMember
 - OTF2_GlobalDefReaderCallbacks.h, [382](#)
- OTF2_GlobalDefReaderCallback_Parameter

- OTF2_GlobalDefReaderCallbacks.h, 383
- OTF2_GlobalDefReaderCallback_Region
 - OTF2_GlobalDefReaderCallbacks.h, 383
- OTF2_GlobalDefReaderCallback_RmaWin
 - OTF2_GlobalDefReaderCallbacks.h, 384
- OTF2_GlobalDefReaderCallback_String
 - OTF2_GlobalDefReaderCallbacks.h, 385
- OTF2_GlobalDefReaderCallback_SystemTreeNodeString
 - OTF2_GlobalDefReaderCallbacks.h, 385
- OTF2_GlobalDefReaderCallback_SystemTreeNodeDomain
 - OTF2_GlobalDefReaderCallbacks.h, 386
- OTF2_GlobalDefReaderCallback_SystemTreeNodeProperty
 - OTF2_GlobalDefReaderCallbacks.h, 386
- OTF2_GlobalDefReaderCallback_Unknown
 - OTF2_GlobalDefReaderCallbacks.h, 387
- OTF2_GlobalDefReaderCallbacks.h, 369
 - OTF2_GlobalDefReaderCallback_
 - Attribute, 375
 - Callpath, 375
 - Callsite, 376
 - ClockProperties, 376
 - Comm, 377
 - Group, 378
 - Location, 378
 - LocationGroup, 379
 - MetricClass, 380
 - MetricClassRecorder, 380
 - OTF2_GlobalDefReaderCallback_
 - MetricInstance, 381
 - MetricMember, 382
 - Parameter, 383
 - Region, 383
 - RmaWin, 384
 - String, 385
 - SystemTreeNode, 385
 - SystemTreeNodeDomain, 386
 - SystemTreeNodeProperty, 386
 - Unknown, 387
 - Clear, 388
 - Delete, 388
 - New, 388
 - SetAttributeCallback, 388
 - SetCallpathCallback, 389
 - SetCallsiteCallback, 389
 - SetClockPropertiesCallback, 390
 - SetCommCallback, 390
 - SetGroupCallback, 391
 - SetLocationCallback, 391
 - SetLocationGroupCallback, 392
 - SetMetricClassCallback, 392

INDEX

OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks.h,
SetMetricClassRecorderCallback, 390
393 OTF2_GlobalDefReaderCallbacks_SetCommCallback
OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks.h,
SetMetricInstanceCallback, 393 390
OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetGroupCallback
SetMetricMemberCallback, 394 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_- 391
SetParameterCallback, 394 OTF2_GlobalDefReaderCallbacks_SetLocationCallback
OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks.h,
SetRegionCallback, 395 391
OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetLocationGroupCallback
SetRmaWinCallback, 395 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_- 392
SetStringCallback, 396 OTF2_GlobalDefReaderCallbacks_SetMetricClassCallback
OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks.h,
SetSystemTreeNodeCallback, 396 392
OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetMetricClassRecorderCallback
SetSystemTreeNodeDomainCallback, OTF2_GlobalDefReaderCallbacks.h,
397 393
OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetMetricInstanceCallback
SetSystemTreeNodePropertyCallback, OTF2_GlobalDefReaderCallbacks.h,
398 393
OTF2_GlobalDefReaderCallbacks_- OTF2_GlobalDefReaderCallbacks_SetMetricMemberCallback
SetUnknownCallback, 398 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_Clear 394
OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetParameterCallback
388 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_Delete 394
OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetRegionCallback
388 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_New 395
OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetRmaWinCallback
388 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_SetAttributeCallback 395
OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetStringCallback
388 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_SetCallpathCallback 396
OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeCallback
389 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_SetCallsiteCallback 394
OTF2_GlobalDefReaderCallbacks.h, OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeDomainCallback
389 OTF2_GlobalDefReaderCallbacks.h,
OTF2_GlobalDefReaderCallbacks_SetClockPropertyCallback 397

INDEX

OTF2_GlobalDefReaderCallbacks_SetSystemTreeDomain, [OTF2_GlobalDefWriter_WriteSystemTreeNode](#),
 OTF2_GlobalDefReaderCallbacks.h, [413](#)
 OTF2_GlobalDefReaderCallbacks_SetUnknownCallback, [414](#)
 OTF2_GlobalDefReaderCallbacks.h, [398](#) OTF2_GlobalDefWriter_WriteSystemTreeNodeDomain,
 OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty,
 OTF2_GlobalDefWriter.h, [399](#) [414](#)
 OTF2_GlobalDefWriter_GetNumberOfDefinitions
 OTF2_GlobalDefWriter_GetNumberOfLocations
 OTF2_GlobalDefWriter_GetNumberOfLocations, [402](#)
 OTF2_GlobalDefWriter_GetNumberOfLocations, [402](#)
 OTF2_GlobalDefWriter_WriteAttribute, [403](#)
 OTF2_GlobalDefWriter_WriteAttribute, [403](#)
 OTF2_GlobalDefWriter_WriteCallpath, [403](#)
 OTF2_GlobalDefWriter_WriteCallpath, [403](#)
 OTF2_GlobalDefWriter_WriteCallsite, [404](#)
 OTF2_GlobalDefWriter_WriteCallsite, [404](#)
 OTF2_GlobalDefWriter_WriteClockProperties, [404](#)
 OTF2_GlobalDefWriter_WriteClockProperties, [404](#)
 OTF2_GlobalDefWriter_WriteComm, [405](#)
 OTF2_GlobalDefWriter_WriteComm, [405](#)
 OTF2_GlobalDefWriter_WriteGroup, [406](#)
 OTF2_GlobalDefWriter_WriteGroup, [406](#)
 OTF2_GlobalDefWriter_WriteLocation, [406](#)
 OTF2_GlobalDefWriter_WriteLocation, [406](#)
 OTF2_GlobalDefWriter_WriteLocationGroup, [407](#)
 OTF2_GlobalDefWriter_WriteLocationGroup, [407](#)
 OTF2_GlobalDefWriter_WriteMetricClass, [408](#)
 OTF2_GlobalDefWriter_WriteMetricClass, [408](#)
 OTF2_GlobalDefWriter_WriteMetricClassRecorder, [408](#)
 OTF2_GlobalDefWriter_WriteMetricClassRecorder, [408](#)
 OTF2_GlobalDefWriter_WriteMetricInstance, [409](#)
 OTF2_GlobalDefWriter_WriteMetricInstance, [409](#)
 OTF2_GlobalDefWriter_WriteMetricMember, [410](#)
 OTF2_GlobalDefWriter_WriteMetricMember, [410](#)
 OTF2_GlobalDefWriter_WriteParameter, [411](#)
 OTF2_GlobalDefWriter_WriteParameter, [411](#)
 OTF2_GlobalDefWriter_WriteRegion, [411](#)
 OTF2_GlobalDefWriter_WriteRegion, [411](#)
 OTF2_GlobalDefWriter_WriteRmaWin, [412](#)
 OTF2_GlobalDefWriter_WriteRmaWin, [412](#)
 OTF2_GlobalDefWriter_WriteString, [413](#)
 OTF2_GlobalDefWriter_WriteString, [413](#)
 OTF2_GlobalDefWriter_WriteSystemTreeNode, [413](#)

INDEX

OTF2_GlobalDefWriter_WriteSystemTreeNode, OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalDefWriter.h, [414](#) [435](#)
 OTF2_GlobalDefWriter_WriteSystemTreeNodeCopy, OTF2_GlobalEvtReaderCallback_MpiIrecvRequest
 OTF2_GlobalDefWriter.h, [414](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReader.h, [415](#) [435](#)
 OTF2_GlobalEvtReader_HasEvent, OTF2_GlobalEvtReaderCallback_MpiIsend
[416](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReader_ReadEvent, [436](#)
[416](#) OTF2_GlobalEvtReaderCallback_MpiIsendComplete
 OTF2_GlobalEvtReader_ReadEvents, OTF2_GlobalEvtReaderCallbacks.h,
[416](#) [437](#)
 OTF2_GlobalEvtReader_SetCallbacks, OTF2_GlobalEvtReaderCallback_MpiRecv
[417](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReader_HasEvent [437](#)
 OTF2_GlobalEvtReader.h, [416](#) OTF2_GlobalEvtReaderCallback_MpiRequestCancelled
 OTF2_GlobalEvtReader_ReadEvent OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReader.h, [416](#) [438](#)
 OTF2_GlobalEvtReader_ReadEvents OTF2_GlobalEvtReaderCallback_MpiRequestTest
 OTF2_GlobalEvtReader.h, [416](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReader_SetCallbacks [439](#)
 OTF2_GlobalEvtReader.h, [417](#) OTF2_GlobalEvtReaderCallback_MpiSend
 OTF2_GlobalEvtReaderCallback_BufferFlush OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReaderCallbacks.h, [439](#)
 OTF2_GlobalEvtReaderCallback_Enter OTF2_GlobalEvtReaderCallback_OmpAcquireLock
 OTF2_GlobalEvtReaderCallbacks.h, [440](#)
 OTF2_GlobalEvtReaderCallback_Leave OTF2_GlobalEvtReaderCallback_OmpFork
 OTF2_GlobalEvtReaderCallbacks.h, [441](#)
 OTF2_GlobalEvtReaderCallback_MeasurementOnOff OTF2_GlobalEvtReaderCallback_OmpJoin
 OTF2_GlobalEvtReaderCallbacks.h, [442](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReaderCallback_Metric OTF2_GlobalEvtReaderCallback_OmpReleaseLock
 OTF2_GlobalEvtReaderCallbacks.h, [442](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReaderCallback_MpiCollectiveBegin OTF2_GlobalEvtReaderCallback_OmpTaskComplete
 OTF2_GlobalEvtReaderCallbacks.h, [443](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReaderCallback_MpiCollectiveEnd OTF2_GlobalEvtReaderCallback_OmpTaskCreate
 OTF2_GlobalEvtReaderCallbacks.h, [444](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReaderCallback_MpiIrecv OTF2_GlobalEvtReaderCallback_OmpTaskSwitch
 OTF2_GlobalEvtReaderCallbacks.h, [444](#) OTF2_GlobalEvtReaderCallbacks.h,
 OTF2_GlobalEvtReaderCallback_MpiIrecv [444](#)

INDEX

OTF2_GlobalEvtReaderCallback_ParameterInOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 455
445 OTF2_GlobalEvtReaderCallback_RmaRequestLock
OTF2_GlobalEvtReaderCallback_ParameterStorageOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 455
445 OTF2_GlobalEvtReaderCallback_RmaSync
OTF2_GlobalEvtReaderCallback_ParameterUnsignedIntOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 456
446 OTF2_GlobalEvtReaderCallback_RmaTryLock
OTF2_GlobalEvtReaderCallback_RmaAcquireLockOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 457
447 OTF2_GlobalEvtReaderCallback_RmaWaitChange
OTF2_GlobalEvtReaderCallback_RmaAtomicOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 457
448 OTF2_GlobalEvtReaderCallback_RmaWinCreate
OTF2_GlobalEvtReaderCallback_RmaCollectiveBeginOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 458
448 OTF2_GlobalEvtReaderCallback_RmaWinDestroy
OTF2_GlobalEvtReaderCallback_RmaCollectiveEndOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 459
449 OTF2_GlobalEvtReaderCallback_ThreadAcquireLock
OTF2_GlobalEvtReaderCallback_RmaGetOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 459
450 OTF2_GlobalEvtReaderCallback_ThreadFork
OTF2_GlobalEvtReaderCallback_RmaGroupSyncOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 460
451 OTF2_GlobalEvtReaderCallback_ThreadJoin
OTF2_GlobalEvtReaderCallback_RmaOpCompleteBlockingOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 461
451 OTF2_GlobalEvtReaderCallback_ThreadReleaseLock
OTF2_GlobalEvtReaderCallback_RmaOpCompleteNonBlockingOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 461
452 OTF2_GlobalEvtReaderCallback_ThreadTaskComplete
OTF2_GlobalEvtReaderCallback_RmaOpCompleteRemoteOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 462
453 OTF2_GlobalEvtReaderCallback_ThreadTaskCreate
OTF2_GlobalEvtReaderCallback_RmaOpTestOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 463
453 OTF2_GlobalEvtReaderCallback_ThreadTaskSwitch
OTF2_GlobalEvtReaderCallback_RmaPutOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 463
454 OTF2_GlobalEvtReaderCallback_ThreadTeamBegin
OTF2_GlobalEvtReaderCallback_RmaReleaseLockOTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks.h, 464
454

INDEX

- OTF2_GlobalEvtReaderCallback_ThreadPool, 442
- OTF2_GlobalEvtReaderCallbacks.h, 465
- OTF2_GlobalEvtReaderCallback_Unknown, 465
- OTF2_GlobalEvtReaderCallbacks.h, 418
- OTF2_GlobalEvtReaderCallback_-BufferFlush, 430
- OTF2_GlobalEvtReaderCallback_-Enter, 431
- OTF2_GlobalEvtReaderCallback_-Leave, 431
- OTF2_GlobalEvtReaderCallback_-MeasurementOnOff, 432
- OTF2_GlobalEvtReaderCallback_-Metric, 432
- OTF2_GlobalEvtReaderCallback_-MpiCollectiveBegin, 433
- OTF2_GlobalEvtReaderCallback_-MpiCollectiveEnd, 434
- OTF2_GlobalEvtReaderCallback_-MpiIrecv, 435
- OTF2_GlobalEvtReaderCallback_-MpiIrecvRequest, 435
- OTF2_GlobalEvtReaderCallback_-MpiIsend, 436
- OTF2_GlobalEvtReaderCallback_-MpiIsendComplete, 437
- OTF2_GlobalEvtReaderCallback_-MpiRecv, 437
- OTF2_GlobalEvtReaderCallback_-MpiRequestCancelled, 438
- OTF2_GlobalEvtReaderCallback_-MpiRequestTest, 439
- OTF2_GlobalEvtReaderCallback_-MpiSend, 439
- OTF2_GlobalEvtReaderCallback_-OmpAcquireLock, 440
- OTF2_GlobalEvtReaderCallback_-OmpFork, 441
- OTF2_GlobalEvtReaderCallback_-OmpJoin, 442
- OTF2_GlobalEvtReaderCallback_-OmpReleaseLock, 442
- OTF2_GlobalEvtReaderCallback_-OmpTaskComplete, 443
- OTF2_GlobalEvtReaderCallback_-OmpTaskCreate, 444
- OTF2_GlobalEvtReaderCallback_-OmpTaskSwitch, 444
- OTF2_GlobalEvtReaderCallback_-ParameterInt, 445
- OTF2_GlobalEvtReaderCallback_-ParameterString, 445
- OTF2_GlobalEvtReaderCallback_-ParameterUnsignedInt, 446
- OTF2_GlobalEvtReaderCallback_-RmaAcquireLock, 447
- OTF2_GlobalEvtReaderCallback_-RmaAtomic, 448
- OTF2_GlobalEvtReaderCallback_-RmaCollectiveBegin, 448
- OTF2_GlobalEvtReaderCallback_-RmaCollectiveEnd, 449
- OTF2_GlobalEvtReaderCallback_-RmaGet, 450
- OTF2_GlobalEvtReaderCallback_-RmaGroupSync, 451
- OTF2_GlobalEvtReaderCallback_-RmaOpCompleteBlocking, 451
- OTF2_GlobalEvtReaderCallback_-RmaOpCompleteNonBlocking, 452
- OTF2_GlobalEvtReaderCallback_-RmaOpCompleteRemote, 453
- OTF2_GlobalEvtReaderCallback_-RmaOpTest, 453
- OTF2_GlobalEvtReaderCallback_-RmaPut, 454
- OTF2_GlobalEvtReaderCallback_-RmaReleaseLock, 455
- OTF2_GlobalEvtReaderCallback_-RmaRequestLock, 455
- OTF2_GlobalEvtReaderCallback_-RmaSync, 456

-
- OTF2_GlobalEvtReaderCallback_-
RmaTryLock, [457](#)
 - OTF2_GlobalEvtReaderCallback_-
RmaWaitChange, [457](#)
 - OTF2_GlobalEvtReaderCallback_-
RmaWinCreate, [458](#)
 - OTF2_GlobalEvtReaderCallback_-
RmaWinDestroy, [459](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadAcquireLock, [459](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadFork, [460](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadJoin, [461](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadReleaseLock, [461](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadTaskComplete, [462](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadTaskCreate, [463](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadTaskSwitch, [463](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadTeamBegin, [464](#)
 - OTF2_GlobalEvtReaderCallback_-
ThreadTeamEnd, [465](#)
 - OTF2_GlobalEvtReaderCallback_-
Unknown, [465](#)
 - OTF2_GlobalEvtReaderCallbacks_-
Clear, [466](#)
 - OTF2_GlobalEvtReaderCallbacks_-
Delete, [466](#)
 - OTF2_GlobalEvtReaderCallbacks_-
New, [466](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetBufferFlushCallback, [467](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetEnterCallback, [467](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetLeaveCallback, [468](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMeasurementOnOffCallback, [468](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMetricCallback, [469](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiCollectiveBeginCallback, [469](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiCollectiveEndCallback, [470](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiIrecvCallback, [470](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiIrecvRequestCallback, [471](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiIsendCallback, [471](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiIsendCompleteCallback, [472](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiRecvCallback, [472](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiRequestCancelledCallback, [473](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiRequestTestCallback, [473](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetMpiSendCallback, [474](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetOmpAcquireLockCallback, [474](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetOmpForkCallback, [475](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetOmpJoinCallback, [475](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetOmpReleaseLockCallback, [476](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetOmpTaskCompleteCallback, [476](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetOmpTaskCreateCallback, [477](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetOmpTaskSwitchCallback, [477](#)
 - OTF2_GlobalEvtReaderCallbacks_-
SetParameterIntCallback, [478](#)

INDEX

- OTF2_GlobalEvtReaderCallbacks_-
SetParameterStringCallback, [478](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetParameterUnsignedIntCallback, [479](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaAcquireLockCallback, [480](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaAtomicCallback, [480](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaCollectiveBeginCallback, [481](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaCollectiveEndCallback, [481](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaGetCallback, [482](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaGroupSyncCallback, [482](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpCompleteBlockingCallback, [483](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpCompleteNonBlockingCallback, [483](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpCompleteRemoteCallback, [484](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaOpTestCallback, [484](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaPutCallback, [485](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaReleaseLockCallback, [485](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaRequestLockCallback, [486](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaSyncCallback, [486](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaTryLockCallback, [487](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaWaitChangeCallback, [487](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaWinCreateCallback, [488](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetRmaWinDestroyCallback, [488](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadAcquireLockCallback, [489](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadForkCallback, [489](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadJoinCallback, [490](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadReleaseLockCallback, [490](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTaskCompleteCallback, [491](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTaskCreateCallback, [491](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTaskSwitchCallback, [492](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTeamBeginCallback, [492](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetThreadTeamEndCallback, [493](#)
- OTF2_GlobalEvtReaderCallbacks_-
SetUnknownCallback, [493](#)
- OTF2_GlobalEvtReaderCallbacks_Clear
- OTF2_GlobalEvtReaderCallbacks.h, [466](#)
- OTF2_GlobalEvtReaderCallbacks_Delete
- OTF2_GlobalEvtReaderCallbacks.h, [466](#)
- OTF2_GlobalEvtReaderCallbacks_New
- OTF2_GlobalEvtReaderCallbacks.h, [466](#)
- OTF2_GlobalEvtReaderCallbacks_SetBufferFlushCallback
- OTF2_GlobalEvtReaderCallbacks.h, [467](#)
- OTF2_GlobalEvtReaderCallbacks_SetEnterCallback
- OTF2_GlobalEvtReaderCallbacks.h, [467](#)
- OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback

INDEX

OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetOmpJoinCallback
468 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMeasurementOffCallback
468 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetOmpReleaseLockCallback
468 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMetricCallback
468 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCompleteCallback
469 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiCollectiveBeginCallback
469 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetOmpTaskCreateCallback
469 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiCollectiveEndCallback
477 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetOmpTaskSwitchCallback
470 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvCallback
479 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetParameterIntCallback
470 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvRequestCallback
478 OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetParameterStringCallback
471 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCallback
478 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetParameterUnsignedIntCallback
471 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiIsendCompleteCallback
479 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaAcquireLockCallback
472 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiRecvCallback
480 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaAtomicCallback
472 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiRequestCancelledCallback
480 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveBeginCallback
473 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiRequestTestCallback
481 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaCollectiveEndCallback
473 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetMpiSendCallback
481 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaGetCallback
474 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetOmpAcquireLockCallback
482 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaGroupSyncCallback
474 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetOmpForwardCallback
482 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteBlockingCallback
475

INDEX

OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadReleaseLockCallback
483 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadTaskCompleteCallback
483 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaOpCompleteRemoteCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadTaskCreateCallback
484 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaOpTestCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadTaskSwitchCallback
484 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaPutCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadTeamBeginCallback
485 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaReleaseLockCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetThreadTeamEndCallback
485 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaRequestLockCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalEvtReaderCallbacks_SetUnknownCallback
486 OTF2_GlobalEvtReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetRmaSyncCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReader.h, 494
486 OTF2_GlobalSnapReader_ReadSnapshots,
OTF2_GlobalEvtReaderCallbacks_SetRmaTryLockCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReader_SetCallbacks,
487 OTF2_GlobalSnapReader_ReadSnapshots
OTF2_GlobalEvtReaderCallbacks_SetRmaWaitChangeCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReader.h, 495
487 OTF2_GlobalSnapReader_SetCallbacks
OTF2_GlobalEvtReaderCallbacks_SetRmaWinCreateCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReader.h, 495
488 OTF2_GlobalSnapReaderCallback_Enter
OTF2_GlobalEvtReaderCallbacks_SetRmaWinDestroyCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallback_MeasurementOnOff
488 OTF2_GlobalSnapReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetThreadAcquireLockCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallback_Metric
489 OTF2_GlobalSnapReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetThreadForkCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin
489 OTF2_GlobalSnapReaderCallbacks.h,
OTF2_GlobalEvtReaderCallbacks_SetThreadJoinCallback
OTF2_GlobalEvtReaderCallbacks.h, OTF2_GlobalSnapReaderCallback_MpiCollectiveEnd
490

OTF2_GlobalSnapReaderCallbacks.h 505
 OTF2_GlobalSnapReaderCallback_SnapshotStart 517
 OTF2_GlobalSnapReaderCallback_MpiIrecv 516
 OTF2_GlobalSnapReaderCallbacks.h 506
 OTF2_GlobalSnapReaderCallback_MpiIrecvRequest 517
 OTF2_GlobalSnapReaderCallbacks.h 506
 OTF2_GlobalSnapReaderCallback_MpiIsend 517
 OTF2_GlobalSnapReaderCallbacks.h 507
 OTF2_GlobalSnapReaderCallback_MpiIsendComplete 502
 OTF2_GlobalSnapReaderCallbacks.h 508
 OTF2_GlobalSnapReaderCallback_MpiRecv 509
 OTF2_GlobalSnapReaderCallbacks.h 509
 OTF2_GlobalSnapReaderCallback_MpiSend 510
 OTF2_GlobalSnapReaderCallbacks.h 510
 OTF2_GlobalSnapReaderCallback_OmpAcquireLock 510
 OTF2_GlobalSnapReaderCallbacks.h 510
 OTF2_GlobalSnapReaderCallback_OmpFork 511
 OTF2_GlobalSnapReaderCallbacks.h 511
 OTF2_GlobalSnapReaderCallback_OmpTaskCreate 512
 OTF2_GlobalSnapReaderCallbacks.h 512
 OTF2_GlobalSnapReaderCallback_OmpTaskSwitch 513
 OTF2_GlobalSnapReaderCallbacks.h 513
 OTF2_GlobalSnapReaderCallback_ParameterInt 513
 OTF2_GlobalSnapReaderCallbacks.h 513
 OTF2_GlobalSnapReaderCallback_ParameterString 514
 OTF2_GlobalSnapReaderCallbacks.h 514
 OTF2_GlobalSnapReaderCallback_ParameterStringGlobal 515
 OTF2_GlobalSnapReaderCallbacks.h 515
 OTF2_GlobalSnapReaderCallback_SnapshotEnd 516
 OTF2_GlobalSnapReaderCallbacks.h 516
 OTF2_GlobalSnapReaderCallbacks.h 502
 OTF2_GlobalSnapReaderCallback_MeasurementOnOff, 502
 OTF2_GlobalSnapReaderCallback_Metric, 503
 OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin, 504
 OTF2_GlobalSnapReaderCallback_MpiCollectiveEnd, 505
 OTF2_GlobalSnapReaderCallback_MpiIrecv, 506
 OTF2_GlobalSnapReaderCallback_MpiIrecvRequest, 506
 OTF2_GlobalSnapReaderCallback_MpiIsend, 507
 OTF2_GlobalSnapReaderCallback_MpiIsendComplete, 508
 OTF2_GlobalSnapReaderCallback_MpiRecv, 509
 OTF2_GlobalSnapReaderCallback_MpiSend, 510
 OTF2_GlobalSnapReaderCallback_OmpAcquireLock, 510
 OTF2_GlobalSnapReaderCallback_OmpFork, 511
 OTF2_GlobalSnapReaderCallback_OmpTaskCreate, 512
 OTF2_GlobalSnapReaderCallback_OmpTaskSwitch, 513
 OTF2_GlobalSnapReaderCallback_ParameterInt, 513
 OTF2_GlobalSnapReaderCallback_ParameterString, 514

INDEX

- OTF2_GlobalSnapReaderCallback_-ParameterUnsignedInt, [515](#)
- OTF2_GlobalSnapReaderCallback_-SnapshotEnd, [516](#)
- OTF2_GlobalSnapReaderCallback_-SnapshotStart, [516](#)
- OTF2_GlobalSnapReaderCallback_-Unknown, [517](#)
- OTF2_GlobalSnapReaderCallbacks, [517](#)
- OTF2_GlobalSnapReaderCallbacks_-Clear, [518](#)
- OTF2_GlobalSnapReaderCallbacks_-Delete, [518](#)
- OTF2_GlobalSnapReaderCallbacks_-New, [518](#)
- OTF2_GlobalSnapReaderCallbacks_-SetEnterCallback, [519](#)
- OTF2_GlobalSnapReaderCallbacks_-SetMeasurementOnOffCallback, [519](#)
- OTF2_GlobalSnapReaderCallbacks_-SetMetricCallback, [520](#)
- OTF2_GlobalSnapReaderCallbacks_-OTF2_GlobalSnapReaderCallbacks_DeleteSetMpiCollectiveBeginCallback, [520](#)
- OTF2_GlobalSnapReaderCallbacks_-OTF2_GlobalSnapReaderCallbacks_NewSetMpiCollectiveEndCallback, [521](#)
- OTF2_GlobalSnapReaderCallbacks_-OTF2_GlobalSnapReaderCallbacks_SetEnterCallbackSetMpiIrecvCallback, [522](#)
- OTF2_GlobalSnapReaderCallbacks_-SetMpiIrecvRequestCallback, [522](#)
- OTF2_GlobalSnapReaderCallbacks_-SetMpiIsendCallback, [523](#)
- OTF2_GlobalSnapReaderCallbacks_-OTF2_GlobalSnapReaderCallbacks_SetMetricCallbackSetMpiIsendCompleteCallback, [523](#)
- OTF2_GlobalSnapReaderCallbacks_-OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveBeginCallbackSetMpiRecvCallback, [524](#)
- OTF2_GlobalSnapReaderCallbacks_-SetMpiSendCallback, [525](#)
- OTF2_GlobalSnapReaderCallbacks_-SetOmpAcquireLockCallback, [521](#)
- OTF2_GlobalSnapReaderCallbacks_-SetOmpForkCallback, [526](#)
- OTF2_GlobalSnapReaderCallbacks_-SetOmpTaskCreateCallback, [526](#)
- OTF2_GlobalSnapReaderCallbacks_-SetOmpTaskSwitchCallback, [527](#)
- OTF2_GlobalSnapReaderCallbacks_-SetParameterIntCallback, [527](#)
- OTF2_GlobalSnapReaderCallbacks_-SetParameterStringCallback, [528](#)
- OTF2_GlobalSnapReaderCallbacks_-SetParameterUnsignedIntCallback, [529](#)
- OTF2_GlobalSnapReaderCallbacks_-SetSnapshotEndCallback, [529](#)
- OTF2_GlobalSnapReaderCallbacks_-SetSnapshotStartCallback, [530](#)
- OTF2_GlobalSnapReaderCallbacks_-SetUnknownCallback, [530](#)
- OTF2_GlobalSnapReaderCallbacks_Clear
- OTF2_GlobalSnapReaderCallbacks.h, [518](#)
- OTF2_GlobalSnapReaderCallbacks_Delete
- OTF2_GlobalSnapReaderCallbacks.h, [518](#)
- OTF2_GlobalSnapReaderCallbacks_New
- OTF2_GlobalSnapReaderCallbacks.h, [518](#)
- OTF2_GlobalSnapReaderCallbacks_SetEnterCallback
- OTF2_GlobalSnapReaderCallbacks.h, [519](#)
- OTF2_GlobalSnapReaderCallbacks_SetMeasurementOnOffCallback
- OTF2_GlobalSnapReaderCallbacks.h, [519](#)
- OTF2_GlobalSnapReaderCallbacks_SetMetricCallback
- OTF2_GlobalSnapReaderCallbacks.h, [520](#)
- OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveBeginCallback
- OTF2_GlobalSnapReaderCallbacks.h, [520](#)
- OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveEndCallback
- OTF2_GlobalSnapReaderCallbacks.h, [521](#)

OTF2_GlobalSnapReaderCallbacks_SetMpiRecvCallback, [OTF2_GlobalSnapReaderCallbacks.h](#),
 OTF2_GlobalSnapReaderCallbacks.h, [530](#)
[522](#) OTF2_GlobalSnapReaderCallbacks_SetUnknownCallback
 OTF2_GlobalSnapReaderCallbacks_SetMpiRecvCallback, [OTF2_GlobalSnapReaderCallbacks.h](#),
 OTF2_GlobalSnapReaderCallbacks.h, [530](#)
[522](#) OTF2_GroupFlag_enum
 OTF2_GlobalSnapReaderCallbacks_SetMpiSendCallback, [OTF2_Definitions.h](#), [161](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_GroupType_enum](#)
[523](#) OTF2_Definitions.h, [161](#)
 OTF2_GlobalSnapReaderCallbacks_SetMpiSendCompleteCallback, [OTF2_IdMap](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap.h](#), [533](#)
[523](#) OTF2_IdMap.h, [531](#)
 OTF2_GlobalSnapReaderCallbacks_SetMpiRecvCallback, [OTF2_IdMap](#), [533](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap_AddIdPair](#), [533](#)
[524](#) OTF2_IdMap_Clear, [534](#)
 OTF2_GlobalSnapReaderCallbacks_SetMpiSendCallback, [OTF2_IdMap_Create](#), [534](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap_CreateFromUint32Array](#),
[525](#) [534](#)
 OTF2_GlobalSnapReaderCallbacks_SetOmpAcquireLockCallback, [OTF2_IdMap_CreateFromUint64Array](#),
 OTF2_GlobalSnapReaderCallbacks.h, [535](#)
[525](#) OTF2_IdMap_Free, [535](#)
 OTF2_GlobalSnapReaderCallbacks_SetOmpForkCallback, [OTF2_IdMap_GetGlobalId](#), [536](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap_GetMode](#), [536](#)
[526](#) OTF2_IdMap_GetSize, [536](#)
 OTF2_GlobalSnapReaderCallbacks_SetOmpTaskCreateCallback, [OTF2_IdMap_Traverse](#), [537](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMapMode](#), [533](#)
[526](#) OTF2_IdMapMode_enum, [533](#)
 OTF2_GlobalSnapReaderCallbacks_SetOmpTaskSwitchCallback, [OTF2_IdMap_AddIdPair](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap.h](#), [533](#)
[527](#) OTF2_IdMap_Clear
 OTF2_GlobalSnapReaderCallbacks_SetParameterIntCallback, [OTF2_IdMap.h](#), [534](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap_Create](#)
[527](#) OTF2_IdMap_CreateFromUint32Array
 OTF2_GlobalSnapReaderCallbacks_SetParameterStringCallback, [OTF2_IdMap.h](#), [534](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap_CreateFromUint64Array](#)
[528](#) OTF2_IdMap.h, [535](#)
 OTF2_GlobalSnapReaderCallbacks_SetParameterUnsignedIntCallback, [OTF2_IdMap_Free](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap.h](#), [535](#)
[529](#) OTF2_IdMap_GetGlobalId
 OTF2_GlobalSnapReaderCallbacks_SetSnapshotsCallback, [OTF2_IdMap](#), [536](#)
 OTF2_GlobalSnapReaderCallbacks.h, [OTF2_IdMap_GetMode](#)
[529](#) OTF2_IdMap.h, [536](#)
 OTF2_GlobalSnapReaderCallbacks_SetSnapshotsCallback, [OTF2_IdMap_GetSize](#)

INDEX

OTF2_IdMap.h, 536
OTF2_IdMap_Traverse
 OTF2_IdMap.h, 537
OTF2_IdMapMode
 OTF2_IdMap.h, 533
OTF2_IdMapMode_enum
 OTF2_IdMap.h, 533
OTF2_LocationGroupType_enum
 OTF2_Definitions.h, 162
OTF2_LocationType_enum
 OTF2_Definitions.h, 162
OTF2_LockType_enum
 OTF2_Events.h, 228
OTF2_MappingType_enum
 OTF2_GeneralDefinitions.h, 365
OTF2_Marker.h, 537
 OTF2_MarkerScope_enum, 538
 OTF2_MarkerSeverity_enum, 538
OTF2_MarkerReader.h, 539
 OTF2_MarkerReader_ReadMarkers,
 540
 OTF2_MarkerReader_SetCallbacks,
 540
OTF2_MarkerReader_ReadMarkers
 OTF2_MarkerReader.h, 540
OTF2_MarkerReader_SetCallbacks
 OTF2_MarkerReader.h, 540
OTF2_MarkerReaderCallback_DefMarker
 OTF2_MarkerReaderCallbacks.h, 542
OTF2_MarkerReaderCallback_Marker
 OTF2_MarkerReaderCallbacks.h, 543
OTF2_MarkerReaderCallback_Unknown
 OTF2_MarkerReaderCallbacks.h, 543
OTF2_MarkerReaderCallbacks.h, 541
 OTF2_MarkerReaderCallback_DefMarker,
 542
 OTF2_MarkerReaderCallback_Marker,
 543
 OTF2_MarkerReaderCallback_Unknown,
 543
OTF2_MarkerReaderCallbacks_Clear,
 544
OTF2_MarkerReaderCallbacks_Delete
 544
OTF2_MarkerReaderCallbacks_New,
 545
OTF2_MarkerReaderCallbacks_SetDefMarkerCallback,
 545
OTF2_MarkerReaderCallbacks_SetMarkerCallback,
 545
OTF2_MarkerReaderCallbacks_SetUnknownCallback,
 546
OTF2_MarkerReaderCallbacks_Clear
 OTF2_MarkerReaderCallbacks.h, 544
OTF2_MarkerReaderCallbacks_Delete
 OTF2_MarkerReaderCallbacks.h, 544
OTF2_MarkerReaderCallbacks_New
 OTF2_MarkerReaderCallbacks.h, 545
OTF2_MarkerReaderCallbacks_SetDefMarkerCallback
 OTF2_MarkerReaderCallbacks.h, 545
OTF2_MarkerReaderCallbacks_SetMarkerCallback
 OTF2_MarkerReaderCallbacks.h, 545
OTF2_MarkerReaderCallbacks_SetUnknownCallback
 OTF2_MarkerReaderCallbacks.h, 546
OTF2_MarkerScope_enum
 OTF2_Marker.h, 538
OTF2_MarkerSeverity_enum
 OTF2_Marker.h, 538
OTF2_MarkerWriter.h, 547
 OTF2_MarkerWriter_WriteDefMarker,
 548
 OTF2_MarkerWriter_WriteMarker,
 548
OTF2_MarkerWriter_WriteDefMarker
 OTF2_MarkerWriter.h, 548
OTF2_MarkerWriter_WriteMarker
 OTF2_MarkerWriter.h, 548
OTF2_MasterSlaveMode
OTF2_Archive.h, 102
 OTF2_MasterSlaveMode_enum
OTF2_Archive.h, 102
OTF2_MeasurementMode_enum
OTF2_Events.h, 228
OTF2_MemoryAllocate
 OTF2_Callbacks.h, 153
OTF2_MemoryCallbacks, 92
OTF2_MemoryFreeAll
 OTF2_Callbacks.h, 154

- OTF2_MetricBase_enum
 - OTF2_Definitions.h, [163](#)
- OTF2_MetricMode_enum
 - OTF2_Definitions.h, [163](#)
- OTF2_MetricOccurrence_enum
 - OTF2_Definitions.h, [163](#)
- OTF2_MetricScope_enum
 - OTF2_Definitions.h, [164](#)
- OTF2_MetricTiming_enum
 - OTF2_Definitions.h, [164](#)
- OTF2_MetricType_enum
 - OTF2_Definitions.h, [165](#)
- OTF2_MetricValue_union, [92](#)
- OTF2_MetricValueProperty_enum
 - OTF2_Definitions.h, [165](#)
- OTF2_Paradigm_enum
 - OTF2_GeneralDefinitions.h, [366](#)
- OTF2_ParameterType_enum
 - OTF2_Definitions.h, [166](#)
- OTF2_PostFlushCallback
 - OTF2_Callbacks.h, [154](#)
- OTF2_PreFlushCallback
 - OTF2_Callbacks.h, [155](#)
- OTF2_Reader.h, [549](#)
 - OTF2_Reader_Close, [554](#)
 - OTF2_Reader_CloseDefReader, [555](#)
 - OTF2_Reader_CloseEvtReader, [555](#)
 - OTF2_Reader_CloseGlobalDefReader, [555](#)
 - OTF2_Reader_CloseGlobalEvtReader, [556](#)
 - OTF2_Reader_CloseGlobalSnapReader, [556](#)
 - OTF2_Reader_CloseMarkerReader, [556](#)
 - OTF2_Reader_CloseMarkerWriter, [557](#)
 - OTF2_Reader_CloseSnapReader, [557](#)
 - OTF2_Reader_CloseThumbReader, [558](#)
 - OTF2_Reader_GetBoolProperty, [558](#)
 - OTF2_Reader_GetChunkSize, [559](#)
 - OTF2_Reader_GetCompression, [559](#)
 - OTF2_Reader_GetCreator, [559](#)
 - OTF2_Reader_GetDefReader, [560](#)
 - OTF2_Reader_GetDescription, [560](#)
 - OTF2_Reader_GetEvtReader, [560](#)
 - OTF2_Reader_GetFileSubstrate, [561](#)
 - OTF2_Reader_GetGlobalDefReader, [561](#)
 - OTF2_Reader_GetGlobalEvtReader, [561](#)
 - OTF2_Reader_GetGlobalSnapReader, [562](#)
 - OTF2_Reader_GetMachineName, [562](#)
 - OTF2_Reader_GetMarkerReader, [562](#)
 - OTF2_Reader_GetMarkerWriter, [563](#)
 - OTF2_Reader_GetNumberOfGlobalDefinitions, [563](#)
 - OTF2_Reader_GetNumberOfLocations, [564](#)
 - OTF2_Reader_GetNumberOfSnapshots, [564](#)
 - OTF2_Reader_GetNumberOfThumbnails, [564](#)
 - OTF2_Reader_GetProperty, [565](#)
 - OTF2_Reader_GetPropertyNames, [565](#)
 - OTF2_Reader_GetSnapReader, [566](#)
 - OTF2_Reader_GetThumbReader, [566](#)
 - OTF2_Reader_GetTraceId, [566](#)
 - OTF2_Reader_GetVersion, [567](#)
 - OTF2_Reader_HasGlobalEvent, [567](#)
 - OTF2_Reader_Open, [568](#)
 - OTF2_Reader_ReadAllGlobalDefinitions, [568](#)
 - OTF2_Reader_ReadAllGlobalEvents, [568](#)
 - OTF2_Reader_ReadAllGlobalSnapshots, [569](#)
 - OTF2_Reader_ReadAllLocalDefinitions, [569](#)
 - OTF2_Reader_ReadAllLocalEvents, [570](#)
 - OTF2_Reader_ReadAllLocalSnapshots, [570](#)
 - OTF2_Reader_ReadAllMarkers, [570](#)
 - OTF2_Reader_ReadGlobalDefinitions, [571](#)

INDEX

OTF2_Reader_ReadGlobalEvent, [571](#) OTF2_Reader_CloseSnapReader
OTF2_Reader_ReadGlobalEvents, [572](#) OTF2_Reader.h, [557](#)
OTF2_Reader_ReadGlobalSnapshots, OTF2_Reader_CloseThumbReader
[572](#) OTF2_Reader.h, [558](#)
OTF2_Reader_ReadLocalDefinitions, OTF2_Reader_GetBoolProperty
[573](#) OTF2_Reader.h, [558](#)
OTF2_Reader_ReadLocalEvents, [573](#) OTF2_Reader_GetChunkSize
OTF2_Reader_ReadLocalEventsBackward, OTF2_Reader.h, [559](#)
[574](#) OTF2_Reader_GetCompression
OTF2_Reader_ReadLocalSnapshots, OTF2_Reader.h, [559](#)
[574](#) OTF2_Reader_GetCreator
OTF2_Reader_ReadMarkers, [575](#) OTF2_Reader.h, [559](#)
OTF2_Reader_RegisterDefCallbacks, OTF2_Reader_GetDefReader
[575](#) OTF2_Reader.h, [560](#)
OTF2_Reader_RegisterEvtCallbacks, OTF2_Reader_GetDescription
[576](#) OTF2_Reader.h, [560](#)
OTF2_Reader_RegisterGlobalDefCallbacks, OTF2_Reader_GetEvtReader
[576](#) OTF2_Reader.h, [560](#)
OTF2_Reader_RegisterGlobalEvtCallbacks, OTF2_Reader_GetFileSubstrate
[576](#) OTF2_Reader.h, [561](#)
OTF2_Reader_RegisterGlobalSnapCallbacks, OTF2_Reader_GetGlobalDefReader
[577](#) OTF2_Reader.h, [561](#)
OTF2_Reader_RegisterMarkerCallbacks, OTF2_Reader_GetGlobalEvtReader
[577](#) OTF2_Reader.h, [561](#)
OTF2_Reader_RegisterSnapCallbacks, OTF2_Reader_GetGlobalSnapReader
[578](#) OTF2_Reader.h, [562](#)
OTF2_Reader_SetFileSionCallbacks, OTF2_Reader_GetMachineName
[578](#) OTF2_Reader.h, [562](#)
OTF2_Reader_Close OTF2_Reader_GetMarkerReader
OTF2_Reader.h, [554](#) OTF2_Reader.h, [562](#)
OTF2_Reader_CloseDefReader OTF2_Reader_GetMarkerWriter
OTF2_Reader.h, [555](#) OTF2_Reader.h, [563](#)
OTF2_Reader_CloseEvtReader OTF2_Reader_GetNumberOfGlobalDefinitions
OTF2_Reader.h, [555](#) OTF2_Reader.h, [563](#)
OTF2_Reader_CloseGlobalDefReader OTF2_Reader_GetNumberOfLocations
OTF2_Reader.h, [555](#) OTF2_Reader.h, [564](#)
OTF2_Reader_CloseGlobalEvtReader OTF2_Reader_GetNumberOfSnapshots
OTF2_Reader.h, [556](#) OTF2_Reader.h, [564](#)
OTF2_Reader_CloseGlobalSnapReader OTF2_Reader_GetNumberOfThumbnails
OTF2_Reader.h, [556](#) OTF2_Reader.h, [564](#)
OTF2_Reader_CloseMarkerReader OTF2_Reader_GetProperty
OTF2_Reader.h, [556](#) OTF2_Reader.h, [565](#)
OTF2_Reader_CloseMarkerWriter OTF2_Reader_GetPropertyNames
OTF2_Reader.h, [557](#) OTF2_Reader.h, [565](#)

OTF2_Reader_GetSnapReader	OTF2_Reader_RegisterDefCallbacks
OTF2_Reader.h, 566	OTF2_Reader.h, 575
OTF2_Reader_GetThumbReader	OTF2_Reader_RegisterEvtCallbacks
OTF2_Reader.h, 566	OTF2_Reader.h, 576
OTF2_Reader_GetTraceId	OTF2_Reader_RegisterGlobalDefCallbacks
OTF2_Reader.h, 566	OTF2_Reader.h, 576
OTF2_Reader_GetVersion	OTF2_Reader_RegisterGlobalEvtCallbacks
OTF2_Reader.h, 567	OTF2_Reader.h, 576
OTF2_Reader_HasGlobalEvent	OTF2_Reader_RegisterGlobalSnapCallbacks
OTF2_Reader.h, 567	OTF2_Reader.h, 577
OTF2_Reader_Open	OTF2_Reader_RegisterMarkerCallbacks
OTF2_Reader.h, 568	OTF2_Reader.h, 577
OTF2_Reader_ReadAllGlobalDefinitions	OTF2_Reader_RegisterSnapCallbacks
OTF2_Reader.h, 568	OTF2_Reader.h, 578
OTF2_Reader_ReadAllGlobalEvents	OTF2_Reader_SetFileSionCallbacks
OTF2_Reader.h, 568	OTF2_Reader.h, 578
OTF2_Reader_ReadAllGlobalSnapshots	OTF2_RecorderKind_enum
OTF2_Reader.h, 569	OTF2_Definitions.h, 166
OTF2_Reader_ReadAllLocalDefinitions	OTF2_RegionFlag_enum
OTF2_Reader.h, 569	OTF2_Definitions.h, 166
OTF2_Reader_ReadAllLocalEvents	OTF2_RegionRole_enum
OTF2_Reader.h, 570	OTF2_Definitions.h, 167
OTF2_Reader_ReadAllLocalSnapshots	OTF2_RmaAtomicType_enum
OTF2_Reader.h, 570	OTF2_Events.h, 229
OTF2_Reader_ReadAllMarkers	OTF2_RmaSyncLevel_enum
OTF2_Reader.h, 570	OTF2_Events.h, 229
OTF2_Reader_ReadGlobalDefinitions	OTF2_RmaSyncType_enum
OTF2_Reader.h, 571	OTF2_Events.h, 229
OTF2_Reader_ReadGlobalEvent	OTF2_SnapReader.h, 579
OTF2_Reader.h, 571	OTF2_SnapReader_GetLocationID,
OTF2_Reader_ReadGlobalEvents	580
OTF2_Reader.h, 572	OTF2_SnapReader_ReadSnapshots,
OTF2_Reader_ReadGlobalSnapshots	580
OTF2_Reader.h, 572	OTF2_SnapReader_Seek, 581
OTF2_Reader_ReadLocalDefinitions	OTF2_SnapReader_SetCallbacks, 581
OTF2_Reader.h, 573	OTF2_SnapReader_GetLocationID
OTF2_Reader_ReadLocalEvents	OTF2_SnapReader.h, 580
OTF2_Reader.h, 573	OTF2_SnapReader_ReadSnapshots
OTF2_Reader_ReadLocalEventsBackward	OTF2_SnapReader.h, 580
OTF2_Reader.h, 574	OTF2_SnapReader_Seek
OTF2_Reader_ReadLocalSnapshots	OTF2_SnapReader.h, 581
OTF2_Reader.h, 574	OTF2_SnapReader_SetCallbacks
OTF2_Reader_ReadMarkers	OTF2_SnapReader.h, 581
OTF2_Reader.h, 575	OTF2_SnapReaderCallback_Enter

INDEX

OTF2_SnapReaderCallbacks.h, 587	OTF2_SnapReaderCallback_Enter, 587
OTF2_SnapReaderCallback_MeasurementOn, 588	OTF2_SnapReaderCallback_MeasurementOnOff, 588
OTF2_SnapReaderCallback_Metric, 589	OTF2_SnapReaderCallback_Metric, 589
OTF2_SnapReaderCallback_MpiCollectiveBegin, 590	OTF2_SnapReaderCallback_MpiCollectiveBegin, 590
OTF2_SnapReaderCallback_MpiCollectiveEnd, 590	OTF2_SnapReaderCallback_MpiCollectiveEnd, 590
OTF2_SnapReaderCallback_MpiIrecv, 591	OTF2_SnapReaderCallback_MpiIrecv, 591
OTF2_SnapReaderCallback_MpiIrecvRequest, 592	OTF2_SnapReaderCallback_MpiIrecvRequest, 592
OTF2_SnapReaderCallback_MpiIsend, 593	OTF2_SnapReaderCallback_MpiIsend, 593
OTF2_SnapReaderCallback_MpiIsendComplete, 594	OTF2_SnapReaderCallback_MpiIsendComplete, 594
OTF2_SnapReaderCallback_MpiRecv, 594	OTF2_SnapReaderCallback_MpiRecv, 594
OTF2_SnapReaderCallback_MpiSend, 595	OTF2_SnapReaderCallback_MpiSend, 595
OTF2_SnapReaderCallback_OmpAcquireLock, 596	OTF2_SnapReaderCallback_OmpAcquireLock, 596
OTF2_SnapReaderCallback_OmpFork, 597	OTF2_SnapReaderCallback_OmpFork, 597
OTF2_SnapReaderCallback_OmpTaskCreate, 598	OTF2_SnapReaderCallback_OmpTaskCreate, 598
OTF2_SnapReaderCallback_OmpTaskSwitch, 598	OTF2_SnapReaderCallback_OmpTaskSwitch, 598
OTF2_SnapReaderCallback_ParameterInt, 599	OTF2_SnapReaderCallback_ParameterInt, 599
OTF2_SnapReaderCallback_ParameterString, 600	OTF2_SnapReaderCallback_ParameterString, 600
OTF2_SnapReaderCallback_ParameterUnsignedInt, 601	OTF2_SnapReaderCallback_ParameterUnsignedInt, 601
OTF2_SnapReaderCallback_SnapshotEnd, 602	OTF2_SnapReaderCallback_SnapshotEnd, 602
OTF2_SnapReaderCallback_SnapshotStart, 602	OTF2_SnapReaderCallback_SnapshotStart, 602
OTF2_SnapReaderCallback_Unknown, 603	OTF2_SnapReaderCallback_Unknown, 603
OTF2_SnapReaderCallbacks, 603	OTF2_SnapReaderCallbacks, 603
OTF2_SnapReaderCallbacks.h, 603	OTF2_SnapReaderCallbacks_Clear, 604
OTF2_SnapReaderCallbacks.h, 582	

[OTF2_SnapReaderCallbacks_Delete](#), [OTF2_SnapReaderCallbacks_SetUnknownCallback](#),
[604](#) [616](#)
[OTF2_SnapReaderCallbacks_New](#), [OTF2_SnapReaderCallbacks_Clear](#)
[604](#) [OTF2_SnapReaderCallbacks.h](#), [604](#)
[OTF2_SnapReaderCallbacks_SetEnterCallback](#), [OTF2_SnapReaderCallbacks_Delete](#)
[604](#) [OTF2_SnapReaderCallbacks.h](#), [604](#)
[OTF2_SnapReaderCallbacks_SetMeasurementOnOffCallback](#), [OTF2_SnapReaderCallbacks_New](#)
[605](#) [OTF2_SnapReaderCallbacks.h](#), [604](#)
[OTF2_SnapReaderCallbacks_SetMetricCallback](#), [OTF2_SnapReaderCallbacks_SetEnterCallback](#)
[606](#) [OTF2_SnapReaderCallbacks.h](#), [604](#)
[OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback](#), [OTF2_SnapReaderCallbacks_SetMeasurementOnOffCallback](#)
[606](#) [OTF2_SnapReaderCallbacks.h](#), [605](#)
[OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback](#), [OTF2_SnapReaderCallbacks_SetMetricCallback](#)
[607](#) [OTF2_SnapReaderCallbacks.h](#), [606](#)
[OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback](#), [OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback](#)
[607](#) [OTF2_SnapReaderCallbacks.h](#), [606](#)
[OTF2_SnapReaderCallbacks_SetMpiIrecvCallback](#), [OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback](#)
[608](#) [OTF2_SnapReaderCallbacks.h](#), [607](#)
[OTF2_SnapReaderCallbacks_SetMpiIrecvRequestCallback](#), [OTF2_SnapReaderCallbacks_SetMpiIrecvCallback](#)
[609](#) [OTF2_SnapReaderCallbacks.h](#), [608](#)
[OTF2_SnapReaderCallbacks_SetMpiIsendCallback](#), [OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback](#)
[610](#) [OTF2_SnapReaderCallbacks.h](#), [608](#)
[OTF2_SnapReaderCallbacks_SetMpiIsendCompleteCallback](#), [OTF2_SnapReaderCallbacks_SetMpiRecvCallback](#)
[611](#) [OTF2_SnapReaderCallbacks.h](#), [610](#)
[OTF2_SnapReaderCallbacks_SetMpiSendCallback](#), [OTF2_SnapReaderCallbacks_SetMpiSendCallback](#)
[611](#) [OTF2_SnapReaderCallbacks.h](#), [610](#)
[OTF2_SnapReaderCallbacks_SetMpiAcquireLockCallback](#), [OTF2_SnapReaderCallbacks_SetMpiAcquireLockCallback](#)
[612](#) [OTF2_SnapReaderCallbacks.h](#), [611](#)
[OTF2_SnapReaderCallbacks_SetMpiForkCallback](#), [OTF2_SnapReaderCallbacks_SetMpiForkCallback](#)
[612](#) [OTF2_SnapReaderCallbacks.h](#), [611](#)
[OTF2_SnapReaderCallbacks_SetMpiTaskCreateCallback](#), [OTF2_SnapReaderCallbacks_SetMpiTaskCreateCallback](#)
[613](#) [OTF2_SnapReaderCallbacks.h](#), [612](#)
[OTF2_SnapReaderCallbacks_SetMpiTaskSwitchCallback](#), [OTF2_SnapReaderCallbacks_SetMpiTaskSwitchCallback](#)
[613](#) [OTF2_SnapReaderCallbacks.h](#), [612](#)
[OTF2_SnapReaderCallbacks_SetMpiParameterIntCallback](#), [OTF2_SnapReaderCallbacks_SetMpiParameterIntCallback](#)
[614](#) [OTF2_SnapReaderCallbacks.h](#), [613](#)
[OTF2_SnapReaderCallbacks_SetMpiParameterStringCallback](#), [OTF2_SnapReaderCallbacks_SetMpiParameterStringCallback](#)
[615](#) [OTF2_SnapReaderCallbacks.h](#), [613](#)
[OTF2_SnapReaderCallbacks_SetMpiParameterUnsignedIntCallback](#), [OTF2_SnapReaderCallbacks_SetMpiParameterUnsignedIntCallback](#)
[615](#) [OTF2_SnapReaderCallbacks.h](#), [614](#)

INDEX

OTF2_SnapReaderCallbacks_SetSnapshot, [OTF2_SnapWriter_GetLocationID](#)
OTF2_SnapReaderCallbacks.h, [615](#) OTF2_SnapWriter.h, [620](#)
OTF2_SnapReaderCallbacks_SetSnapshotStart, [OTF2_SnapWriter_MeasurementOnOff](#)
OTF2_SnapReaderCallbacks.h, [615](#) OTF2_SnapWriter.h, [621](#)
OTF2_SnapReaderCallbacks_SetUnknownCallback, [OTF2_SnapWriter_Metric](#)
OTF2_SnapReaderCallbacks.h, [616](#) OTF2_SnapWriter.h, [621](#)
OTF2_SnapWriter OTF2_SnapWriter_MpiCollectiveBegin
OTF2_SnapWriter.h, [619](#) OTF2_SnapWriter.h, [622](#)
OTF2_SnapWriter.h, [616](#) OTF2_SnapWriter_MpiCollectiveEnd
OTF2_SnapWriter, [619](#) OTF2_SnapWriter.h, [623](#)
OTF2_SnapWriter_Enter, [620](#) OTF2_SnapWriter_MpiIrecv
OTF2_SnapWriter_GetLocationID, OTF2_SnapWriter.h, [623](#)
[620](#) OTF2_SnapWriter_MpiIrecvRequest
OTF2_SnapWriter_MeasurementOnOff, OTF2_SnapWriter.h, [624](#)
[621](#) OTF2_SnapWriter_MpiIsend
OTF2_SnapWriter_Metric, [621](#) OTF2_SnapWriter.h, [625](#)
OTF2_SnapWriter_MpiCollectiveBegin, [OTF2_SnapWriter_MpiIsendComplete](#)
[622](#) OTF2_SnapWriter.h, [626](#)
OTF2_SnapWriter_MpiCollectiveEnd, [OTF2_SnapWriter_MpiRecv](#)
[623](#) OTF2_SnapWriter.h, [626](#)
OTF2_SnapWriter_MpiIrecv, [623](#) OTF2_SnapWriter_MpiSend
OTF2_SnapWriter_MpiIrecvRequest, OTF2_SnapWriter.h, [627](#)
[624](#) OTF2_SnapWriter_OmpAcquireLock
OTF2_SnapWriter_MpiIsend, [625](#) OTF2_SnapWriter.h, [628](#)
OTF2_SnapWriter_MpiIsendComplete, [OTF2_SnapWriter_OmpFork](#)
[626](#) OTF2_SnapWriter.h, [629](#)
OTF2_SnapWriter_MpiRecv, [626](#) OTF2_SnapWriter_OmpTaskCreate
OTF2_SnapWriter_MpiSend, [627](#) OTF2_SnapWriter.h, [629](#)
OTF2_SnapWriter_OmpAcquireLock, [OTF2_SnapWriter_OmpTaskSwitch](#)
[628](#) OTF2_SnapWriter.h, [630](#)
OTF2_SnapWriter_OmpFork, [629](#) OTF2_SnapWriter_ParameterInt
OTF2_SnapWriter_OmpTaskCreate, OTF2_SnapWriter.h, [631](#)
[629](#) OTF2_SnapWriter_ParameterString
OTF2_SnapWriter_OmpTaskSwitch, OTF2_SnapWriter.h, [631](#)
[630](#) OTF2_SnapWriter_ParameterUnsignedInt
OTF2_SnapWriter_ParameterInt, [631](#) OTF2_SnapWriter.h, [632](#)
OTF2_SnapWriter_ParameterString, [OTF2_SnapWriter_SnapshotEnd](#)
[631](#) OTF2_SnapWriter.h, [633](#)
OTF2_SnapWriter_ParameterUnsignedInt, [OTF2_SnapWriter_SnapshotStart](#)
[632](#) OTF2_SnapWriter.h, [633](#)
OTF2_SnapWriter_SnapshotEnd, [633](#) OTF2_SystemTreeDomain_enum
OTF2_SnapWriter_SnapshotStart, [633](#) OTF2_Definitions.h, [168](#)
OTF2_SnapWriter_Enter OTF2_Thumbnail.h, [634](#)
OTF2_SnapWriter.h, [620](#) OTF2_ThumbReader_GetHeader, [635](#)

- OTF2_ThumbReader_ReadSample,
 [636](#)
- OTF2_ThumbWriter_WriteSample,
 [636](#)
- OTF2_ThumbnailType_enum
 - OTF2_GeneralDefinitions.h, [366](#)
- OTF2_ThumbReader_GetHeader
 - OTF2_Thumbnail.h, [635](#)
- OTF2_ThumbReader_ReadSample
 - OTF2_Thumbnail.h, [636](#)
- OTF2_ThumbWriter_WriteSample
 - OTF2_Thumbnail.h, [636](#)
- OTF2_Type_enum
 - OTF2_GeneralDefinitions.h, [367](#)
- OTF2_UNDEFINED_TYPE
 - OTF2_GeneralDefinitions.h, [363](#)