

# OPEN TRACE FORMAT 2 USER MANUAL

1.4 (revision 3771)

Mon May 12 2014 11:57:03

---

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

---

## CONTENTS

---

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

## CONTENTS

---

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

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



## CONTENTS

---

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

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

---

## CONTENTS

---

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



# Chapter 1

## Open Trace Format 2

### 1.1 Introduction

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

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

### 1.2 Get started

[OTF2 usage examples](#)

[OTF2 records](#)

[OTF2 callbacks](#)

[Usage of OTF2 tools](#)



# **Appendices**





## Appendix A

# OTF2 INSTALL

For generic installation instructions see below.

Configuration of OTF2

\*\*\*\*\*

'configure' configures scorep to adapt to many kinds of systems.

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

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

Defaults for the options are specified in brackets.

Configuration:

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

Installation directories:

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

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

---

## APPENDIX A. OTF2 INSTALL

---

For better control, use the options below.

Fine tuning of the installation directories:

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

Program names:

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

System types:

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

Optional Features:

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

Optional Packages:

--with-PACKAGE[=ARG]	use PACKAGE [ARG=yes]
----------------------	-----------------------

---

```

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

```

Some influential environment variables:

```

CC_FOR_BUILD          C compiler command for the frontend build
CXX_FOR_BUILD         C++ compiler command for the frontend build
F77_FOR_BUILD         Fortran 77 compiler command for the frontend build
FC_FOR_BUILD          Fortran compiler command for the frontend build
CPPFLAGS_FOR_BUILD    (Objective) C/C++ preprocessor flags for the frontend build,
                        e.g. -I<include dir> if you have headers in a nonstandard
                        directory <include dir>
CFLAGS_FOR_BUILD      C compiler flags for the frontend build
CXXFLAGS_FOR_BUILD    C++ compiler flags for the frontend build
FFLAGS_FOR_BUILD      Fortran 77 compiler flags for the frontend build
FCFLAGS_FOR_BUILD     Fortran compiler flags for the frontend build
LDFLAGS_FOR_BUILD     linker flags for the frontend build, e.g. -L<lib dir> if you
                        have libraries in a nonstandard directory <lib dir>
LIBS_FOR_BUILD        libraries to pass to the linker for the frontend build, e.g.
                        -l<library>
CC                    C compiler command
CFLAGS                C compiler flags
LDFLAGS               linker flags, e.g. -L<lib dir> if you have libraries in a
                        nonstandard directory <lib dir>
LIBS                  libraries to pass to the linker, e.g. -l<library>
CPPFLAGS              (Objective) C/C++ preprocessor flags, e.g. -I<include dir> if
                        you have headers in a nonstandard directory <include dir>
CXX                   C++ compiler command
CXXFLAGS              C++ compiler flags
CPP                   C preprocessor
CXXCPP                C++ preprocessor

```

Use these variables to override the choices made by 'configure' or to help it to find libraries and programs with nonstandard names/locations.

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

---

## APPENDIX A. OTF2 INSTALL

---

### Installation Instructions

\*\*\*\*\*

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

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

### Basic Installation

=====

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

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

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

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

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

The simplest way to compile this package is:

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

---

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

2. Type 'make' to compile the package.
3. Optionally, type 'make check' to run any self-tests that come with the package, generally using the just-built uninstalled binaries.
4. Type 'make install' to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the 'make install' phase executed with root privileges.
5. Optionally, type 'make installcheck' to repeat any self-tests, but this time using the binaries in their final installed location. This target does not install anything. Running this target as a regular user, particularly if the prior 'make install' required root privileges, verifies that the installation completed correctly.
6. You can remove the program binaries and object files from the source code directory by typing 'make clean'. To also remove the files that 'configure' created (so you can compile the package for a different kind of computer), type 'make distclean'. There is also a 'make maintainer-clean' target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.
7. Often, you can also type 'make uninstall' to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.
8. Some packages, particularly those that use Automake, provide 'make distcheck', which can be used by developers to test that all other targets like 'make install' and 'make uninstall' work correctly. This target is generally not run by end users.

#### Compilers and Options

=====

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

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

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

---

## APPENDIX A. OTF2 INSTALL

---

\*Note Defining Variables::, for more details.

### Compiling For Multiple Architectures

=====

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

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

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

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

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

### Installation Names

=====

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

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

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

---

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

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

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

#### Optional Features =====

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

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

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

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

---

## APPENDIX A. OTF2 INSTALL

---

Particular systems  
=====

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

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

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

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

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

and if that doesn't work, try

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

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

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

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

Specifying the System Type  
=====

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

CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

OS  
KERNEL-OS

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



---

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

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

#### Sharing Defaults

=====

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

#### Defining Variables

=====

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

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

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

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

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

#### `'configure'` Invocation

=====

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

`'--help'`

`'-h'`

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

`'--help=short'`

`'--help=recursive'`

Print a summary of the options unique to this package's

`'configure'`, and exit. The `'short'` variant lists options used

---

## APPENDIX A. OTF2 INSTALL

only in the top level, while the 'recursive' variant lists options also present in any nested packages.

'--version'

'-V'

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

'--cache-file=FILE'

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

'--config-cache'

'-C'

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

'--quiet'

'--silent'

'-q'

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

'--srcdir=DIR'

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

'--prefix=DIR'

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

'--no-create'

'-n'

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

'configure' also accepts some other, not widely useful, options. Run

'configure --help' for more details.

## Appendix B

### Deprecated List

Global **[OTF2\\_AttributeList\\_AddString](#)**(OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute,   
Use *[OTF2\\_AttributeList\\_AddStringRef\(\)](#)* instead.

Global **[OTF2\\_AttributeList\\_GetString](#)**(const OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attri   
Use *[OTF2\\_AttributeList\\_GetStringRef\(\)](#)* instead.

Global **[OTF2\\_EventSizeEstimator\\_GetSizeOfOmpAcquireLockEvent](#)**(OTF2\_EventSizeEstimator \*estimat   
In version 1.2

Global **[OTF2\\_EventSizeEstimator\\_GetSizeOfOmpForkEvent](#)**(OTF2\_EventSizeEstimator \*estimator)   
In version 1.2

Global **[OTF2\\_EventSizeEstimator\\_GetSizeOfOmpJoinEvent](#)**(OTF2\_EventSizeEstimator \*estimator)   
In version 1.2

Global **[OTF2\\_EventSizeEstimator\\_GetSizeOfOmpReleaseLockEvent](#)**(OTF2\_EventSizeEstimator \*estimat   
In version 1.2

Global **[OTF2\\_EventSizeEstimator\\_GetSizeOfOmpTaskCompleteEvent](#)**(OTF2\_EventSizeEstimator \*estima   
In version 1.2

Global **[OTF2\\_EventSizeEstimator\\_GetSizeOfOmpTaskCreateEvent](#)**(OTF2\_EventSizeEstimator \*estimator   
In version 1.2

---

## APPENDIX B. DEPRECATED LIST

---

Global **OTF2\_EventSizeEstimator\_GetSizeOfOmpTaskSwitchEvent**(OTF2\_EventSizeEstimator \*estimator,

In version 1.2

Global **OTF2\_EvtWriter\_OmpAcquireLock**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList,

In version 1.2

Global **OTF2\_EvtWriter\_OmpFork**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_

In version 1.2

Global **OTF2\_EvtWriter\_OmpJoin**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_T

In version 1.2

Global **OTF2\_EvtWriter\_OmpReleaseLock**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList,

In version 1.2

Global **OTF2\_EvtWriter\_OmpTaskComplete**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList,

In version 1.2

Global **OTF2\_EvtWriter\_OmpTaskCreate**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, C

In version 1.2

Global **OTF2\_EvtWriter\_OmpTaskSwitch**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, C

In version 1.2

Group **records\_event** In version 1.2

In version 1.2

In version 1.2

In version 1.2

In version 1.2

In version 1.2

In version 1.2

## Appendix C

# Module Documentation

### C.1 Usage of OTF2 tools

#### Modules

- [OTF2 config tool](#)
- [OTF2 print tool](#)
- [OTF2 snapshots tool](#)
- [OTF2 marker tool](#)
- [OTF2 estimator tool](#)

### C.2 OTF2 config tool

A call to otf2-config has the following syntax:

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

#### Commands:

--cflags	prints additional compiler flags. They already contain the include flags
--cppflags	prints the include flags for the OTF2 headers
--libs	prints the required libraries for linking
--ldflags	prints the required linker flags
--cc	prints the C compiler name
--features <FEATURE-CATEGORY>	prints available features selected by <FEATURE-CATEGORY>. available feature categories: <ul style="list-style-type: none"><li>* substrates</li><li>* compressions</li></ul>
--help	prints this usage information

---

## APPENDIX C. MODULE DOCUMENTATION

---

```
--version      prints the version number of the OTF2 package and
--otf2-revision prints the revision number of the OTF2 package
--common-revision prints the revision number of the common package
--interface-version prints the interface version number
```

### Options:

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

## C.3 OTF2 print tool

A call to otf2-print has the following syntax:

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

Print selected content of the OTF2 archive specified by ANCHORFILE.

### Options:

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

## C.4 OTF2 snapshots tool

---

### C.4 OTF2 snapshots tool

A call to oft2-snapshots has the following syntax:

Usage: oft2-snapshots [OPTION]... ANCHORFILE

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

Options:

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

### C.5 OTF2 marker tool

A call to oft2-marker has the following syntax:

Usage: oft2-marker [OPTION] [ARGUMENTS]... ANCHORFILE

Read or edit a marker file.

Options:

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

Argument descriptions:

<GROUP>, <CATEGORY>, <TEXT>

---

## APPENDIX C. MODULE DOCUMENTATION

---

Arbitrary strings.

<SEVERITY> One of:

- \* NONE
- \* LOW
- \* MEDIUM
- \* HIGH

<TIME> One of the following formats:

- \* <TIMESTAMP>  
A valid timestamp inside the trace range 'global offset' and 'global offset' + 'trace length'.
- \* <TIMESTAMP>+<DURATION>  
<TIMESTAMP> and <TIMESTAMP> + <DURATION> must be valid timestamps inside the trace range 'global offset' and 'global offset' + 'trace length'.
- \* <TIMESTAMP-START>-<TIMESTAMP-END>  
Two valid timestamps inside the trace range 'global offset' and 'global offset' + 'trace length', with <TIMESTAMP-START> <= <TIMESTAMP-END>.

See the CLOCK\_PROPERTIES definition with the help of the 'otf2-print -G' tool.

<SCOPE>[:<SCOPE-REF>]

The <SCOPE> must be one of:

- \* GLOBAL
- \* LOCATION:<LOCATION-REF>
- \* LOCATION\_GROUP:<LOCATION-GROUP-REF>
- \* SYSTEM\_TREE\_NODE:<SYSTEM-TREE-NODE-REF>
- \* GROUP:<GROUP-REF>
- \* COMM:<COMMUNICATOR-REF>

<SCOPE-REF> must be a valid definition reference of the specified scope. Use 'otf2-print -G' for a list of defined references.

There is no <SCOPE-REF> for <SCOPE> 'GLOBAL'.

For a scope 'GROUP' the type of the referenced group must be 'OTF2\_GROUP\_TYPE\_LOCATIONS' or 'OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS'.

### C.6 OTF2 estimator tool

A call to otf2-estimator has the following syntax:

Usage: otf2-estimator [OPTION]...

This tool estimates the size of OTF2 events.

It will open a prompt to type in commands.

Options:

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

Commands:

list definitions	Lists all known definition names.
------------------	-----------------------------------



## C.7 OTF2 records

---

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

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

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

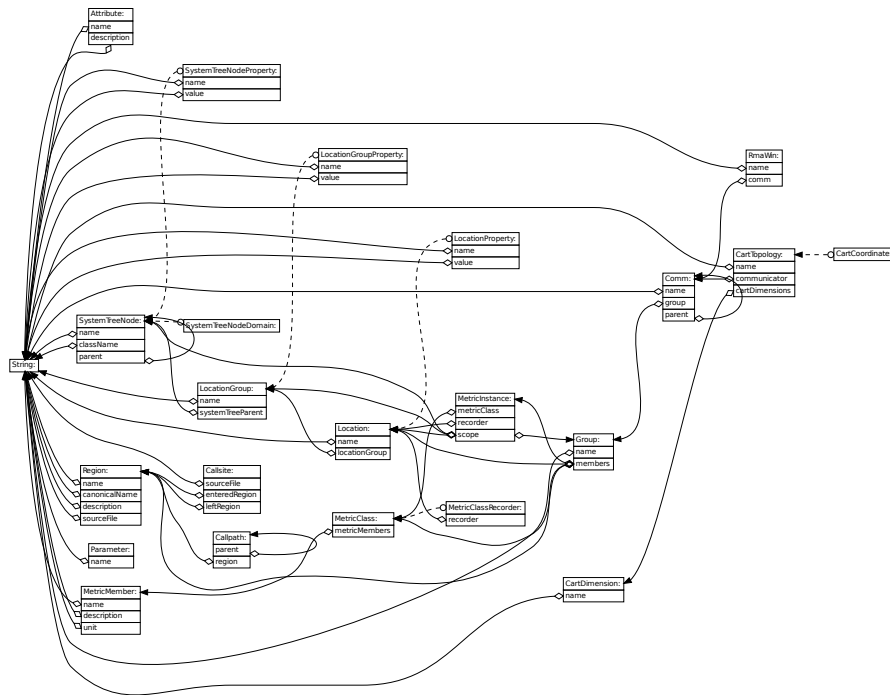
```
cat <<EOC | otf2-estimator
set Region 4
set Metric 1
get Timestamp
get Enter
get Leave
get Metric 4
exit
EOC
Timestamp 9
Enter 3
Leave 3
Metric 4 44
```

## C.7 OTF2 records

### Modules

- [List of all definition records](#)
- [List of all event records](#)
- [List of all marker records](#)
- [List of all snapshot records](#)

## C.8 List of all definition records



## C.9 ClockProperties

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

This definition is only valid as a global definition.

## Attributes

uint64_t	timerResolution	Ticks per seconds.
uint64_t	globalOffset	A timestamp smaller than all event timestamps.
uint64_t	traceLength	A timespan which includes the timespan between the smallest and greatest timestamp of all event timestamps.

## C.11 ClockOffset

---

### See also

[OTF2\\_GlobalDefWriter\\_WriteClockProperties\(\)](#)

### Since

Version 1.0

## C.10 MappingTable

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

This definition is only valid as a local definition.

### Attributes

<a href="#">OTF2_MappingType</a>	mapping-Type	Says to what type of ID the mapping table has to be applied.
const OTF2_IdMap*	idMap	Mapping table.

### See also

[OTF2\\_DefWriter\\_WriteMappingTable\(\)](#)

### Since

Version 1.0

## C.11 ClockOffset

Clock offsets are used for clock corrections.

This definition is only valid as a local definition.

### Attributes

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

**See also**

[OTF2\\_DefWriter\\_WriteClockOffset\(\)](#)

**Since**

Version 1.0

**C.12 [OTF2\\_StringRef](#) String**

The string definitions.

**Attributes**

<a href="#">OTF2_StringRef</a>	const char*	string	The string, null terminated.
--------------------------------	-------------	--------	------------------------------

**See also**

[OTF2\\_GlobalDefWriter\\_WriteString\(\)](#)

[OTF2\\_DefWriter\\_WriteString\(\)](#)

**Since**

Version 1.0

**C.13 [OTF2\\_AttributeRef](#) Attribute**

The attribute definition.

**Attributes**

<a href="#">OTF2_StringRef</a>	name	Name of the attribute. References a <a href="#">String</a> definition.
<a href="#">OTF2_StringRef</a>	description	Description of the attribute. References a <a href="#">String</a> definition. Since version 1.4.
<a href="#">OTF2_Type</a>	type	Type of the attribute value.

**See also**

[OTF2\\_GlobalDefWriter\\_WriteAttribute\(\)](#)

[OTF2\\_DefWriter\\_WriteAttribute\(\)](#)

## C.15 LocationGroup

---

### Since

Version 1.0

## C.14 *OTF2\_SystemTreeNodeRef* SystemTreeNode

The system tree node definition.

### Attributes

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

### Supplements

*SystemTreeNodeProperty*  
*SystemTreeNodeDomain*

### See also

*OTF2\_GlobalDefWriter\_WriteSystemTreeNode()*  
*OTF2\_DefWriter\_WriteSystemTreeNode()*

### Since

Version 1.0

## C.15 *OTF2\_LocationGroupRef* LocationGroup

The location group definition.

### Attributes

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

## APPENDIX C. MODULE DOCUMENTATION

---

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

### Supplements

[\*LocationGroupProperty\*](#)

### See also

[OTF2\\_GlobalDefWriter\\_WriteLocationGroup\(\)](#)  
[OTF2\\_DefWriter\\_WriteLocationGroup\(\)](#)

### Since

Version 1.0

## C.16 [\*OTF2\\_LocationRef\*](#) Location

The location definition.

### Attributes

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

### Supplements

[\*LocationProperty\*](#)

### See also

[OTF2\\_GlobalDefWriter\\_WriteLocation\(\)](#)  
[OTF2\\_DefWriter\\_WriteLocation\(\)](#)

## C.18 Callsite

---

### Since

Version 1.0

## C.17 *OTF2\_RegionRef* Region

The region definition.

### Attributes

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

### See also

[OTF2\\_GlobalDefWriter\\_WriteRegion\(\)](#)  
[OTF2\\_DefWriter\\_WriteRegion\(\)](#)

### Since

Version 1.0

## C.18 *OTF2\_CallsiteRef* Callsite

The callsite definition.

**Attributes**

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

**See also**

[OTF2\\_GlobalDefWriter\\_WriteCallsite\(\)](#)  
[OTF2\\_DefWriter\\_WriteCallsite\(\)](#)

**Since**

Version 1.0

**C.19 [\*OTF2\\_CallpathRef\*](#) Callpath**

The callpath definition.

**Attributes**

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

**See also**

[OTF2\\_GlobalDefWriter\\_WriteCallpath\(\)](#)  
[OTF2\\_DefWriter\\_WriteCallpath\(\)](#)

**Since**

Version 1.0

**C.20 [\*OTF2\\_GroupRef\*](#) Group**

The group definition.



## C.21 MetricMember

---

### Attributes

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

### See also

[OTF2\\_GlobalDefWriter\\_WriteGroup\(\)](#)  
[OTF2\\_DefWriter\\_WriteGroup\(\)](#)

### Since

Version 1.0

## C.21 [\*OTF2\\_MetricMemberRef\*](#) MetricMember

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

### Attributes

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

<a href="#"><i>OTF2_Type</i></a>	valueType	Type of the value. Only <a href="#"><i>OTF2_TYPE_INT64</i></a> , <a href="#"><i>OTF2_TYPE_UINT64</i></a> , and <a href="#"><i>OTF2_TYPE_DOUBLE</i></a> are valid types. If this metric member is recorded in an <a href="#"><i>Metric</i></a> event, than this type and the type in the event must match.
<a href="#"><i>OTF2_MetricBase</i></a>	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 <a href="#"><i>OTF2_BASE_BINARY</i></a> 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.
<a href="#"><i>OTF2_StringRef</i></a>	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 <a href="#"><i>String</i></a> definition.

### See also

[\*OTF2\\_GlobalDefWriter\\_WriteMetricMember\(\)\*](#)  
[\*OTF2\\_DefWriter\\_WriteMetricMember\(\)\*](#)

### Since

Version 1.0

## C.22 [\*OTF2\\_MetricRef\*](#) MetricClass

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

## C.23 MetricInstance

---

### Attributes

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

### Supplements

[\*MetricClassRecorder\*](#)

### See also

[OTF2\\_GlobalDefWriter\\_WriteMetricClass\(\)](#)  
[OTF2\\_DefWriter\\_WriteMetricClass\(\)](#)

### Since

Version 1.0

## C.23 [\*OTF2\\_MetricRef\*](#) MetricInstance

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type [\*OTF2\\_METRIC\\_ASYNCHRONOUS\*](#).

### Attributes

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

<a href="#"><i>OTF2_MetricScope</i></a>	metric-Scope	Defines type of scope: location, location group, system tree node, or a generic group of locations.
uint64_t	scope	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

**See also**

[OTF2\\_GlobalDefWriter\\_WriteMetricInstance\(\)](#)  
[OTF2\\_DefWriter\\_WriteMetricInstance\(\)](#)

**Since**

Version 1.0

## **C.24 [\*OTF2\\_CommRef\*](#) Comm**

The communicator definition.

**Attributes**

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

**See also**

[OTF2\\_GlobalDefWriter\\_WriteComm\(\)](#)  
[OTF2\\_DefWriter\\_WriteComm\(\)](#)

## C.26 RmaWin

---

### Since

Version 1.0

## C.25 *OTF2\_ParameterRef* Parameter

The parameter definition.

### Attributes

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

### See also

[OTF2\\_GlobalDefWriter\\_WriteParameter\(\)](#)  
[OTF2\\_DefWriter\\_WriteParameter\(\)](#)

### Since

Version 1.0

## C.26 *OTF2\_RmaWinRef* RmaWin

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

### Attributes

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

### See also

[OTF2\\_GlobalDefWriter\\_WriteRmaWin\(\)](#)  
[OTF2\\_DefWriter\\_WriteRmaWin\(\)](#)

## Since

Version 1.2

### C.27 MetricClassRecorder

The metric class recorder definition.

#### Attributes

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

#### See also

[OTF2\\_GlobalDefWriter\\_WriteMetricClassRecorder\(\)](#)  
[OTF2\\_DefWriter\\_WriteMetricClassRecorder\(\)](#)

## Since

Version 1.2

### C.28 SystemTreeNodeProperty

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

#### Attributes

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

## C.30 LocationGroupProperty

---

### See also

[OTF2\\_GlobalDefWriter\\_WriteSystemTreeNodeProperty\(\)](#)  
[OTF2\\_DefWriter\\_WriteSystemTreeNodeProperty\(\)](#)

### Since

Version 1.2

## C.29 SystemTreeNodeDomain

The system tree node domain definition.

### Attributes

<a href="#">OTF2_-SystemTreeNodeRef</a>	sys- temTreeN- ode	Parent <a href="#">SystemTreeNode</a> definition to which this one is a supplementary definition. References a <a href="#">SystemTreeNode</a> definition.
<a href="#">OTF2_-SystemTreeNodeDomain</a>	sys- temTreeDo- main	The domain in which the referenced <a href="#">SystemTreeNode</a> operates in.

### See also

[OTF2\\_GlobalDefWriter\\_WriteSystemTreeNodeDomain\(\)](#)  
[OTF2\\_DefWriter\\_WriteSystemTreeNodeDomain\(\)](#)

### Since

Version 1.2

## C.30 LocationGroupProperty

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

### Attributes

<a href="#">OTF2_-LocationGroupRef</a>	location- Group	Parent <a href="#">LocationGroup</a> definition to which this one is a supplementary definition. References a <a href="#">LocationGroup</a> definition.
--	--------------------	---

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

**See also**

[OTF2\\_GlobalDefWriter\\_WriteLocationGroupProperty\(\)](#)  
[OTF2\\_DefWriter\\_WriteLocationGroupProperty\(\)](#)

**Since**

Version 1.3

**C.31 LocationProperty**

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

**Attributes**

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

**See also**

[OTF2\\_GlobalDefWriter\\_WriteLocationProperty\(\)](#)  
[OTF2\\_DefWriter\\_WriteLocationProperty\(\)](#)

**Since**

Version 1.3

**C.32 [\*OTF2\\_CartDimensionRef\*](#) CartDimension**

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



### C.33 CartTopology

---

#### Attributes

<a href="#">OTF2_StringRef</a>	name	The name of the cartesian topology dimension. References a <a href="#">String</a> definition.
uint32_t	size	The size of the cartesian topology dimension.
<a href="#">OTF2_CartPeriodicity</a>	cartPeriodicity	Periodicity of the cartesian topology dimension.

#### See also

[OTF2\\_GlobalDefWriter\\_WriteCartDimension\(\)](#)  
[OTF2\\_DefWriter\\_WriteCartDimension\(\)](#)

#### Since

Version 1.3

### C.33 [OTF2\\_CartTopologyRef](#) CartTopology

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

#### Attributes

<a href="#">OTF2_StringRef</a>	name	The name of the topology. References a <a href="#">String</a> definition.
<a href="#">OTF2_CommRef</a>	communicator	Communicator object used to create the topology. References a <a href="#">Comm</a> definition.
uint8_t	numberOfDimensions	Number of dimensions.
<a href="#">OTF2_CartDimensionRef</a>	cartDimensions [ numberOfDimensions ]	The dimensions of this topology. References a <a href="#">CartDimension</a> definition.

#### Supplements

[CartCoordinate](#)

**See also**

[OTF2\\_GlobalDefWriter\\_WriteCartTopology\(\)](#)  
[OTF2\\_DefWriter\\_WriteCartTopology\(\)](#)

**Since**

Version 1.3

### C.34 CartCoordinate

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

**Attributes**

<a href="#"><i>OTF2-CartTopologyRef</i></a>	cartTopology	Parent <a href="#"><i>CartTopology</i></a> definition to which this one is a supplementary definition. References a <a href="#"><i>CartTopology</i></a> definition.
uint32_t	rank	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
uint8_t	numberOfDimensions	Number of dimensions.
uint32_t	coordinates [ numberOfDimensions ]	Coordinates, indexed by dimension.

**See also**

[OTF2\\_GlobalDefWriter\\_WriteCartCoordinate\(\)](#)  
[OTF2\\_DefWriter\\_WriteCartCoordinate\(\)](#)

**Since**

Version 1.3

## C.37 MeasurementOnOff

---

### C.35 List of all event records

### C.36 BufferFlush

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

#### Attributes

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location where this event happened.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The time when this event happened.
OTF2_TimeStamp	stopTime	The time the buffer flush finished.

#### See also

[\*OTF2\\_EvtWriter\\_BufferFlush\(\)\*](#)

#### Since

Version 1.0

## C.37 MeasurementOnOff

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

#### Attributes

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

#### See also

[\*OTF2\\_EvtWriter\\_MeasurementOnOff\(\)\*](#)

#### Since

Version 1.0

### **C.38 Enter**

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

#### **Attributes**

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

#### **See also**

[OTF2\\_EvtWriter\\_Enter\(\)](#)

#### **Since**

Version 1.0

### **C.39 Leave**

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

#### **Attributes**

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

#### **See also**

[OTF2\\_EvtWriter\\_Leave\(\)](#)

#### **Since**

Version 1.0

## C.41 MpilSend

---

### C.40 MpiSend

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

#### Attributes

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

#### See also

[OTF2\\_EvtWriter\\_MpiSend\(\)](#)

#### Since

Version 1.0

### C.41 MpilSend

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

#### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
uint32_t	receiver	MPI rank of receiver in communicator.

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

**See also**

[OTF2\\_EvtWriter\\_MpiSend\(\)](#)

**Since**

Version 1.0

**C.42 MpiSendComplete**

Signals the completion of non-blocking send request.

**Attributes**

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
uint64_t	requestID	ID of the related request

**See also**

[OTF2\\_EvtWriter\\_MpiSendComplete\(\)](#)

**Since**

Version 1.0

**C.43 MpiRecvRequest**

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

## C.44 MpiRecv

---

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
uint64_t	requestID	ID of the requested receive

### See also

[OTF2\\_EvtWriter\\_MpiRecvRequest\(\)](#)

### Since

Version 1.0

## C.44 MpiRecv

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

### Attributes

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

### See also

[OTF2\\_EvtWriter\\_MpiRecv\(\)](#)

### Since

Version 1.0

## C.45 MpiIrecv

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

### Attributes

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

### See also

[OTF2\\_EvtWriter\\_MpiIrecv\(\)](#)

### Since

Version 1.0

## C.46 MpiRequestTest

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
uint64_t	requestID	ID of the related request

### See also

[OTF2\\_EvtWriter\\_MpiRequestTest\(\)](#)



## C.48 MpiCollectiveBegin

---

### Since

Version 1.0

## C.47 MpiRequestCancelled

This events appears if the program canceled a request.

### Attributes

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

### See also

[OTF2\\_EvtWriter\\_MpiRequestCancelled\(\)](#)

### Since

Version 1.0

## C.48 MpiCollectiveBegin

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.).

### Attributes

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location where this event happened.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The time when this event happened.

### See also

[OTF2\\_EvtWriter\\_MpiCollectiveBegin\(\)](#)

### Since

Version 1.0

## C.49 MpiCollectiveEnd

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

### Attributes

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

### See also

[OTF2\\_EvtWriter\\_MpiCollectiveEnd\(\)](#)

### Since

Version 1.0

## C.50 OmpFork

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

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
uint32_t	numberOfRequestedThreads	Requested size of the team.

## C.52 OmpAcquireLock

---

### See also

[OTF2\\_EvtWriter\\_OmpFork\(\)](#)

### Since

Version 1.0

### Deprecated

In version 1.2

## C.51 OmpJoin

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

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.

### See also

[OTF2\\_EvtWriter\\_OmpJoin\(\)](#)

### Since

Version 1.0

### Deprecated

In version 1.2

## C.52 OmpAcquireLock

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

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

### Attributes

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

### See also

[\*OTF2\\_EvtWriter\\_OmpAcquireLock\(\)\*](#)

### Since

Version 1.0

### Deprecated

In version 1.2

## C.53 OmpReleaseLock

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

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

### Attributes

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

## C.55 OmpTaskSwitch

---

### See also

[OTF2\\_EvtWriter\\_OmpReleaseLock\(\)](#)

### Since

Version 1.0

### Deprecated

In version 1.2

## C.54 OmpTaskCreate

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

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
uint64_t	taskID	Identifier of the newly created task instance.

### See also

[OTF2\\_EvtWriter\\_OmpTaskCreate\(\)](#)

### Since

Version 1.0

### Deprecated

In version 1.2

## C.55 OmpTaskSwitch

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

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

**Attributes**

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
uint64_t	taskID	Identifier of the now active task instance.

**See also**

[OTF2\\_EvtWriter\\_OmpTaskSwitch\(\)](#)

**Since**

Version 1.0

**Deprecated**

In version 1.2

## C.56 OmpTaskComplete

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

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

**Attributes**

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
uint64_t	taskID	Identifier of the completed task instance.

**See also**

[OTF2\\_EvtWriter\\_OmpTaskComplete\(\)](#)

**Since**

Version 1.0

**Deprecated**

In version 1.2

## C.58 ParameterString

---

### C.57 Metric

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

#### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_MetricRef</a>	metric	Could be a metric class or a metric instance. References a <a href="#">MetricClass</a> , or a <a href="#">MetricInstance</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_METRIC</a> is available.
uint8_t	numberOfMetrics	Number of metrics with in the set.
<a href="#">OTF2_Type</a>	typeIDs [ numberOfMetrics ]	List of metric types. These types must match that of the corresponding <a href="#">MetricMember</a> definitions.
OTF2_MetricValue	metricValues [ numberOfMetrics ]	List of metric values.

#### See also

[OTF2\\_EvtWriter\\_Metric\(\)](#)

#### Since

Version 1.0

## C.58 ParameterString

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

### Attributes

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location where this event happened.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The time when this event happened.
<a href="#"><i>OTF2_-ParameterRef</i></a>	parameter	Parameter ID. References a <a href="#"><i>Parameter</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_PARAMETER</i></a> is available.
<a href="#"><i>OTF2_StringRef</i></a>	string	Value: Handle of a string definition References a <a href="#"><i>String</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_STRING</i></a> is available.

### See also

[\*OTF2\\_EvtWriter\\_ParameterString\(\)\*](#)

### Since

Version 1.0

## C.59 ParameterInt

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

### Attributes

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

### See also

[\*OTF2\\_EvtWriter\\_ParameterInt\(\)\*](#)



## C.61 RmaWinCreate

---

### Since

Version 1.0

## C.60 ParameterUnsignedInt

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_ParameterRef</a>	parameter	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
uint64_t	value	Value of the recorded parameter.

### See also

[OTF2\\_EvtWriter\\_ParameterUnsignedInt\(\)](#)

### Since

Version 1.0

## C.61 RmaWinCreate

An RmaWinCreate record denotes the creation of an RMA window.

### Attributes

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

**See also**

[OTF2\\_EvtWriter\\_RmaWinCreate\(\)](#)

**Since**

Version 1.2

## C.62 RmaWinDestroy

An RmaWinDestroy record denotes the destruction of an RMA window.

**Attributes**

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

**See also**

[OTF2\\_EvtWriter\\_RmaWinDestroy\(\)](#)

**Since**

Version 1.2

## C.63 RmaCollectiveBegin

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

**Attributes**

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.

**See also**

[OTF2\\_EvtWriter\\_RmaCollectiveBegin\(\)](#)

## C.65 RmaGroupSync

---

### Since

Version 1.2

## C.64 RmaCollectiveEnd

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

### Attributes

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

### See also

[OTF2\\_EvtWriter\\_RmaCollectiveEnd\(\)](#)

### Since

Version 1.2

## C.65 RmaGroupSync

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
----------------------------------	----------	---

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

**See also**

[\*OTF2\\_EvtWriter\\_RmaGroupSync\(\)\*](#)

**Since**

Version 1.2

## C.66 RmaRequestLock

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

**Attributes**

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location where this event happened.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The time when this event happened.
<a href="#"><i>OTF2_RmaWinRef</i></a>	win	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_-MAPPING_RMA_WIN</i></a> 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.
<a href="#"><i>OTF2_LockType</i></a>	lockType	Type of lock aquired.

## C.68 RmaTryLock

---

### See also

[OTF2\\_EvtWriter\\_RmaRequestLock\(\)](#)

### Since

Version 1.2

## C.67 RmaAcquireLock

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_RmaWinRef</a>	win	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> 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.
<a href="#">OTF2_LockType</a>	lockType	Type of lock aquired.

### See also

[OTF2\\_EvtWriter\\_RmaAcquireLock\(\)](#)

### Since

Version 1.2

## C.68 RmaTryLock

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
----------------------------------	----------	---

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

See also

[OTF2\\_EvtWriter\\_RmaTryLock\(\)](#)

Since

Version 1.2

### C.69 RmaReleaseLock

An RmaReleaseLock record denotes the time the lock was released.

Attributes

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

See also

[OTF2\\_EvtWriter\\_RmaReleaseLock\(\)](#)

Since

Version 1.2

## C.71 RmaWaitChange

---

### C.70 RmaSync

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

#### Attributes

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

#### See also

[OTF2\\_EvtWriter\\_RmaSync\(\)](#)

#### Since

Version 1.2

## C.71 RmaWaitChange

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

#### Attributes

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

#### See also

[OTF2\\_EvtWriter\\_RmaWaitChange\(\)](#)

## Since

Version 1.2

### C.72 RmaPut

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

#### Attributes

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location where this event happened.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The time when this event happened.
<a href="#"><i>OTF2_RmaWinRef</i></a>	win	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> is available.
uint32_t	remote	Rank of the target process.
uint64_t	bytes	Bytes sent to target.
uint64_t	matchingId	ID used for matching the appropriate completion record.

## See also

[\*OTF2\\_EvtWriter\\_RmaPut\(\)\*](#)

## Since

Version 1.2

### C.73 RmaGet

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

#### Attributes

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



## C.74 RmaAtomic

---

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

See also

[OTF2\\_EvtWriter\\_RmaGet\(\)](#)

Since

Version 1.2

## C.74 RmaAtomic

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

Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_RmaWinRef</a>	win	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2-MAPPING_RMA_WIN</a> is available.
uint32_t	remote	Rank of the target process.
<a href="#">OTF2-RmaAtomicType</a>	type	Type of atomic operation.
uint64_t	bytesSent	Bytes sent to target.
uint64_t	bytesReceived	Bytes received from target.
uint64_t	matchingId	ID used for matching the appropriate completion record.

See also

[OTF2\\_EvtWriter\\_RmaAtomic\(\)](#)

Since

Version 1.2

## C.75 RmaOpCompleteBlocking

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_RmaWinRef</a>	win	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
uint64_t	matchingId	ID used for matching the appropriate completion record.

### See also

[OTF2\\_EvtWriter\\_RmaOpCompleteBlocking\(\)](#)

### Since

Version 1.2

## C.76 RmaOpCompleteNonBlocking

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_RmaWinRef</a>	win	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
uint64_t	matchingId	ID used for matching the appropriate completion record.

## C.78 RmaOpCompleteRemote

---

### See also

[OTF2\\_EvtWriter\\_RmaOpCompleteNonBlocking\(\)](#)

### Since

Version 1.2

## C.77 RmaOpTest

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_RmaWinRef</a>	win	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
uint64_t	matchingId	ID used for matching the appropriate completion record.

### See also

[OTF2\\_EvtWriter\\_RmaOpTest\(\)](#)

### Since

Version 1.2

## C.78 RmaOpCompleteRemote

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.

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

**See also**

[\*OTF2\\_EvtWriter\\_RmaOpCompleteRemote\(\)\*](#)

**Since**

Version 1.2

## C.79 ThreadFork

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

**Attributes**

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location where this event happened.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The time when this event happened.
<a href="#"><i>OTF2_Paradigm</i></a>	model	The threading paradigm this event took place.
uint32_t	numberOfRequestedThreads	Requested size of the team.

**See also**

[\*OTF2\\_EvtWriter\\_ThreadFork\(\)\*](#)

**Since**

Version 1.2

## C.81 ThreadTeamBegin

---

### C.80 ThreadJoin

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

#### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_Paradigm</a>	model	The threading paradigm this event took place.

#### See also

[OTF2\\_EvtWriter\\_ThreadJoin\(\)](#)

#### Since

Version 1.2

## C.81 ThreadTeamBegin

The current location enters the specified thread team.

#### Attributes

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

#### See also

[OTF2\\_EvtWriter\\_ThreadTeamBegin\(\)](#)

#### Since

Version 1.2

## C.82 ThreadTeamEnd

The current location leaves the specified thread team.

### Attributes

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

### See also

[OTF2\\_EvtWriter\\_ThreadTeamEnd\(\)](#)

### Since

Version 1.2

## C.83 ThreadAcquireLock

An ThreadAcquireLock record marks that a thread acquires an lock.

### Attributes

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

### See also

[OTF2\\_EvtWriter\\_ThreadAcquireLock\(\)](#)

## C.85 ThreadTaskCreate

---

### Since

Version 1.2

## C.84 ThreadReleaseLock

An ThreadReleaseLock record marks that a thread releases an lock.

### Attributes

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

### See also

[\*OTF2\\_EvtWriter\\_ThreadReleaseLock\(\)\*](#)

### Since

Version 1.2

## C.85 ThreadTaskCreate

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

### Attributes

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

uint32_t	creatingThread	Creating thread of this task.
uint32_t	generationNumber	Thread-private generation number of task's creating thread.

**See also**

[OTF2\\_EvtWriter\\_ThreadTaskCreate\(\)](#)

**Since**

Version 1.2

## C.86 ThreadTaskSwitch

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

**Attributes**

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

**See also**

[OTF2\\_EvtWriter\\_ThreadTaskSwitch\(\)](#)

**Since**

Version 1.2



## C.88 ThreadCreate

---

### C.87 ThreadTaskComplete

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

#### Attributes

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

#### See also

[OTF2\\_EvtWriter\\_ThreadTaskComplete\(\)](#)

#### Since

Version 1.2

## C.88 ThreadCreate

The location created successfully a new thread.

#### Attributes

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

**See also**

[OTF2\\_EvtWriter\\_ThreadCreate\(\)](#)

**Since**

Version 1.3

**C.89 ThreadBegin**

Marks the begin of a thread created by another thread.

**Attributes**

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

**See also**

[OTF2\\_EvtWriter\\_ThreadBegin\(\)](#)

**Since**

Version 1.3

**C.90 ThreadWait**

The location waits for the completion of another thread.

**Attributes**

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.

## C.91 ThreadEnd

---

<a href="#">OTF2_CommRef</a>	thread-Contingent	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
uint64_t	sequence-Count	A threadContingent unique number. The corresponding <a href="#">ThreadEnd</a> event does have the same number.

### See also

[OTF2\\_EvtWriter\\_ThreadWait\(\)](#)

### Since

Version 1.3

## C.91 ThreadEnd

Marks the end of a thread.

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<a href="#">OTF2_CommRef</a>	thread-Contingent	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
uint64_t	sequence-Count	A threadContingent unique number. The corresponding <a href="#">ThreadWait</a> event does have the same number. <a href="#">OTF2_UNDEFINED_UINT64</a> in case no corresponding <a href="#">ThreadWait</a> event exists.

### See also

[OTF2\\_EvtWriter\\_ThreadEnd\(\)](#)

### Since

Version 1.3

## C.92 List of all marker records

### C.93 *OTF2\_MarkerRef* DefMarker

Group markers by name and severity.

#### Attributes

const char*	marker-Group	Group name, e.g., "MUST", ...
const char*	marker-Category	Marker category, e.g., "Argument type error", ...
<i>OTF2_MarkerSeverity</i>	severity	The severity for these markers.

#### See also

*OTF2\_MarkerWriter\_WriteDefMarker()*

#### Since

Version 1.2

## C.94 Marker

A user marker instance, with implied time stamp.

#### Attributes

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

#### See also

*OTF2\_MarkerWriter\_WriteMarker()*

## C.97 SnapshotEnd

---

### Since

Version 1.2

## C.95 List of all snapshot records

## C.96 SnapshotStart

This record marks the start of a snapshot.

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
uint64_t	num-berOfRecord	Number of snapshot event records in this snapshot. Excluding the <a href="#">SnapshotEnd</a> record.

### See also

[OTF2\\_SnapWriter\\_SnapshotStart\(\)](#)

### Since

Version 1.2

## C.97 SnapshotEnd

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use [OTF2\\_EvtReader\\_Seek](#) with `contReadPos` as the position.

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

### See also

[OTF2\\_SnapWriter\\_SnapshotEnd\(\)](#)

### Since

Version 1.2

## C.98 MeasurementOnOffSnap

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.
<a href="#">OTF2_MeasurementMode</a>	measurementMode	Is the measurement turned on ( <a href="#">OTF2_MEASUREMENT_ON</a> ) or off ( <a href="#">OTF2_MEASUREMENT_OFF</a> )?

### See also

*MeasurementOnOff* event  
[OTF2\\_SnapWriter\\_MeasurementOnOff\(\)](#)

### Since

Version 1.2

## C.99 EnterSnap

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

### Attributes

## C.100 MpiSendSnap

---

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.
<a href="#">OTF2_RegionRef</a>	region	Needs to be defined in a definition record References a <a href="#">Region</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_REGION</a> is available.

### See also

[Enter](#) event

[OTF2\\_SnapWriter\\_Enter\(\)](#)

### Since

Version 1.2

## C.100 MpiSendSnap

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.
uint32_t	receiver	MPI rank of receiver in communicator.
<a href="#">OTF2_CommRef</a>	communicator	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length

**See also**

[MpiSend](#) event  
[OTF2\\_SnapWriter\\_MpiSend\(\)](#)

**Since**

Version 1.2

## C.101 MpisendSnap

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

**Attributes**

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

**See also**

[MpiIsend](#) event  
[OTF2\\_SnapWriter\\_MpiIsend\(\)](#)

**Since**

Version 1.2



## C.103 MpiRecvSnap

---

### C.102 MpisendCompleteSnap

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

#### Attributes

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

#### See also

*MpiSendComplete* event  
*OTF2\_SnapWriter\_MpiSendComplete()*

#### Since

Version 1.2

## C.103 MpiRecvSnap

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

#### Attributes

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

uint32_t	msgTag	Message tag
uint64_t	msgLength	Message length

**See also**

[MpiRecv](#) event  
[OTF2\\_SnapWriter\\_MpiRecv\(\)](#)

**Since**

Version 1.2

## C.104 MpiIrecvRequestSnap

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

**Attributes**

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint64_t	requestID	ID of the requested receive

**See also**

[MpiIrecvRequest](#) event  
[OTF2\\_SnapWriter\\_MpiIrecvRequest\(\)](#)

**Since**

Version 1.2

## C.106 MpiCollectiveBeginSnap

---

### C.105 MpiIrecvSnap

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

#### Attributes

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

#### See also

[MpiIrecv](#) event  
[OTF2\\_SnapWriter\\_MpiIrecv\(\)](#)

#### Since

Version 1.2

## C.106 MpiCollectiveBeginSnap

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

#### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
----------------------------------	----------	-------------------------------

<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.

**See also**

[\*MpiCollectiveBegin\*](#) event  
[OTF2\\_SnapWriter\\_MpiCollectiveBegin\(\)](#)

**Since**

Version 1.2

## C.107 MpiCollectiveEndSnap

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

**Attributes**

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

**See also**

[\*MpiCollectiveEnd\*](#) event  
[OTF2\\_SnapWriter\\_MpiCollectiveEnd\(\)](#)

## C.109 OmpAcquireLockSnap

---

### Since

Version 1.2

## C.108 OmpForkSnap

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

### Attributes

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location of the snapshot.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint32_t	num-berOfRe-quest-edThreads	Requested size of the team.

### See also

[\*OmpFork\*](#) event  
[\*OTF2\\_SnapWriter\\_OmpFork\(\)\*](#)

### Since

Version 1.2

## C.109 OmpAcquireLockSnap

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

### Attributes

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location of the snapshot.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint32_t	lockID	ID of the lock.

uint32_t	acquisitionOrder	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.
----------	------------------	---

**See also**

[\*OmpAcquireLock\*](#) event  
[OTF2\\_SnapWriter\\_OmpAcquireLock\(\)](#)

**Since**

Version 1.2

### C.110 OmpTaskCreateSnap

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

**Attributes**

<a href="#"><i>OTF2_LocationRef</i></a>	location	The location of the snapshot.
<a href="#"><i>OTF2_TimeStamp</i></a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint64_t	taskID	Identifier of the newly created task instance.

**See also**

[\*OmpTaskCreate\*](#) event  
[OTF2\\_SnapWriter\\_OmpTaskCreate\(\)](#)

**Since**

Version 1.2

## C.112 MetricSnap

---

### C.111 OmpTaskSwitchSnap

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

#### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happended.
uint64_t	taskID	Identifier of the now active task instance.

#### See also

[OmpTaskSwitch](#) event  
[OTF2\\_SnapWriter\\_OmpTaskSwitch\(\)](#)

#### Since

Version 1.2

### C.112 MetricSnap

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

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

#### Attributes

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

## APPENDIX C. MODULE DOCUMENTATION

uint8_t	numberOfMetrics	Number of metrics with in the set.
OTF2_Type	typeIDs [ numberOfMetrics ]	List of metric types. These types must match that of the corresponding <i>MetricMember</i> definitions.
OTF2_MetricValue	metricValues [ numberOfMetrics ]	List of metric values.

### See also

*Metric* event  
`OTF2_SnapWriter_Metric()`

### Since

Version 1.2

## C.113 ParameterStringSnap

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

### Attributes

<i>OTF2_LocationRef</i>	location	The location of the snapshot.
<i>OTF2_TimeStamp</i>	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEventTime	The original time this event happened.
<i>OTF2_ParameterRef</i>	parameter	Parameter ID. References a <i>Parameter</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_PARAMETER</i> is available.
<i>OTF2_StringRef</i>	string	Value: Handle of a string definition References a <i>String</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_STRING</i> is available.



## C.115 ParameterUnsignedIntSnap

---

### See also

[ParameterString](#) event  
[OTF2\\_SnapWriter\\_ParameterString\(\)](#)

### Since

Version 1.2

## C.114 ParameterIntSnap

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

### Attributes

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

### See also

[ParameterInt](#) event  
[OTF2\\_SnapWriter\\_ParameterInt\(\)](#)

### Since

Version 1.2

## C.115 ParameterUnsignedIntSnap

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

**Attributes**

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

**See also**

[\*ParameterUnsignedInt\*](#) event  
[\*OTF2\\_SnapWriter\\_ParameterUnsignedInt\(\)\*](#)

**Since**

Version 1.2

**C.116 OTF2 usage examples****Modules**

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

**C.117 Usage in writing mode - a simple example**

This is a short example of how to use the OTF2 writing interface. This example is available as source code in the file `otf2_writer_example.c`.

First include the OTF2 header.

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

For this example an additional include statement is necessary.

### C.117 Usage in writing mode - a simple example

---

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

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

```
static OTF2_TimeStamp
get_time( void )
{
    static uint64_t sequence;
    return sequence++;
}
```

Define a pre and post flush callback. If no memory is left in OTF2's internal memory buffer or the writer handle is closed a memory buffer flushing routine is triggered. The pre flush callback is triggered right before a buffer flush. It needs to return either `OTF2_FLUSH` to flush the recorded data to a file or `OTF2_NO_FLUSH` to suppress flushing data to a file. The post flush callback is triggered right after a memory buffer flush. It has to return a current timestamp which is recorded to mark the time spent in a buffer flush. The callbacks are passed via a struct to OTF2.

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

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

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

Now everything is prepared to begin with the main program.

```
int
main( int    argc,
```

---

## APPENDIX C. MODULE DOCUMENTATION

---

```
char** argv )  
{
```

Create new archive handle.

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

Set the previously defined flush callbacks.

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

We will operate in an serial context.

```
OTF2_Archive_SetSerialCollectiveCallbacks( archive );
```

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

```
OTF2_Archive_OpenEvtFiles( archive );
```

Get a local event writer for location 0.

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

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

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

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

### C.117 Usage in writing mode - a simple example

---

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

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

```
OTF2_Archive_CloseEvtFiles( archive );
```

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

```
OTF2_GlobalDefWriter* global_def_writer = OTF2_Archive_GetGlobalDefWriter( archive );
```

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

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

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

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

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

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

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

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

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

At the end, close the archive and exit.

```
OTF2_Archive_Close( archive );

return EXIT_SUCCESS;
}
```

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

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

Now you can link your program with:

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

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

---

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

First create an attribute list handle.

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

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

```
OTF2_AttributeValue attr_value;  
attr_value.uint32 = attribute_value;  
OTF2_AttributeList_AddAttribute( attribute_list, attribute_id, OTF2_UINT8, attr  
_value );  
...
```

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

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

## C.119 OTF2 callbacks

### Modules

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

## C.120 Controlling OTF2 flush behavior in writing mode

### Data Structures

- struct [OTF2\\_FlushCallbacks](#)

*Structure holding the flush callbacks.*

### Typedefs

- typedef [OTF2\\_TimeStamp](#)(\* [OTF2\\_PostFlushCallback](#))(void \*userData, [OTF2\\_FileType](#) fileType, [OTF2\\_LocationRef](#) location)

*Definition for the post flush callback.*

- typedef [OTF2\\_FlushType](#)(\* [OTF2\\_PreFlushCallback](#))(void \*userData, [OTF2\\_FileType](#) fileType, [OTF2\\_LocationRef](#) location, void \*callerData, bool final)

*Definition for the pre flush callback.*

#### C.120.1 Detailed Description

The flushing behavior from OTF2 can be controlled via callbacks. Calling [OTF2\\_Archive\\_SetFlushCallbacks](#) is mandatory when writing and erroneous when reading an archive.

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

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

#### C.120.2 Typedef Documentation

##### C.120.2.1 typedef OTF2\_TimeStamp( \* OTF2\_PostFlushCallback)(void \*userData, OTF2\_FileType fileType, OTF2\_LocationRef location)

Definition for the post flush callback.

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

#### Parameters

<i>userData</i>	Data passed to the call <a href="#">OTF2_Archive_SetFlushCallbacks</a> .
<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 <a href="#">OTF2_UNDEFINED_LOCATION</a> ).

#### Returns

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



## C.121 Memory pooling for OTF2

---

**C.120.2.2** `typedef OTF2_FlushType( * OTF2_PreFlushCallback)(void *userData, OTF2_FileType fileType, OTF2_LocationRef location, void *callerData, bool final)`

Definition for the pre flush callback.

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

### Parameters

<i>userData</i>	Data passed to the call <a href="#">OTF2_Archive_SetFlushCallbacks</a> .
<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 <a href="#">OTF2_FILETYPE_LOCAL_DEFS</a> or <a href="#">OTF2_FILETYPE_EVENTS</a> . For other files this is <a href="#">OTF2_UNDEFINED_LOCATION</a> . A special case exists for files of type <a href="#">OTF2_FILETYPE_EVENTS</a> in writing mode. The location ID may still be <a href="#">OTF2_UNDEFINED_LOCATION</a> . 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 <a href="#">OTF2_EvtWriter_SetLocationID()</a> and utilizing the <i>callerData</i> argument.
<i>callerData</i>	Depending of the fileType, this can be an <a href="#">OTF2_EvtWriter</a> , <a href="#">OTF2_GlobalDefWriter</a> , <a href="#">OTF2_DefWriter</a> .
<i>final</i>	Indicates whether this is the final flush when closing the writer objects.

### Returns

Returns [OTF2\\_FLUSH](#) or [OTF2\\_NO\\_FLUSH](#).

## C.121 Memory pooling for OTF2

### Data Structures

- struct [OTF2\\_MemoryCallbacks](#)  
*Structure holding the memory callbacks.*

### Typedefs

- typedef void \*(\* [OTF2\\_MemoryAllocate](#) )(void \*userData, OTF2\_FileType fileType, OTF2\_LocationRef location, void \*\*perBufferData, uint64\_t chunkSize)  
*Function pointer for allocating memory for chunks.*

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

*Function pointer to release all allocated chunks.*

### C.121.1 Detailed Description

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

### C.121.2 Typedef Documentation

- C.121.2.1 `typedef void*( * OTF2_MemoryAllocate)(void *userData, OTF2_FileType fileType, OTF2_LocationRef location, void **perBufferData, uint64_t chunkSize)`

Function pointer for allocating memory for chunks.

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

This function must return a pointer to a valid allocated memory location (just like malloc). This memory location must be of exact same size as the parameter 'chunkSize' provided with [OTF2\\_Archive\\_Open\(\)](#).

#### Parameters

<i>userData</i>	Data passed to the call <a href="#">OTF2_Archive_SetMemoryCallbacks</a> .
<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 <a href="#">OTF2_UNDEFINED_LOCATION</a> ).
<i>perBufferData</i>	A writable pointer to store callee data. For the first call this will be NULL.
<i>chunkSize</i>	The size of the requested chunk.

#### Returns

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

## C.122 Operating OTF2 in an collective context

---

**C.121.2.2** `typedef void( * OTF2_MemoryFreeAll)(void *userData, OTF2_FileType fileType, OTF2_LocationRef location, void **perBufferData, bool final)`

Function pointer to release all allocated chunks.

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

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

### Parameters

<i>userData</i>	Data passed to the call <a href="#">OTF2_Archive_SetMemoryCallbacks</a> .
<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 <a href="#">OTF2_UNDEFINED_LOCATION</a> ).
<i>perBufferData</i>	A writable pointer to store callee data. For the first call this will be NULL.
<i>final</i>	Indicates whether this is the final free when closing the writer objects. <code>perBufferData</code> should be handled than.

## C.122 Operating OTF2 in an collective context

### Data Structures

- struct [OTF2\\_CollectiveCallbacks](#)  
*Struct which holds all collective callbacks.*

### Typedefs

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_Barrier](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext)  
*Performs an barrier collective on the given communication context.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_Bcast](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext, void \*data, uint32\_t numberElements, [OTF2\\_Type](#) type, uint32\_t root)  
*Performs an broadcast collective on the given communication context.*

- `typedef OTF2_CallbackCode(* OTF2_Collectives_CreateLocalComm)(void *userData, OTF2_CollectiveContext **localCommContext, OTF2_CollectiveContext *globalCommContext, uint32_t globalRank, uint32_t globalSize, uint32_t localRank, uint32_t localSize, uint32_t fileNumber, uint32_t numberOfFiles)`  
*Create a new disjoint partitioning of the the globalCommContext communication context. numberOfFiles denotes the number of the partitions. fileNumber denotes in which of the partitions this OTF2\_Archive should belong. localSize is the size of this partition and localRank the rank of this OTF2\_Archive in the partition.*
- `typedef OTF2_CallbackCode(* OTF2_Collectives_FreeLocalComm)(void *userData, OTF2_CollectiveContext *localCommContext)`  
*Destroys the communication context previous created by the OTF2\_Collectives\_CreateLocalComm callback.*
- `typedef OTF2_CallbackCode(* OTF2_Collectives_Gather)(void *userData, OTF2_CollectiveContext *commContext, const void *inData, void *outData, uint32_t numberElements, OTF2_Type type, uint32_t root)`  
*Performs an gather collective on the given communication context where each ranks contribute the same number of elements. outData is only valid at rank root.*
- `typedef OTF2_CallbackCode(* OTF2_Collectives_Gatherv)(void *userData, OTF2_CollectiveContext *commContext, const void *inData, uint32_t inElements, void *outData, const uint32_t *outElements, OTF2_Type type, uint32_t root)`  
*Performs an gather collective on the given communication context where each ranks contribute different number of elements. outData and outElements are only valid at rank root.*
- `typedef OTF2_CallbackCode(* OTF2_Collectives_GetRank)(void *userData, OTF2_CollectiveContext *commContext, uint32_t *rank)`  
*Returns the rank of this OTF2\_Archive objects in this communication context. A number between 0 and one less of the size of the communication context.*
- `typedef OTF2_CallbackCode(* OTF2_Collectives_GetSize)(void *userData, OTF2_CollectiveContext *commContext, uint32_t *size)`  
*Returns the number of OTF2\_Archive objects operating in this communication context.*
- `typedef void(* OTF2_Collectives_Release)(void *userData, OTF2_CollectiveContext *globalCommContext, OTF2_CollectiveContext *localCommContext)`  
*Optionally called in OTF2\_Archive\_Close or OTF2\_Reader\_Close respectively.*
- `typedef OTF2_CallbackCode(* OTF2_Collectives_Scatter)(void *userData, OTF2_CollectiveContext *commContext, const void *inData, void *outData, uint32_t numberElements, OTF2_Type type, uint32_t root)`  
*Performs an scatter collective on the given communication context where each ranks contribute the same number of elements. inData is only valid at rank root.*

## C.122 Operating OTF2 in an collective context

---

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_Scatterv](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext, const void \*inData, const uint32\_t \*inElements, void \*outData, uint32\_t outElements, [OTF2\\_Type](#) type, uint32\_t root)

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

### C.122.1 Detailed Description

To operate multiple [OTF2\\_Archive](#) objects in an collective context, the following callbacks need to be implemented. These are mandatory, when writing an trace file with multiple [OTF2\\_Archive](#) objects. For reading a set of serial callbacks are provided (See [OTF2\\_Archive\\_SetSerialCollectiveCallbacks](#) and [OTF2\\_Reader\\_SetSerialCollectiveCallbacks](#)). The struct [OTF2\\_CollectiveContext](#) needs to be declared too.

Only [OTF2\\_Type](#) of the integer and floating point category need to be considered as values when the callbacks are called.

Except for the [OTF2\\_Collectives\\_GetSize](#) and [OTF2\\_Collectives\\_GetRank](#) callbacks, the return value must always be the same for all participating tasks. In particular all calls should either return [OTF2\\_CALLBACK\\_SUCCESS](#) or [!OTF2\\_CALLBACK\\_SUCCESS](#), but it is undefined, if some of the calls return [OTF2\\_CALLBACK\\_SUCCESS](#) and other [!OTF2\\_CALLBACK\\_SUCCESS](#).

The [OTF2\\_Collectives\\_CreateLocalComm](#) and [OTF2\\_Collectives\\_FreeLocalComm](#) are ignored when writing and optional when reading, but than both are mandatory. These are used to created the same local communication context as was given at writing time, if possible.

On the contrary the *localCommContext* to [OTF2\\_Archive\\_SetCollectiveCallbacks](#) is ignored when reading and optional (i.e., not NULL) when writing. It determines the number of files to use when the SION substrate is used. these *localCommContext* must be an disjoint partitioning of the used *globalCommContext* than.

The [OTF2\\_Collectives\\_Release](#) is optional and will be called as one of the last actions before the [OTF2\\_Archive](#) or the [OTF2\\_Reader](#) will be closed.

### C.122.2 Typedef Documentation

#### C.122.2.1 typedef [OTF2\\_CallbackCode](#)( \* [OTF2\\_Collectives\\_Barrier](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext)

Performs an barrier collective on the given communication context.

**Since**

Version 1.3

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

**C.122.2.2** `typedef OTF2_CallbackCode( * OTF2_Collectives_Bcast)(void  
*userData, OTF2_CollectiveContext *commContext, void *data, uint32_t  
numberElements, OTF2_Type type, uint32_t root)`

Performs an broadcast collective on the given communication context.

**Since**

Version 1.3

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

**C.122.2.3** `typedef OTF2_CallbackCode( * OTF2_Collectives_-  
CreateLocalComm)(void *userData, OTF2_CollectiveContext  
**localCommContext, OTF2_CollectiveContext *globalCommContext,  
uint32_t globalRank, uint32_t globalSize, uint32_t localRank, uint32_t localSize,  
uint32_t fileNumber, uint32_t numberOfFiles)`

Create a new disjoint partitioning of the the *globalCommContext* communication context. *numberOfFiles* denotes the number of the partitions. *fileNumber* denotes in which of the partitions this OTF2\_Archive should belong. *localSize* is the size of this partition and *localRank* the rank of this OTF2\_Archive in the partition.

**Since**

Version 1.3

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

## C.122 Operating OTF2 in an collective context

---

**C.122.2.4** `typedef OTF2_CallbackCode( * OTF2_Collectives_-  
FreeLocalComm)(void *userData, OTF2_CollectiveContext  
*localCommContext)`

Destroys the communication context previous created by the *OTF2\_Collectives\_-CreateLocalComm* callback.

### Since

Version 1.3

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

**C.122.2.5** `typedef OTF2_CallbackCode( * OTF2_Collectives_Gather)(void  
*userData, OTF2_CollectiveContext *commContext, const void *inData,  
void *outData, uint32_t numberElements, OTF2_Type type, uint32_t root)`

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

### Since

Version 1.3

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

**C.122.2.6** `typedef OTF2_CallbackCode( * OTF2_Collectives_Gatherv)(void  
*userData, OTF2_CollectiveContext *commContext, const void *inData,  
uint32_t inElements, void *outData, const uint32_t *outElements, OTF2_Type  
type, uint32_t root)`

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

### Since

Version 1.3

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

**C.122.2.7** `typedef OTF2_CallbackCode( * OTF2_Collectives_GetRank)(void  
*userData, OTF2_CollectiveContext *commContext, uint32_t *rank)`

Returns the rank of this OTF2\_Archive objects in this communication context. A number between 0 and one less of the size of the communication context.

**Since**

Version 1.3

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

**C.122.2.8** `typedef OTF2_CallbackCode( * OTF2_Collectives_GetSize)(void  
*userData, OTF2_CollectiveContext *commContext, uint32_t *size)`

Returns the number of OTF2\_Archive objects operating in this communication context.

**Since**

Version 1.3

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

**C.122.2.9** `typedef void( * OTF2_Collectives_Release)(void *userData, OTF2_-  
CollectiveContext *globalCommContext, OTF2_CollectiveContext  
*localCommContext)`

Optionally called in *OTF2\_Archive\_Close* or *OTF2\_Reader\_Close* respectively.

**Since**

Version 1.3

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.



## C.123 Usage in reading mode - MPI example

---

**C.122.2.10** `typedef OTF2_CallbackCode( * OTF2_Collectives_Scatter)(void  
*userData, OTF2_CollectiveContext *commContext, const void *inData,  
void *outData, uint32_t numberElements, OTF2_Type type, uint32_t root)`

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

### Since

Version 1.3

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

**C.122.2.11** `typedef OTF2_CallbackCode( * OTF2_Collectives_Scatterv)(void  
*userData, OTF2_CollectiveContext *commContext, const void *inData,  
const uint32_t *inElements, void *outData, uint32_t outElements, OTF2_Type  
type, uint32_t root)`

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

### Since

Version 1.3

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_ERROR*.

## C.123 Usage in reading mode - MPI example

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

We start with inclusion of some standard headers.

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

And then include the MPI and OTF2 header.

```
#include <mpi.h>

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

Now prepare the inclusion of the [<otf2/OTF2\\_MPI\\_Collectives.h>](#) header. As it is an header-only interface, it needs some information about the used MPI environment. In particular the MPI datatypes which match the C99 types `uint64_t` and `int64_t`. In case you have an MPI 3.0 conforming MPI implementation you can skip this. If not, provide `#define`'s for the following macros prior the `#include` statement. In this example, we assume an LP64 platform.

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

After this preparatory step, we can include the [<otf2/OTF2\\_MPI\\_Collectives.h>](#) header.

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

The following section until describing `main` is the same as in the [Usage in reading mode - a simple example](#).

Define an event callback for entering and leaving a region.

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

    return OTF2_CALLBACK_SUCCESS;
}

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

### C.123 Usage in reading mode - MPI example

---

```
        region, location, time );

    return OTF2_CALLBACK_SUCCESS;
}
```

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

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

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

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

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

    return OTF2_CALLBACK_SUCCESS;
}
```

Now everything is prepared to begin with the main program.

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

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

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

```
int rank;
MPI_Comm_rank( MPI_COMM_WORLD, &rank );
```

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

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

Now we provide the OTF2 reader object the MPI collectives.

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

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

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

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

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

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

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

### C.123 Usage in reading mode - MPI example

---

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

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

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

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

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

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

When the files are opened the event and definition reader handle can be requested. We distribute the locations round-robin to all ranks in `MPI_COMM_WORLD`. To apply mapping tables stored in the local definitions, the local definitions must be read. Though the existence of these files are optional. The call to [\*OTF2\\_Reader\\_GetEvtReader\*](#) is mandatory, but the result is unused.

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

    if ( successful_open_def_files )
    {
```

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

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

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

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

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

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

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

Read all events in the OTF2 archive. The events are automatically ordered by the time they occurred in the trace. Everytime an enter or leave event is read, the

### C.123 Usage in reading mode - MPI example

---

previously registered callbacks are triggered. In `events_read` the number of read events is returned.

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

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

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

As the call to *[OTF2\\_Reader\\_CloseEvtFiles](#)* is an collective operation all ranks need to call this, not only those which read events.

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

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

```
OTF2_Reader_Close( reader );

free( locations );

MPI_Finalize();

return EXIT_SUCCESS;
}
```

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

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

Now you can link your program with:

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

## **C.124 Usage in writing mode - MPI example**

This is a short example of how to use the OTF2 writing interface with MPI. This example is available as source code in the file `otf2_mpi_writer_example.c`. We start with inclusion of some standard headers.

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

And then include the MPI and OTF2 header.

```
#include <mpi.h>

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

Now prepare the inclusion of the [<otf2/OTF2\\_MPI\\_Collectives.h>](#) header. As it is an header-only interface, it needs some information about the used MPI environment. In particular the MPI datatypes which match the C99 types `uint64_t` and `int64_t`. In case you have an MPI 3.0 conforming MPI implementation you can skip this. If not, provide `#define`'s for the following macros prior the `#include` statement. In this example, we assume an LP64 platform.

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

After this preparatory step, we can include the [<otf2/OTF2\\_MPI\\_Collectives.h>](#) header.

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

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

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

Define a pre and post flush callback. If no memory is left in OTF2's internal memory buffer or the writer handle is closed a memory buffer flushing routine is triggered. The pre flush callback is triggered right before a buffer flush. It needs



## C.124 Usage in writing mode - MPI example

---

to return either OTF2\_FLUSH to flush the recorded data to a file or OTF2\_NO\_FLUSH to suppress flushing data to a file. The post flush callback is triggered right after a memory buffer flush. It has to return a current timestamp which is recorded to mark the time spent in a buffer flush. The callbacks are passed via a struct to OTF2.

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

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

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

Now everything is prepared to begin with the main program.

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

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

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

Create new archive handle.

```
    OTF2_Archive* archive = OTF2_Archive_Open( "ArchivePath",
                                                "ArchiveName",
```

## APPENDIX C. MODULE DOCUMENTATION

---

```
OTF2_FILEMODE_WRITE,  
1024 * 1024 /* event chunk size */  
  
,  
  
4 * 1024 * 1024 /* def chunk size  
*/,  
  
OTF2_SUBSTRATE_POSIX,  
OTF2_COMPRESSION_NONE );
```

Set the previously defined flush callbacks.

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

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

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

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

```
OTF2_Archive_OpenEvtFiles( archive );
```

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

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

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

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

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

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

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

## C.124 Usage in writing mode - MPI example

---

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

Now we can do the `MPI_Barrier` call.

```
MPI_Barrier( MPI_COMM_WORLD );
```

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

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

Finally we leave the region again with the `leave region`.

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

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

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

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

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

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

```
OTF2_Archive_CloseEvtFiles( archive );
```

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

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

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

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

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

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

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

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

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

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

```
OTF2_GlobalDefWriter_WriteRegion( global_def_writer,
                                  0 /* id */,
```

---

## C.124 Usage in writing mode - MPI example

---

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

Write the system tree to the global definition writer.

```
OTF2_GlobalDefWriter_WriteSystemTreeNode( global_def_writer,  
                                          0 /* id */,  
                                          5 /* name */,  
                                          6 /* class */,  
  
OTF2_UNDEFINED_SYSTEM_TREE_NODE /* parent */ );
```

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

```
for ( int r = 0; r < size; r++ )  
{  
    char process_name[ 32 ];  
    sprintf( process_name, "MPI Rank %d", r );  
    OTF2_GlobalDefWriter_WriteString( global_def_writer,  
                                      9 + r,  
                                      process_name );  
  
    OTF2_GlobalDefWriter_WriteLocationGroup( global_def_writer,  
                                             r /* id */,  
                                             9 + r /* name */,  
  
OTF2_LOCATION_GROUP_TYPE_PROCESS,  
                                             0 /* system tree */ );  
  
    OTF2_GlobalDefWriter_WriteLocation( global_def_writer,  
                                        r /* id */,  
                                        1 /* name */,  
                                        OTF2_LOCATION_TYPE_CPU_THREAD,  
                                        4 /* # events */,  
                                        r /* location group */ );  
}
```

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

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

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

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

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

OTF2_Archive_CloseGlobalDefWriter( archive,
                                   global_def_writer );
}
```

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

```
MPI_Barrier( MPI_COMM_WORLD );
```

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

## C.125 Usage in reading mode - a simple example

---

```
OTF2_Archive_Close( archive );

MPI_Finalize();

return EXIT_SUCCESS;
}
```

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

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

Now you can link your program with:

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

## C.125 Usage in reading mode - a simple example

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

First include the OTF2 header.

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

For this example some additional include statements are necessary.

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

Define an event callback for entering and leaving a region.

```
static OTF2_CallbackCode
Enter_print( OTF2_LocationRef    location,
             OTF2_TimeStamp      time,
             void*               userData,
```

## APPENDIX C. MODULE DOCUMENTATION

---

```
        OTF2_AttributeList* attributes,
        OTF2_RegionRef      region )
{
    printf( "Entering region %u at location %" PRIu64 " at time %" PRIu64 ".\n",
            region, location, time );

    return OTF2_CALLBACK_SUCCESS;
}

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

    return OTF2_CALLBACK_SUCCESS;
}
```

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

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

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

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

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

    return OTF2_CALLBACK_SUCCESS;
}
```



## C.125 Usage in reading mode - a simple example

---

Now everything is prepared to begin with the main program.

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

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

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

We will operate in an serial context.

```
OTF2_Reader_SetSerialCollectiveCallbacks( reader );
```

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

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

Get the global definition reader from the reader handle.

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

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

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

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

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

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

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

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

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

When the files are opened the event and definition reader handle can be requested. In this example for all. To apply mapping tables stored in the local definitions, the local definitions must be read. Though the existence of these files are optional. The call to *OTF2\_Reader\_GetEvtReader* is mandatory, but the result is unused.

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

## C.125 Usage in reading mode - a simple example

---

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

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

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

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

```
OTF2_GlobalEvtReader* global_evt_reader = OTF2_Reader_GetGlobalEvtReader( reader );
```

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

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

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

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

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

---

## APPENDIX C. MODULE DOCUMENTATION

---

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

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

```
OTF2_Reader_Close( reader );

free( locations );

return EXIT_SUCCESS;
}
```

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

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

Now you can link your program with:

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

## Appendix D

# Data Structure Documentation

### D.1 OTF2\_AttributeValue Union Reference

Value container for an attributes.

```
#include <otf2/OTF2_AttributeList.h>
```

#### Data Fields

- [OTF2\\_AttributeRef attributeRef](#)  
*References a [Attribute](#) definition and will be mapped to the global definition if a mapping table of type [OTF2\\_MAPPING\\_ATTRIBUTE](#) is available.*
- [OTF2\\_CommRef commRef](#)  
*References a [Comm](#) definition and will be mapped to the global definition if a mapping table of type [OTF2\\_MAPPING\\_COMM](#) is available.*
- float [float32](#)  
*Arbitrary value of type float.*
- double [float64](#)  
*Arbitrary value of type double.*
- [OTF2\\_GroupRef groupRef](#)  
*References a [Group](#) definition and will be mapped to the global definition if a mapping table of type [OTF2\\_MAPPING\\_GROUP](#) is available.*
- int16\_t [int16](#)  
*Arbitrary value of type int16\_t.*
- int32\_t [int32](#)  
*Arbitrary value of type int32\_t.*
- int64\_t [int64](#)  
*Arbitrary value of type int64\_t.*

- `int8_t` [int8](#)  
*Arbitrary value of type `int8_t`.*
- `OTF2_LocationRef` [locationRef](#)  
*References a [Location](#) definition and will be mapped to the global definition if a mapping table of type `OTF2_MAPPING_LOCATION` is available.*
- `OTF2_MetricRef` [metricRef](#)  
*References a [MetricClass](#), or a [MetricInstance](#) definition and will be mapped to the global definition if a mapping table of type `OTF2_MAPPING_METRIC` is available.*
- `OTF2_ParameterRef` [parameterRef](#)  
*References a [Parameter](#) definition and will be mapped to the global definition if a mapping table of type `OTF2_MAPPING_PARAMETER` is available.*
- `OTF2_RegionRef` [regionRef](#)  
*References a [Region](#) definition and will be mapped to the global definition if a mapping table of type `OTF2_MAPPING_REGION` is available.*
- `OTF2_RmaWinRef` [rmaWinRef](#)  
*References a [RmaWin](#) definition and will be mapped to the global definition if a mapping table of type `OTF2_MAPPING_RMA_WIN` is available.*
- `OTF2_StringRef` [stringRef](#)  
*References a [String](#) definition and will be mapped to the global definition if a mapping table of type `OTF2_MAPPING_STRING` is available.*
- `uint16_t` [uint16](#)  
*Arbitrary value of type `uint16_t`.*
- `uint32_t` [uint32](#)  
*Arbitrary value of type `uint32_t`.*
- `uint64_t` [uint64](#)  
*Arbitrary value of type `uint64_t`.*
- `uint8_t` [uint8](#)  
*Arbitrary value of type `uint8_t`.*

### D.1.1 Detailed Description

Value container for an attributes.

For definition references ([OTF2\\_MappingType](#)) use the same data type as the definition.

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

- [otf2/OTF2\\_AttributeList.h](#)

## D.2 OTF2\_CollectiveCallbacks Struct Reference

---

### D.2 OTF2\_CollectiveCallbacks Struct Reference

Struct which holds all collective callbacks.

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

#### D.2.1 Detailed Description

Struct which holds all collective callbacks.

##### Since

Version 1.3

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

- [otf2/OTF2\\_Callbacks.h](#)

### D.3 OTF2\_CollectiveContext Struct Reference

Collective context which wraps an MPI communicator.

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

#### D.3.1 Detailed Description

Collective context which wraps an MPI communicator.

User provided type for collective groups.

##### Since

Version 1.3

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

- [otf2/OTF2\\_MPI\\_Collectives.h](#)

### D.4 OTF2\_FlushCallbacks Struct Reference

Structure holding the flush callbacks.

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

### Data Fields

- [OTF2\\_PostFlushCallback otf2\\_post\\_flush](#)  
*Callback which is called after a flush.*
- [OTF2\\_PreFlushCallback otf2\\_pre\\_flush](#)  
*Callback which is called prior a flush.*

#### D.4.1 Detailed Description

Structure holding the flush callbacks.

To be used in a call to [OTF2\\_Archive\\_SetFlushCallbacks](#).

otf2\_post\_flush callback may be NULL to suppress writing a BufferFlush record.

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

- [otf2/OTF2\\_Callbacks.h](#)

### D.5 OTF2\_MemoryCallbacks Struct Reference

Structure holding the memory callbacks.

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

### Data Fields

- [OTF2\\_MemoryAllocate otf2\\_allocate](#)  
*Callback which is called to allocate a new chunk.*
- [OTF2\\_MemoryFreeAll otf2\\_free\\_all](#)  
*Callback which is called to release all previous allocated chunks.*

#### D.5.1 Detailed Description

Structure holding the memory callbacks.

To be used in a call to [OTF2\\_Archive\\_SetMemoryCallbacks](#).

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

- [otf2/OTF2\\_Callbacks.h](#)



## D.6 OTF2\_MetricValue Union Reference

---

### D.6 OTF2\_MetricValue Union Reference

Metric value.

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

#### D.6.1 Detailed Description

Metric value.

Wrapper for enum *OTF2\_MetricValue\_union*.

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

- [otf2/OTF2\\_Events.h](#)

### D.7 OTF2\_MPI\_UserData Struct Reference

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

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

#### D.7.1 Detailed Description

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

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

- [otf2/OTF2\\_MPI\\_Collectives.h](#)

## **APPENDIX D. DATA STRUCTURE DOCUMENTATION**

---

## Appendix E

# File Documentation

### E.1 otf2/OTF2\_ErrorCodes.h File Reference

Error codes and error handling.

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

#### Typedefs

- typedef [OTF2\\_ErrorCode](#)(\* [OTF2\\_ErrorCallback](#) )(void \*userData, const char \*file, uint64\_t line, const char \*function, [OTF2\\_ErrorCode](#) errorCode, const char \*msgFormatString, va\_list va)

#### Enumerations

- enum [OTF2\\_ErrorCode](#) {  
    [OTF2\\_DEPRECATED](#) = -3,  
    [OTF2\\_ABORT](#) = -2,  
    [OTF2\\_WARNING](#) = -1,  
    [OTF2\\_SUCCESS](#) = 0,  
    [OTF2\\_ERROR\\_INVALID](#) = 1,  
    [OTF2\\_ERROR\\_E2BIG](#),  
    [OTF2\\_ERROR\\_EACCES](#),  
    [OTF2\\_ERROR\\_EADDRNOTAVAIL](#),

OTF2\_ERROR\_EAFNOSUPPORT,  
OTF2\_ERROR\_EAGAIN,  
OTF2\_ERROR\_EALREADY,  
OTF2\_ERROR\_EBADF,  
OTF2\_ERROR\_EBADMSG,  
OTF2\_ERROR\_EBUSY,  
OTF2\_ERROR\_ECANCELED,  
OTF2\_ERROR\_ECHILD,  
OTF2\_ERROR\_ECONNREFUSED,  
OTF2\_ERROR\_ECONNRESET,  
OTF2\_ERROR\_EDEADLK,  
OTF2\_ERROR\_EDESTADDRREQ,  
OTF2\_ERROR\_EDOM,  
OTF2\_ERROR\_EDQUOT,  
OTF2\_ERROR\_EEXIST,  
OTF2\_ERROR\_EFAULT,  
OTF2\_ERROR\_EFBIG,  
OTF2\_ERROR\_EINPROGRESS,  
OTF2\_ERROR\_EINTR,  
OTF2\_ERROR\_EINVAL,  
OTF2\_ERROR\_EIO,  
OTF2\_ERROR\_EISCONN,  
OTF2\_ERROR\_EISDIR,  
OTF2\_ERROR\_ELOOP,  
OTF2\_ERROR\_EMFILE,  
OTF2\_ERROR\_EMLINK,  
OTF2\_ERROR\_EMMSGSIZE,  
OTF2\_ERROR\_EMULTIHOP,  
OTF2\_ERROR\_ENAMETOOLONG,  
OTF2\_ERROR\_ENETDOWN,  
OTF2\_ERROR\_ENETRESET,  
OTF2\_ERROR\_ENETUNREACH,  
OTF2\_ERROR\_ENFILE,

## E.1 otf2/OTF2\_ErrorCodes.h File Reference

---

OTF2\_ERROR\_ENOBUFS,  
OTF2\_ERROR\_ENODATA,  
OTF2\_ERROR\_ENODEV,  
OTF2\_ERROR\_ENOENT,  
OTF2\_ERROR\_ENOEXEC,  
OTF2\_ERROR\_ENOLCK,  
OTF2\_ERROR\_ENOLINK,  
OTF2\_ERROR\_ENOMEM,  
OTF2\_ERROR\_ENOMSG,  
OTF2\_ERROR\_ENOPROTOOPT,  
OTF2\_ERROR\_ENOSPC,  
OTF2\_ERROR\_ENOSR,  
OTF2\_ERROR\_ENOSTR,  
OTF2\_ERROR\_ENOSYS,  
OTF2\_ERROR\_ENOTCONN,  
OTF2\_ERROR\_ENOTDIR,  
OTF2\_ERROR\_ENOTEMPTY,  
OTF2\_ERROR\_ENOTSOCK,  
OTF2\_ERROR\_ENOTSUP,  
OTF2\_ERROR\_ENOTTY,  
OTF2\_ERROR\_ENXIO,  
OTF2\_ERROR\_EOPNOTSUPP,  
OTF2\_ERROR\_EOVERFLOW,  
OTF2\_ERROR\_EPERM,  
OTF2\_ERROR\_EPIPE,  
OTF2\_ERROR\_EPROTO,  
OTF2\_ERROR\_EPROTONOSUPPORT,  
OTF2\_ERROR\_EPROTOTYPE,  
OTF2\_ERROR\_ERANGE,  
OTF2\_ERROR\_EROFS,  
OTF2\_ERROR\_ESPIPE,  
OTF2\_ERROR\_ESRCH,  
OTF2\_ERROR\_ESTALE,

---

## APPENDIX E. FILE DOCUMENTATION

---

OTF2\_ERROR\_ETIME,  
OTF2\_ERROR\_ETIMEDOUT,  
OTF2\_ERROR\_ETXTBSY,  
OTF2\_ERROR\_EWOULDBLOCK,  
OTF2\_ERROR\_EXDEV,  
OTF2\_ERROR\_END\_OF\_FUNCTION,  
OTF2\_ERROR\_INVALID\_CALL,  
OTF2\_ERROR\_INVALID\_ARGUMENT,  
OTF2\_ERROR\_INVALID\_RECORD,  
OTF2\_ERROR\_INVALID\_DATA,  
OTF2\_ERROR\_INVALID\_SIZE\_GIVEN,  
OTF2\_ERROR\_UNKNOWN\_TYPE,  
OTF2\_ERROR\_INTEGRITY\_FAULT,  
OTF2\_ERROR\_MEM\_FAULT,  
OTF2\_ERROR\_MEM\_ALLOC\_FAILED,  
OTF2\_ERROR\_PROCESSED\_WITH\_FAULTS,  
OTF2\_ERROR\_INDEX\_OUT\_OF\_BOUNDS,  
OTF2\_ERROR\_INVALID\_LINENO,  
OTF2\_ERROR\_END\_OF\_BUFFER,  
OTF2\_ERROR\_FILE\_INTERACTION,  
OTF2\_ERROR\_FILE\_CAN\_NOT\_OPEN,  
OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK,  
OTF2\_ERROR\_PROPERTY\_NAME\_INVALID,  
OTF2\_ERROR\_PROPERTY\_EXISTS,  
OTF2\_ERROR\_PROPERTY\_NOT\_FOUND,  
OTF2\_ERROR\_PROPERTY\_VALUE\_INVALID,  
OTF2\_ERROR\_FILE\_COMPRESSION\_NOT\_SUPPORTED,  
OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE,  
OTF2\_ERROR\_INVALID\_FILE\_MODE\_TRANSITION,  
OTF2\_ERROR\_COLLECTIVE\_CALLBACK,  
OTF2\_ERROR\_FILE\_SUBSTRATE\_NOT\_SUPPORTED }

## E.1 otf2/OTF2\_ErrorCodes.h File Reference

---

### Functions

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

#### E.1.1 Detailed Description

Error codes and error handling.

#### E.1.2 Typedef Documentation

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

Signature of error handler callback functions. The error handler can be set with [OTF2\\_Error\\_RegisterCallback](#).

### Parameters

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

### Returns

Should return the errorCode

#### E.1.3 Enumeration Type Documentation

**E.1.3.1** `enum OTF2_ErrorCode`

This is the list of error codes for OTF2.

### Enumerator:

***OTF2\_DEPRECATED*** Special marker for error messages which indicates

an deprecation.

**OTF2\_ABORT** Special marker when the application will be aborted.

**OTF2\_WARNING** Special marker for error messages which are only warnings.

**OTF2\_SUCCESS** Operation successful

**OTF2\_ERROR\_INVALID** Invalid error code

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

**OTF2\_ERROR\_E2BIG** The list of arguments is too long

**OTF2\_ERROR\_EACCES** Not enough rights

**OTF2\_ERROR\_EADDRNOTAVAIL** Address is not available

**OTF2\_ERROR\_EAFNOSUPPORT** Address family is not supported

**OTF2\_ERROR\_EAGAIN** Resource temporarily not available

**OTF2\_ERROR\_EALREADY** Connection is already processed

**OTF2\_ERROR\_EBADF** Invalid file pointer

**OTF2\_ERROR\_EBADMSG** Invalid message

**OTF2\_ERROR\_EBUSY** Resource or device is busy

**OTF2\_ERROR\_ECANCELED** Operation was aborted

**OTF2\_ERROR\_ECHILD** No child process available

**OTF2\_ERROR\_ECONNREFUSED** Connection was refused

**OTF2\_ERROR\_ECONNRESET** Connection was reset

**OTF2\_ERROR\_EDEADLK** Resolved deadlock

**OTF2\_ERROR\_EDESTADDRREQ** Destination address was expected

**OTF2\_ERROR\_EDOM** Domain error

**OTF2\_ERROR\_EDQUOT** Reserved

**OTF2\_ERROR\_EEXIST** File does already exist

**OTF2\_ERROR\_EFAULT** Invalid Address

**OTF2\_ERROR\_EFBIG** File is too big

**OTF2\_ERROR\_EINPROGRESS** Operation is work in progress

**OTF2\_ERROR\_EINTR** Interruption of an operating system call

**OTF2\_ERROR\_EINVAL** Invalid argument

**OTF2\_ERROR\_EIO** Generic I/O error

**OTF2\_ERROR\_EISCONN** Socket is already connected

**OTF2\_ERROR\_EISDIR** Target is a directory

**OTF2\_ERROR\_ELOOP** Too many layers of symbolic links

**OTF2\_ERROR\_EMFILE** Too many opened files



## E.1 otf2/OTF2\_ErrorCodes.h File Reference

---

***OTF2\_ERROR\_EMLINK*** To many links

***OTF2\_ERROR\_MSGSIZE*** Message buffer is to small

***OTF2\_ERROR\_EMULTIHOP*** Reserved

***OTF2\_ERROR\_ENAMETOOLONG*** Filename is to long

***OTF2\_ERROR\_ENETDOWN*** Network is down

***OTF2\_ERROR\_ENETRESET*** Connection was reset from the network

***OTF2\_ERROR\_ENETUNREACH*** Network is not reachable

***OTF2\_ERROR\_ENFILE*** To much opened files

***OTF2\_ERROR\_ENOBUFS*** No buffer space available

***OTF2\_ERROR\_ENODATA*** No more data left in the queue

***OTF2\_ERROR\_ENODEV*** This device does not support this function

***OTF2\_ERROR\_ENOENT*** File or Directory does not exist

***OTF2\_ERROR\_ENOEXEC*** Cannot execute binary

***OTF2\_ERROR\_ENOLCK*** Locking failed

***OTF2\_ERROR\_ENOLINK*** Reserved

***OTF2\_ERROR\_ENOMEM*** Not enough main memory available

***OTF2\_ERROR\_ENOMSG*** Message has not the expected type

***OTF2\_ERROR\_ENOPROTOOPT*** This protocol is not available

***OTF2\_ERROR\_ENOSPC*** No space left on device

***OTF2\_ERROR\_ENOSR*** No stream available

***OTF2\_ERROR\_ENOSTR*** This is not a stream

***OTF2\_ERROR\_ENOSYS*** Requested function is not implemented

***OTF2\_ERROR\_ENOTCONN*** Socket is not connected

***OTF2\_ERROR\_ENOTDIR*** This is not an directory

***OTF2\_ERROR\_ENOTEMPTY*** This directory is not empty

***OTF2\_ERROR\_ENOTSOCK*** No socket

***OTF2\_ERROR\_ENOTSUP*** This operation is not supported

***OTF2\_ERROR\_ENOTTY*** This IOCTL is not supported by the device

***OTF2\_ERROR\_ENXIO*** Device is not yet configured

***OTF2\_ERROR\_EOPNOTSUPP*** Operation is not supported by this socket

***OTF2\_ERROR\_EOVERFLOW*** Value is to long for the datatype

***OTF2\_ERROR\_EPERM*** Operation is not permitted

***OTF2\_ERROR\_EPIPE*** Broken pipe

***OTF2\_ERROR\_EPROTO*** Protocoll error

---

## APPENDIX E. FILE DOCUMENTATION

---

***OTF2\_ERROR\_EPROTONOSUPPORT*** Protocoll is not supported

***OTF2\_ERROR\_EPROTOTYPE*** Wrong protocoll type for this socket

***OTF2\_ERROR\_ERANGE*** Value is out of range

***OTF2\_ERROR\_EROFS*** Filesystem is read only

***OTF2\_ERROR\_ESPIPE*** This seek is not allowed

***OTF2\_ERROR\_ESRCH*** No matching process found

***OTF2\_ERROR\_ESTALE*** Reserved

***OTF2\_ERROR\_ETIME*** Timeout in file stream or IOCTL

***OTF2\_ERROR\_ETIMEDOUT*** Connection timed out

***OTF2\_ERROR\_ETXTBSY*** File couldn't be executed while it is opened

***OTF2\_ERROR\_EWOULDBLOCK*** Operation would be blocking

***OTF2\_ERROR\_EXDEV*** Invalid link between devices

***OTF2\_ERROR\_END\_OF\_FUNCTION*** Unintentional reached end of function

***OTF2\_ERROR\_INVALID\_CALL*** Function call not allowed in current state

***OTF2\_ERROR\_INVALID\_ARGUMENT*** Parameter value out of range

***OTF2\_ERROR\_INVALID\_RECORD*** Invalid definition or event record

***OTF2\_ERROR\_INVALID\_DATA*** Invalid or inconsistent record data

***OTF2\_ERROR\_INVALID\_SIZE\_GIVEN*** The given size cannot be used

***OTF2\_ERROR\_UNKNOWN\_TYPE*** The given type is not known

***OTF2\_ERROR\_INTEGRITY\_FAULT*** The structural integrity is not given

***OTF2\_ERROR\_MEM\_FAULT*** This could not be done with the given memory

***OTF2\_ERROR\_MEM\_ALLOC\_FAILED*** Memory allocation failed

***OTF2\_ERROR\_PROCESSED\_WITH\_FAULTS*** An error appeared when data was processed

***OTF2\_ERROR\_INDEX\_OUT\_OF\_BOUNDS*** Index out of bounds

***OTF2\_ERROR\_INVALID\_LINENO*** Invalid source code line number

***OTF2\_ERROR\_END\_OF\_BUFFER*** End of buffer/file reached

***OTF2\_ERROR\_FILE\_INTERACTION*** Invalid file operation

***OTF2\_ERROR\_FILE\_CAN\_NOT\_OPEN*** Unable to open file

***OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK*** Record reading interrupted by reader callback

***OTF2\_ERROR\_PROPERTY\_NAME\_INVALID*** Property name does not conform to the naming scheme

## E.1 otf2/OTF2\_ErrorCodes.h File Reference

---

***OTF2\_ERROR\_PROPERTY\_EXISTS*** Property already exists

***OTF2\_ERROR\_PROPERTY\_NOT\_FOUND*** Property not found found in this archive

***OTF2\_ERROR\_PROPERTY\_VALUE\_INVALID*** Property value does not have the expected value

***OTF2\_ERROR\_FILE\_COMPRESSION\_NOT\_SUPPORTED*** Missing library support for requested compression mode

***OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE*** Multiple definitions for the same mapping type

***OTF2\_ERROR\_INVALID\_FILE\_MODE\_TRANSITION*** File mode transition not permitted

***OTF2\_ERROR\_COLLECTIVE\_CALLBACK*** Collective callback failed

***OTF2\_ERROR\_FILE\_SUBSTRATE\_NOT\_SUPPORTED*** Missing library support for requested file substrate

### E.1.4 Function Documentation

#### E.1.4.1 `const char* OTF2_Error_GetDescription ( OTF2_ErrorCode errorCode )`

Returns the description of an error code.

##### Parameters

<i>errorCode</i>	: Error Code
------------------	--------------

##### Returns

Returns the description of a known error code.

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

Returns the name of an error code.

##### Parameters

<i>errorCode</i>	: Error Code
------------------	--------------

##### Returns

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

**E.1.4.3 OTF2\_ErrorCallback OTF2\_Error\_RegisterCallback ( OTF2\_ErrorCallback errorCallbackIn, void \* userData )**

Register a programmers callback function for error handling.

**Parameters**

<i>errorCall- backIn</i>	: Fucntion will be called instead of printing a default message to standard error.
<i>userData</i>	: Data pointer passed to the callback.

**Returns**

Function pointer to the former error handling function.

**E.2 otf2/otf2.h File Reference**

Main include file for applications using OTF2.

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

**E.2.1 Detailed Description**

Main include file for applications using OTF2.

**E.3 otf2/OTF2\_Archive.h File Reference**

Writing interface for OTF2 archives.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Callbacks.h>
#include <otf2/OTF2_DefWriter.h>
#include <otf2/OTF2_DefReader.h>
#include <otf2/OTF2_EvtWriter.h>
#include <otf2/OTF2_EvtReader.h>
#include <otf2/OTF2_SnapWriter.h>
#include <otf2/OTF2_SnapReader.h>
#include <otf2/OTF2_GlobalDefWriter.h>
```

### E.3 otf2/OTF2\_Archive.h File Reference

---

```
#include <otf2/OTF2_GlobalDefReader.h>
#include <otf2/OTF2_GlobalEvtReader.h>
#include <otf2/OTF2_GlobalSnapReader.h>
#include <otf2/OTF2_Thumbnail.h>
#include <otf2/OTF2_MarkerWriter.h>
#include <otf2/OTF2_MarkerReader.h>
```

#### Defines

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

#### Typedefs

- `typedef struct OTF2_Archive_struct OTF2_Archive`  
*Keeps all meta-data for an OTF2 archive.*

#### Functions

- `OTF2_StatusCode OTF2_Archive_Close (OTF2_Archive *archive)`  
*Close an opened archive.*
- `OTF2_StatusCode OTF2_Archive_CloseDefFiles (OTF2_Archive *archive)`  
*Closes the local definitions file container.*
- `OTF2_StatusCode OTF2_Archive_CloseDefReader (OTF2_Archive *archive, OTF2_DefReader *reader)`  
*Close an opened local definition reader.*
- `OTF2_StatusCode OTF2_Archive_CloseDefWriter (OTF2_Archive *archive, OTF2_DefWriter *writer)`  
*Close an opened local definition writer.*
- `OTF2_StatusCode OTF2_Archive_CloseEvtFiles (OTF2_Archive *archive)`  
*Closes the events file container.*
- `OTF2_StatusCode OTF2_Archive_CloseEvtReader (OTF2_Archive *archive, OTF2_EvtReader *reader)`

*Close an opened local event reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseEvtWriter](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_EvtWriter](#) \*writer)

*Close an opened local event writer.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseGlobalDefReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_GlobalDefReader](#) \*globalDefReader)

*Closes the global definition reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseGlobalDefWriter](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_GlobalDefWriter](#) \*writer)

*Close an opened global definition writer.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseGlobalEvtReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_GlobalEvtReader](#) \*globalEvtReader)

*Closes the global event reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseGlobalSnapReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_GlobalSnapReader](#) \*globalSnapReader)

*Close the opened global snapshot reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseMarkerReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_MarkerReader](#) \*markerReader)

*Closes the marker reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseMarkerWriter](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_MarkerWriter](#) \*writer)

*Close an opened marker writer.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseSnapFiles](#) ([OTF2\\_Archive](#) \*archive)

*Closes the snapshots file container.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseSnapReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_SnapReader](#) \*reader)

*Close an opened local snap reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseSnapWriter](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_SnapWriter](#) \*writer)

*Close an opened local snap writer.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_CloseThumbReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_ThumbReader](#) \*reader)

*Close an opened thumbnail reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetChunkSize](#) ([OTF2\\_Archive](#) \*archive, [uint64\\_t](#) \*chunkSizeEvents, [uint64\\_t](#) \*chunkSizeDefs)

*Get the chunksize.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetCompression](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_Compression](#) \*compression)

*Get compression mode (none or zlib)*

### E.3 otF2/OTF2\_Archive.h File Reference

---

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetCreator](#) ([OTF2\\_Archive](#) \*archive, char \*\*creator)  
*Get creator information.*
- [OTF2\\_DefReader](#) \* [OTF2\\_Archive\\_GetDefReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_LocationRef](#) location)  
*Get a local definition reader.*
- [OTF2\\_DefWriter](#) \* [OTF2\\_Archive\\_GetDefWriter](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_LocationRef](#) location)  
*Get a local definition writer.*
- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetDescription](#) ([OTF2\\_Archive](#) \*archive, char \*\*description)  
*Get description.*
- [OTF2\\_EvtReader](#) \* [OTF2\\_Archive\\_GetEvtReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_LocationRef](#) location)  
*Get a local event reader.*
- [OTF2\\_EvtWriter](#) \* [OTF2\\_Archive\\_GetEvtWriter](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_LocationRef](#) location)  
*Get a local event writer.*
- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetFileSubstrate](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_FileSubstrate](#) \*substrate)  
*Get the file substrate (posix, sion, none)*
- [OTF2\\_GlobalDefReader](#) \* [OTF2\\_Archive\\_GetGlobalDefReader](#) ([OTF2\\_Archive](#) \*archive)  
*Get a global definition reader.*
- [OTF2\\_GlobalDefWriter](#) \* [OTF2\\_Archive\\_GetGlobalDefWriter](#) ([OTF2\\_Archive](#) \*archive)  
*Get a global definition writer.*
- [OTF2\\_GlobalEvtReader](#) \* [OTF2\\_Archive\\_GetGlobalEvtReader](#) ([OTF2\\_Archive](#) \*archive)  
*Get a global event reader.*
- [OTF2\\_GlobalSnapReader](#) \* [OTF2\\_Archive\\_GetGlobalSnapReader](#) ([OTF2\\_Archive](#) \*archive)  
*Get a global snap reader.*
- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetMachineName](#) ([OTF2\\_Archive](#) \*archive, char \*\*machineName)  
*Get machine name.*
- [OTF2\\_MarkerReader](#) \* [OTF2\\_Archive\\_GetMarkerReader](#) ([OTF2\\_Archive](#) \*archive)  
*Get a marker reader.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_MarkerWriter](#) \* [OTF2\\_Archive\\_GetMarkerWriter](#) ([OTF2\\_Archive](#) \*archive)

*Get a marker writer.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetNumberOfGlobalDefinitions](#) ([OTF2\\_Archive](#) \*archive, [uint64\\_t](#) \*numberOfDefinitions)

*Get the number of global definitions.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetNumberOfLocations](#) ([OTF2\\_Archive](#) \*archive, [uint64\\_t](#) \*numberOfLocations)

*Get the number of locations.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetNumberOfSnapshots](#) ([OTF2\\_Archive](#) \*archive, [uint32\\_t](#) \*number)

*Get the number of snapshots.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetNumberOfThumbnails](#) ([OTF2\\_Archive](#) \*archive, [uint32\\_t](#) \*number)

*Get the number of thumbnails.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetProperty](#) ([OTF2\\_Archive](#) \*archive, [const char](#) \*name, [char](#) \*\*value)

*Get the value of the named trace file property.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetPropertyNames](#) ([OTF2\\_Archive](#) \*archive, [uint32\\_t](#) \*numberOfProperties, [char](#) \*\*\*names)

*Get the names of all trace file properties.*

- [OTF2\\_SnapReader](#) \* [OTF2\\_Archive\\_GetSnapReader](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_LocationRef](#) location)

*Get a local snap reader.*

- [OTF2\\_SnapWriter](#) \* [OTF2\\_Archive\\_GetSnapWriter](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_LocationRef](#) location)

*Get a local snap writer.*

- [OTF2\\_ThumbReader](#) \* [OTF2\\_Archive\\_GetThumbReader](#) ([OTF2\\_Archive](#) \*archive, [uint32\\_t](#) number)

*Get a thumb reader.*

- [OTF2\\_ThumbWriter](#) \* [OTF2\\_Archive\\_GetThumbWriter](#) ([OTF2\\_Archive](#) \*archive, [const char](#) \*name, [const char](#) \*description, [OTF2\\_ThumbnailType](#) type, [uint32\\_t](#) numberOfSamples, [uint32\\_t](#) numberOfMetrics, [const uint64\\_t](#) \*refsToDefs)

*Get a thumb writer.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetTraceId](#) ([OTF2\\_Archive](#) \*archive, [uint64\\_t](#) \*id)

*Get the identifier of the trace file.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_GetVersion](#) ([OTF2\\_Archive](#) \*archive, [uint8\\_t](#) \*major, [uint8\\_t](#) \*minor, [uint8\\_t](#) \*bugfix)



### E.3 otf2/OTF2\_Archive.h File Reference

---

*Get format version.*

- [OTF2\\_Archive \\* OTF2\\_Archive\\_Open](#) (const char \*archivePath, const char \*archiveName, const [OTF2\\_FileMode](#) fileMode, const uint64\_t chunkSizeEvents, const uint64\_t chunkSizeDefs, const [OTF2\\_FileSubstrate](#) fileSubstrate, const [OTF2\\_Compression](#) compression)

*Create a new archive.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_OpenDefFiles](#) ([OTF2\\_Archive](#) \*archive)

*Open the local definitions file container.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_OpenEvtFiles](#) ([OTF2\\_Archive](#) \*archive)

*Open the events file container.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_OpenSnapFiles](#) ([OTF2\\_Archive](#) \*archive)

*Open the snapshots file container.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_SelectLocation](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_LocationRef](#) location)

*Select a location to be read.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_SetBoolProperty](#) ([OTF2\\_Archive](#) \*archive, const char \*name, bool value, bool overwrite)

*Add or remove a boolean trace file property to this archive.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_SetCollectiveCallbacks](#) ([OTF2\\_Archive](#) \*archive, const [OTF2\\_CollectiveCallbacks](#) \*collectiveCallbacks, void \*collectiveData, [OTF2\\_CollectiveContext](#) \*globalCommContext, [OTF2\\_CollectiveContext](#) \*localCommContext)

*Set the collective callbacks for the archive.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_SetCreator](#) ([OTF2\\_Archive](#) \*archive, const char \*creator)

*Set creator.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_SetDescription](#) ([OTF2\\_Archive](#) \*archive, const char \*description)

*Set a description.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_SetFlushCallbacks](#) ([OTF2\\_Archive](#) \*archive, const [OTF2\\_FlushCallbacks](#) \*flushCallbacks, void \*flushData)

*Set the flush callbacks for the archive.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_SetMachineName](#) ([OTF2\\_Archive](#) \*archive, const char \*machineName)

*Set machine name.*

- [OTF2\\_ErrorCode OTF2\\_Archive\\_SetMemoryCallbacks](#) ([OTF2\\_Archive](#) \*archive, const [OTF2\\_MemoryCallbacks](#) \*memoryCallbacks, void \*memoryData)

*Set the memory callbacks for the archive.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_SetNumberOfSnapshots](#) ([OTF2\\_Archive](#) \*archive, uint32\_t number)

*Set the number of snapshots.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_SetProperty](#) ([OTF2\\_Archive](#) \*archive, const char \*name, const char \*value, bool overwrite)

*Add or remove a trace file property to this archive.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_SetSerialCollectiveCallbacks](#) ([OTF2\\_Archive](#) \*archive)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_Archive\\_SwitchFileMode](#) ([OTF2\\_Archive](#) \*archive, [OTF2\\_FileMode](#) newFileMode)

*Switch file mode of the archive.*

### E.3.1 Detailed Description

Writing interface for OTF2 archives.

### E.3.2 Define Documentation

#### E.3.2.1 `#define OTF2_CHUNK_SIZE_DEFINITIONS_DEFAULT ( 4 * 1024 * 1024 )`

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

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

#### E.3.2.2 `#define OTF2_CHUNK_SIZE_EVENTS_DEFAULT ( 1024 * 1024 )`

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

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

### E.3.3 Typedef Documentation

#### E.3.3.1 `typedef struct OTF2_Archive_struct OTF2_Archive`

Keeps all meta-data for an OTF2 archive.

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

## E.3 otf2/OTF2\_Archive.h File Reference

---

### E.3.4 Function Documentation

#### E.3.4.1 OTF2\_ErrorCode OTF2\_Archive\_Close ( OTF2\_Archive \* *archive* )

Close an opened archive.

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

##### Parameters

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

##### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.3.4.2 OTF2\_ErrorCode OTF2\_Archive\_CloseDefFiles ( OTF2\_Archive \* *archive* )

Closes the local definitions file container.

This function is an collective operation.

##### Parameters

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

##### Since

Version 1.3

##### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.3.4.3 OTF2\_ErrorCode OTF2\_Archive\_CloseDefReader ( OTF2\_Archive \* *archive*, OTF2\_DefReader \* *reader* )

Close an opened local definition reader.

##### Parameters

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

---

## APPENDIX E. FILE DOCUMENTATION

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.4** **OTF2\_StatusCode** **OTF2\_Archive\_CloseDefWriter** ( **OTF2\_Archive** \* *archive*,  
**OTF2\_DefWriter** \* *writer* )

Close an opened local definition writer.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.5** **OTF2\_StatusCode** **OTF2\_Archive\_CloseEvtFiles** ( **OTF2\_Archive** \* *archive* )

Closes the events file container.

This function is an collective operation.

### Parameters

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

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.6** **OTF2\_StatusCode** **OTF2\_Archive\_CloseEvtReader** ( **OTF2\_Archive** \* *archive*,  
**OTF2\_EvtReader** \* *reader* )

Close an opened local event reader.

### Parameters

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

### E.3 otf2/OTF2\_Archive.h File Reference

---

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### E.3.4.7 **OTF2\_ErrorCode** OTF2\_Archive\_CloseEvtWriter ( OTF2\_Archive \* *archive*, OTF2\_EvtWriter \* *writer* )

Close an opened local event writer.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### E.3.4.8 **OTF2\_ErrorCode** OTF2\_Archive\_CloseGlobalDefReader ( OTF2\_Archive \* *archive*, OTF2\_GlobalDefReader \* *globalDefReader* )

Closes the global definition reader.

#### Parameters

<i>archive</i>	Archive handle.
<i>globalDef-Reader</i>	The global definition reader.

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### E.3.4.9 **OTF2\_ErrorCode** OTF2\_Archive\_CloseGlobalDefWriter ( OTF2\_Archive \* *archive*, OTF2\_GlobalDefWriter \* *writer* )

Close an opened global definition writer.

Only the master archive can call this function.

#### Parameters

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

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* if the archive or writer argument is invalid

*OTF2\_ERROR\_INVALID\_CALL* if the archive is not in master mode

**E.3.4.10** **OTF2\_ErrorCode** **OTF2\_Archive\_CloseGlobalEvtReader** ( **OTF2\_Archive** \* *archive*, **OTF2\_GlobalEvtReader** \* *globalEvtReader* )

Closes the global event reader.

This closes also all local event readers.

**Parameters**

<i>archive</i>	Archive handle.
<i>globalEvtReader</i>	The global event reader.

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.3.4.11** **OTF2\_ErrorCode** **OTF2\_Archive\_CloseGlobalSnapReader** ( **OTF2\_Archive** \* *archive*, **OTF2\_GlobalSnapReader** \* *globalSnapReader* )

Close the opened global snapshot reader.

**Parameters**

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

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## E.3 otf2/OTF2\_Archive.h File Reference

---

### E.3.4.12 **OTF2\_ErrorCode** **OTF2\_Archive\_CloseMarkerReader** ( **OTF2\_Archive \*** *archive*, **OTF2\_MarkerReader \*** *markerReader* )

Closes the marker reader.

#### Parameters

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

#### Since

Version 1.2

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### E.3.4.13 **OTF2\_ErrorCode** **OTF2\_Archive\_CloseMarkerWriter** ( **OTF2\_Archive \*** *archive*, **OTF2\_MarkerWriter \*** *writer* )

Close an opened marker writer.

#### Parameters

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

#### Since

Version 1.2

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### E.3.4.14 **OTF2\_ErrorCode** **OTF2\_Archive\_CloseSnapFiles** ( **OTF2\_Archive \*** *archive* )

Closes the snapshots file container.

This function is an collective operation.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.3.4.15** **OTF2\_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

**E.3.4.16** **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.

### Since

Version 1.2



### E.3 otf2/OTF2\_Archive.h File Reference

---

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

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

Close an opened thumbnail reader.

#### Parameters

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

#### Since

Version 1.2

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

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

Get the chunksize.

#### Parameters

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

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

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

Get compression mode (none or zlib)

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

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

Get creator information.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

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

Get a local definition reader.

### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested reader handle.

### Returns

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

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

Get a local definition writer.

### E.3 otf2/OTF2\_Archive.h File Reference

---

#### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested writer handle.

#### Returns

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

#### E.3.4.23 OTF2\_ErrorCode OTF2\_Archive\_GetDescription ( OTF2\_Archive \* *archive*, char \*\* *description* )

Get description.

#### Parameters

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

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.3.4.24 OTF2\_EvtReader\* OTF2\_Archive\_GetEvtReader ( OTF2\_Archive \* *archive*, OTF2\_LocationRef *location* )

Get a local event reader.

#### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested reader handle.

#### Returns

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

#### E.3.4.25 OTF2\_EvtWriter\* OTF2\_Archive\_GetEvtWriter ( OTF2\_Archive \* *archive*, OTF2\_LocationRef *location* )

Get a local event writer.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested writer handle.

### Returns

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

**E.3.4.26** `OTF2_StatusCode OTF2_Archive_GetFileSubstrate ( OTF2_Archive *  
archive, OTF2_FileSubstrate * substrate )`

Get the file substrate (posix, sion, none)

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

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

Get a global definition reader.

Only the master archive can call this function.

### Parameters

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

### Returns

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

**E.3.4.28** `OTF2_GlobalDefWriter* OTF2_Archive_GetGlobalDefWriter ( OTF2_Archive *  
archive )`

Get a global definition writer.

### E.3 otf2/OTF2\_Archive.h File Reference

---

#### Parameters

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

#### Returns

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

#### E.3.4.29 OTF2\_GlobalEvtReader\* OTF2\_Archive\_GetGlobalEvtReader ( OTF2\_Archive \* *archive* )

Get a global event reader.

#### Parameters

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

#### Returns

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

#### E.3.4.30 OTF2\_GlobalSnapReader\* OTF2\_Archive\_GetGlobalSnapReader ( OTF2\_Archive \* *archive* )

Get a global snap reader.

#### Parameters

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

#### Since

Version 1.2

#### Returns

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

#### E.3.4.31 OTF2\_ErrorCode OTF2\_Archive\_GetMachineName ( OTF2\_Archive \* *archive*, char \*\* *machineName* )

Get machine name.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.3.4.32 **OTF2\_MarkerReader\*** **OTF2\_Archive\_GetMarkerReader** ( **OTF2\_Archive** \* ***archive*** )

Get a marker reader.

### Parameters

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

### Since

Version 1.2

### Returns

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

#### E.3.4.33 **OTF2\_MarkerWriter\*** **OTF2\_Archive\_GetMarkerWriter** ( **OTF2\_Archive** \* ***archive*** )

Get a marker writer.

### Parameters

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

### Since

Version 1.2

### Returns

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

### E.3 otf2/OTF2\_Archive.h File Reference

---

**E.3.4.34** **OTF2\_ErrorCode** **OTF2\_Archive\_GetNumberOfGlobalDefinitions** ( **OTF2\_Archive** \* *archive*, **uint64\_t** \* *numberOfDefinitions* )

Get the number of global definitions.

#### Parameters

	<i>archive</i>	Archive handle.
out	<i>numberOfDefinitions</i>	Return pointer to the number of global definitions.

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.35** **OTF2\_ErrorCode** **OTF2\_Archive\_GetNumberOfLocations** ( **OTF2\_Archive** \* *archive*, **uint64\_t** \* *numberOfLocations* )

Get the number of locations.

#### Parameters

	<i>archive</i>	Archive handle.
out	<i>numberOfLocations</i>	Return pointer to the number of locations.

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.36** **OTF2\_ErrorCode** **OTF2\_Archive\_GetNumberOfSnapshots** ( **OTF2\_Archive** \* *archive*, **uint32\_t** \* *number* )

Get the number of snapshots.

#### Parameters

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

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.3.4.37** **OTF2\_ErrorCode** **OTF2\_Archive\_GetNumberOfThumbnails** ( **OTF2\_Archive**  
\* *archive*, **uint32\_t** \* *number* )

Get the number of thumbnails.

**Parameters**

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

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.3.4.38** **OTF2\_ErrorCode** **OTF2\_Archive\_GetProperty** ( **OTF2\_Archive** \* *archive*,  
**const char** \* *name*, **char \*\*** *value* )

Get the value of the named trace file property.

**Parameters**

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

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_PROPERTY\_NOT\_FOUND* if the named property was not  
found



## E.3 otf2/OTF2\_Archive.h File Reference

---

**E.3.4.39** `OTF2_ErrorCode OTF2_Archive_GetPropertyNames ( OTF2_Archive *  
archive, uint32_t * numberOfProperties, char *** names )`

Get the names of all trace file properties.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.40** `OTF2_SnapReader* OTF2_Archive_GetSnapReader ( OTF2_Archive *  
archive, OTF2_LocationRef location )`

Get a local snap reader.

### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested snap handle.

### Since

Version 1.2

### Returns

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

**E.3.4.41** `OTF2_SnapWriter* OTF2_Archive_GetSnapWriter ( OTF2_Archive *  
archive, OTF2_LocationRef location )`

Get a local snap writer.

### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested writer handle.

**Since**

Version 1.2

**Returns**

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

**E.3.4.42** `OTF2_ThumbReader* OTF2_Archive_GetThumbReader ( OTF2_Archive * archive, uint32_t number )`

Get a thumb reader.

**Parameters**

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

**Since**

Version 1.2

**Returns**

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

**E.3.4.43** `OTF2_ThumbWriter* OTF2_Archive_GetThumbWriter ( OTF2_Archive * archive, const char * name, const char * description, OTF2_ThumbnailType type, uint32_t numberOfSamples, uint32_t numberOfMetrics, const uint64_t * refsToDefs )`

Get a thumb writer.

**Parameters**

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

### E.3 otf2/OTF2\_Archive.h File Reference

---

#### Since

Version 1.2

#### Returns

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

**E.3.4.44** `OTF2_ErrorCode OTF2_Archive_GetTraceld ( OTF2_Archive * archive,  
uint64_t * id )`

Get the identifier of the trace file.

#### Note

This call is only allowed when the archive was opened with mode OTF2\_-FILEMODE\_READ.

#### Parameters

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

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.45** `OTF2_ErrorCode OTF2_Archive_GetVersion ( OTF2_Archive * archive,  
uint8_t * major, uint8_t * minor, uint8_t * bugfix )`

Get format version.

#### Parameters

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

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.3.4.46 OTF2\_Archive\* OTF2\_Archive\_Open ( const char \* *archivePath*, const char \* *archiveName*, const OTF2\_FileMode *fileMode*, const uint64\_t *chunkSizeEvents*, const uint64\_t *chunkSizeDefs*, const OTF2\_FileSubstrate *fileSubstrate*, const OTF2\_Compression *compression* )**

Create a new archive.

Creates a new archive handle that keeps all meta data about the archive on runtime.

### Parameters

<i>archivePath</i>	Path to the archive i.e. the directory where the anchor file is located.
<i>archive-Name</i>	Name of the archive. It is used to generate sub pathes e.g. 'archive-Name.otf2'.
<i>fileMode</i>	Determines if in reading or writing mode. Available values are <a href="#">OTF2_FILEMODE_WRITE</a> or <a href="#">OTF2_FILEMODE_READ</a> .
<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 <a href="#">OTF2_CHUNK_SIZE_EVENTS_DEFAULT</a> . In reading mode this value is ignored because the correct chunk size is extracted from the anchor file.
<i>chunk-SizeDefs</i>	Requested size of OTF2's internal definition chunks in writing mode. Available values are from 256kB to 16MB. The definition chunk size affects performance as well as total memory usage. In addition, the definition chunk size must be big enough to carry the largest possible definition record. Therefore, the definition chunk size must be at least 10 times the number of locations. A value satisfying these requirements is about 4MB. If you are not sure which chunk size is the best to use, use <a href="#">OTF2_CHUNK_SIZE_DEFINITIONS_DEFAULT</a> . 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 <a href="#">OTF2_SUBSTRATE_POSIX</a> to use the standard Posix interface, <a href="#">OTF2_SUBSTRATE_SION</a> to use an installed SION library to store multiple logical files into fewer or one physical file, and <a href="#">OTF2_SUBSTRATE_NONE</a> 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 <a href="#">OTF2_COMPRESSION_ZLIB</a> to use an installed zlib and <a href="#">OTF2_COMPRESSION_NONE</a> to disable compression. In reading mode this value is ignored because the correct file compression is extracted from the anchor file.

### E.3 otf2/OTF2\_Archive.h File Reference

---

#### Returns

Returns an archive handle if successful, NULL otherwise.

#### E.3.4.47 OTF2\_ErrorCode OTF2\_Archive\_OpenDefFiles ( OTF2\_Archive \* *archive* )

Open the local definitions file container.

This function is an collective operation.

#### Parameters

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

#### Since

Version 1.3

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.3.4.48 OTF2\_ErrorCode OTF2\_Archive\_OpenEvtFiles ( OTF2\_Archive \* *archive* )

Open the events file container.

This function is an collective operation.

#### Parameters

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

#### Since

Version 1.3

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.3.4.49 OTF2\_ErrorCode OTF2\_Archive\_OpenSnapFiles ( OTF2\_Archive \* *archive* )

Open the snapshots file container.

This function is an collective operation.

## APPENDIX E. FILE DOCUMENTATION

### Parameters

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

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.50** **OTF2\_ErrorCode** **OTF2\_Archive.SelectLocation** ( **OTF2\_Archive** \* *archive*,  
**OTF2\_LocationRef** *location* )

Select a location to be read.

### Parameters

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

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.51** **OTF2\_ErrorCode** **OTF2\_Archive.SetBoolProperty** ( **OTF2\_Archive** \*  
*archive*, **const char** \* *name*, **bool** *value*, **bool** *overwrite* )

Add or remove a boolean trace file property to this archive.

### Note

This call is only allowed when the archive was opened with mode [\*OTF2\\_FILEMODE\\_WRITE\*](#).

### Parameters

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

### E.3 otf2/OTF2\_Archive.h File Reference

---

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_PROPERTY\_NAME\_INVALID** if property name does not conform to the naming scheme

**OTF2\_ERROR\_PROPERTY\_NOT\_FOUND** if property was not found, but requested to remove

**OTF2\_ERROR\_PROPERTY\_EXISTS** if property exists but overwrite was not set

**E.3.4.52** **OTF2\_ErrorCode** **OTF2\_Archive\_SetCollectiveCallbacks** ( **OTF2\_Archive** \* *archive*, **const** **OTF2\_CollectiveCallbacks** \* *collectiveCallbacks*, **void** \* *collectiveData*, **OTF2\_CollectiveContext** \* *globalCommContext*, **OTF2\_CollectiveContext** \* *localCommContext* )

Set the collective callbacks for the archive.

This function is an collective operation.

#### Parameters

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

#### Returns

**OTF2\_SUCCESS** if successful, an error code if an error occurs.

**E.3.4.53** **OTF2\_ErrorCode** **OTF2\_Archive\_SetCreator** ( **OTF2\_Archive** \* *archive*, **const** **char** \* *creator* )

Set creator.

Sets information about the creator of the trace archive. This value is optional. It

---

## APPENDIX E. FILE DOCUMENTATION

---

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

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.3.4.54** **OTF2\_StatusCode** **OTF2\_Archive\_SetDescription** ( **OTF2\_Archive \*** *archive*,  
**const char \*** *description* )

Set a description.

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

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.3.4.55** **OTF2\_StatusCode** **OTF2\_Archive\_SetFlushCallbacks** ( **OTF2\_Archive \***  
*archive*, **const** **OTF2\_FlushCallbacks \*** *flushCallbacks*, **void \*** *flushData* )

Set the flush callbacks for the archive.

### Parameters

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

### Returns

OTF2\_StatusCode, or error code.



## E.3 otf2/OTF2\_Archive.h File Reference

---

**E.3.4.56** **OTF2\_ErrorCode** **OTF2\_Archive\_SetMachineName** ( **OTF2\_Archive** \* *archive*, const char \* *machineName* )

Set machine name.

Sets the name for the machine the trace was recorded. This value is optional. It only needs to be set for an archive handle marked as 'master' or does not need to be set at all.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.57** **OTF2\_ErrorCode** **OTF2\_Archive\_SetMemoryCallbacks** ( **OTF2\_Archive** \* *archive*, const **OTF2\_MemoryCallbacks** \* *memoryCallbacks*, void \* *memoryData* )

Set the memory callbacks for the archive.

### Parameters

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

### Returns

**OTF2\_ErrorCode**, or error code.

**E.3.4.58** **OTF2\_ErrorCode** **OTF2\_Archive\_SetNumberOfSnapshots** ( **OTF2\_Archive** \* *archive*, uint32\_t *number* )

Set the number of snapshots.

### Parameters

---

## APPENDIX E. FILE DOCUMENTATION

---

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

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.3.4.59** `OTF2_ErrorCode OTF2_Archive_SetProperty ( OTF2_Archive * archive,  
const char * name, const char * value, bool overwrite )`

Add or remove a trace file property to this archive.

Removing a trace file property is done by passing "" in the `value` parameter. The `overwrite` parameter is ignored then.

### Note

This call is only allowed when the archive was opened with mode [\*OTF2\\_FILEMODE\\_WRITE\*](#).

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_PROPERTY\\_NAME\\_INVALID\*](#) if property name does not conform to the naming scheme

[\*OTF2\\_ERROR\\_PROPERTY\\_NOT\\_FOUND\*](#) if property was not found, but requested to remove

[\*OTF2\\_ERROR\\_PROPERTY\\_EXISTS\*](#) if property exists but `overwrite` was not set

## E.4 otf2/OTF2\_AttributeList.h File Reference

---

### E.3.4.60 OTF2\_ErrorCode OTF2\_Archive\_SetSerialCollectiveCallbacks ( OTF2\_Archive \* *archive* )

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

#### Parameters

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

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### E.3.4.61 OTF2\_ErrorCode OTF2\_Archive\_SwitchFileMode ( OTF2\_Archive \* *archive*, OTF2\_FileMode *newFileMode* )

Switch file mode of the archive.

Currently only a switch from [\*OTF2\\_FILEMODE\\_READ\*](#) to [\*OTF2\\_FILEMODE\\_WRITE\*](#) is permitted. Currently it is also only permitted when operating on an OTF2 archive with the [\*OTF2\\_SUBSTRATE\\_POSIX\*](#) file substrate.

#### Parameters

<i>archive</i>	Archive handle.
<i>newFile-Mode</i>	New <a href="#"><i>OTF2_FileMode</i></a> to switch to.

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### Since

Version 1.2

## E.4 otf2/OTF2\_AttributeList.h File Reference

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

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

---

## APPENDIX E. FILE DOCUMENTATION

---

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

### Data Structures

- union [OTF2\\_AttributeValue](#)  
*Value container for an attributes.*

### Typedefs

- typedef struct OTF2\_AttributeList\_struct [OTF2\\_AttributeList](#)  
*Attribute list handle.*

### Functions

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddAttribute](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_Type](#) type, [OTF2\\_AttributeValue](#) attribute-Value)  
*Add an attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddAttributeRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_AttributeRef](#) attributeRef)  
*Add an OTF2\_TYPE\_ATTRIBUTE attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddCommRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_CommRef](#) commRef)  
*Add an OTF2\_TYPE\_COMM attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddDouble](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, double float64Value)  
*Add an OTF2\_TYPE\_DOUBLE attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddFloat](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, float float32Value)  
*Add an OTF2\_TYPE\_FLOAT attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddGroupRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_GroupRef](#) groupRef)  
*Add an OTF2\_TYPE\_GROUP attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddInt16](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, int16\_t int16Value)  
*Add an OTF2\_TYPE\_INT16 attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddInt32](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, int32\_t int32Value)  
*Add an OTF2\_TYPE\_INT32 attribute to an attribute list.*

## E.4 otf2/OTF2\_AttributeList.h File Reference

---

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddInt64](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [int64\\_t](#) int64Value)  
*Add an OTF2\_TYPE\_INT64 attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddInt8](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [int8\\_t](#) int8Value)  
*Add an OTF2\_TYPE\_INT8 attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddLocationRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_LocationRef](#) locationRef)  
*Add an OTF2\_TYPE\_LOCATION attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddMetricRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_MetricRef](#) metricRef)  
*Add an OTF2\_TYPE\_METRIC attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddParameterRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_ParameterRef](#) parameterRef)  
*Add an OTF2\_TYPE\_PARAMETER attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddRegionRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_RegionRef](#) regionRef)  
*Add an OTF2\_TYPE\_REGION attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddRmaWinRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_RmaWinRef](#) rmaWinRef)  
*Add an OTF2\_TYPE\_RMA\_WIN attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddString](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_StringRef](#) stringRef)  
*Add an OTF2\_STRING attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddStringRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_StringRef](#) stringRef)  
*Add an OTF2\_TYPE\_STRING attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddUInt16](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [uint16\\_t](#) uint16Value)  
*Add an OTF2\_TYPE\_UINT16 attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddUInt32](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [uint32\\_t](#) uint32Value)  
*Add an OTF2\_TYPE\_UINT32 attribute to an attribute list.*
- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddUInt64](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [uint64\\_t](#) uint64Value)  
*Add an OTF2\_TYPE\_UINT64 attribute to an attribute list.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_AddUInt8](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [uint8\\_t](#) uint8Value)

*Add an OTF2\_TYPE\_UINT8 attribute to an attribute list.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_Delete](#) ([OTF2\\_AttributeList](#) \*attributeList)

*Delete an attribute list handle.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetAttributeByID](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_Type](#) \*type, [OTF2\\_AttributeValue](#) \*attributeValue)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetAttributeByIndex](#) (const [OTF2\\_AttributeList](#) \*attributeList, [uint32\\_t](#) index, [OTF2\\_AttributeRef](#) \*attribute, [OTF2\\_Type](#) \*type, [OTF2\\_AttributeValue](#) \*attributeValue)

*Get an attribute from an attribute list by attribute index.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetAttributeRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_AttributeRef](#) \*attributeRef)

*Get an OTF2\_TYPE\_ATTRIBUTE attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetCommRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_CommRef](#) \*commRef)

*Get an OTF2\_TYPE\_COMM attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetDouble](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [double](#) \*float64Value)

*Get an OTF2\_TYPE\_DOUBLE attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetFloat](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [float](#) \*float32Value)

*Get an OTF2\_TYPE\_FLOAT attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetGroupRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_GroupRef](#) \*groupRef)

*Get an OTF2\_TYPE\_GROUP attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetInt16](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [int16\\_t](#) \*int16Value)

*Get an OTF2\_TYPE\_INT16 attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetInt32](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [int32\\_t](#) \*int32Value)

*Get an OTF2\_TYPE\_INT32 attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetInt64](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [int64\\_t](#) \*int64Value)

## E.4 oftf2/OTF2\_AttributeList.h File Reference

---

*Get an OTF2\_TYPE\_INT64 attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetInt8](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, int8\_t \*int8Value)

*Get an OTF2\_TYPE\_INT8 attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetLocationRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_LocationRef](#) \*locationRef)

*Get an OTF2\_TYPE\_LOCATION attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetMetricRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_MetricRef](#) \*metricRef)

*Get an OTF2\_TYPE\_METRIC attribute from an attribute list by attribute ID.*

- uint32\_t [OTF2\\_AttributeList\\_GetNumberOfElements](#) (const [OTF2\\_AttributeList](#) \*attributeList)

*Get the number of entries in an attribute list.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetParameterRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_ParameterRef](#) \*parameterRef)

*Get an OTF2\_TYPE\_PARAMETER attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetRegionRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_RegionRef](#) \*regionRef)

*Get an OTF2\_TYPE\_REGION attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetRmaWinRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_RmaWinRef](#) \*rmaWinRef)

*Get an OTF2\_TYPE\_RMA\_WIN attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetString](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_StringRef](#) \*stringRef)

*Add an OTF2\_STRING attribute to an attribute list.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetStringRef](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_StringRef](#) \*stringRef)

*Get an OTF2\_TYPE\_STRING attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetUint16](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, uint16\_t \*uint16Value)

*Get an OTF2\_TYPE\_UINT16 attribute from an attribute list by attribute ID.*

- [OTF2\\_ErrorCode](#) [OTF2\\_AttributeList\\_GetUint32](#) (const [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, uint32\_t \*uint32Value)

*Get an OTF2\_TYPE\_UINT32 attribute from an attribute list by attribute ID.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetUint64** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, uint64\_t \*uint64Value)  
*Get an OTF2\_TYPE\_UINT64 attribute from an attribute list by attribute ID.*
- **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetUint8** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, uint8\_t \*uint8Value)  
*Get an OTF2\_TYPE\_UINT8 attribute from an attribute list by attribute ID.*
- **OTF2\_AttributeList** \* **OTF2\_AttributeList\_New** (void)  
*Create a new attribute list handle.*
- **OTF2\_ErrorCode** **OTF2\_AttributeList\_PopAttribute** (**OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** \*attribute, **OTF2\_Type** \*type, **OTF2\_AttributeValue** \*attributeValue)  
*Get first attribute from an attribute list and remove it.*
- **OTF2\_ErrorCode** **OTF2\_AttributeList\_RemoveAllAttributes** (**OTF2\_AttributeList** \*attributeList)  
*Remove all attributes from an attribute list.*
- **OTF2\_ErrorCode** **OTF2\_AttributeList\_RemoveAttribute** (**OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute)  
*Remove an attribute from an attribute list.*
- bool **OTF2\_AttributeList\_TestAttributeByID** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute)  
*Test if an attribute is in the attribute list.*

### E.4.1 Detailed Description

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

#### Source Template:

*template/OTF2\_AttributeList.tmpl.h*

### E.4.2 Function Documentation

- E.4.2.1** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddAttribute** ( **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **OTF2\_Type** *type*, **OTF2\_AttributeValue** *attributeValue* )

Add an attribute to an attribute list.

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



## E.4 oftf2/OTF2\_AttributeList.h File Reference

---

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.2 OTF2\_ErrorCode OTF2\_AttributeList\_AddAttributeRef (**  
**OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*,**  
**OTF2\_AttributeRef *attributeRef* )**

Add an OTF2\_TYPE\_ATTRIBUTE attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.3 OTF2\_ErrorCode OTF2\_AttributeList\_AddCommRef ( OTF2\_AttributeList**  
**\* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_CommRef *commRef* )**

Add an OTF2\_TYPE\_COMM attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.4** `OTF2_StatusCode OTF2_AttributeList_AddDouble ( OTF2_AttributeList *  
attributeList, OTF2_AttributeRef attribute, double float64Value )`

Add an OTF2\_TYPE\_DOUBLE attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.5** `OTF2_StatusCode OTF2_AttributeList_AddFloat ( OTF2_AttributeList *  
attributeList, OTF2_AttributeRef attribute, float float32Value )`

Add an OTF2\_TYPE\_FLOAT attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.6** `OTF2_StatusCode OTF2_AttributeList_AddGroupRef ( OTF2_AttributeList  
* attributeList, OTF2_AttributeRef attribute, OTF2_GroupRef groupRef )`

Add an OTF2\_TYPE\_GROUP attribute to an attribute list.

## E.4 otF2/OTF2\_AttributeList.h File Reference

---

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### E.4.2.7 OTF2\_ErrorCode OTF2\_AttributeList\_AddInt16 ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, int16\_t *int16Value* )

Add an OTF2\_TYPE\_INT16 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### E.4.2.8 OTF2\_ErrorCode OTF2\_AttributeList\_AddInt32 ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, int32\_t *int32Value* )

Add an OTF2\_TYPE\_INT32 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

### E.4.2.9 **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddInt64** ( **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **int64\_t** *int64Value* )

Add an OTF2\_TYPE\_INT64 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### E.4.2.10 **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddInt8** ( **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **int8\_t** *int8Value* )

Add an OTF2\_TYPE\_INT8 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### E.4.2.11 **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddLocationRef** ( **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **OTF2\_LocationRef** *locationRef* )

Add an OTF2\_TYPE\_LOCATION attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

#### Parameters

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

## E.4 otf2/OTF2\_AttributeList.h File Reference

---

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.12** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddMetricRef** ( **OTF2\_AttributeList**  
\* *attributeList*, **OTF2\_AttributeRef** *attribute*, **OTF2\_MetricRef** *metricRef*  
)

Add an OTF2\_TYPE\_METRIC attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.13** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddParameterRef** (  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
**OTF2\_ParameterRef** *parameterRef* )

Add an OTF2\_TYPE\_PARAMETER attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.14** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddRegionRef** (  
    **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
    **OTF2\_RegionRef** *regionRef* )

Add an OTF2\_TYPE\_REGION attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

**Parameters**

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

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.15** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddRmaWinRef** (  
    **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
    **OTF2\_RmaWinRef** *rmaWinRef* )

Add an OTF2\_TYPE\_RMA\_WIN attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

**Parameters**

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

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.16** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddString** ( **OTF2\_AttributeList** \*  
    *attributeList*, **OTF2\_AttributeRef** *attribute*, **OTF2\_StringRef** *stringRef* )

Add an OTF2\_STRING attribute to an attribute list.

**Deprecated**

Use [\*OTF2\\_AttributeList\\_AddStringRef\(\)\*](#) instead.

## E.4 otf2/OTF2\_AttributeList.h File Reference

---

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

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

Add an OTF2\_TYPE\_STRING attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.18** `OTF2_ErrorCode OTF2_AttributeList_AddUint16 ( OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute, uint16_t uint16Value )`

Add an OTF2\_TYPE\_UINT16 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## APPENDIX E. FILE DOCUMENTATION

**E.4.2.19** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddUint32** ( **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **uint32\_t** *uint32Value* )

Add an OTF2\_TYPE\_UINT32 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.20** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddUint64** ( **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **uint64\_t** *uint64Value* )

Add an OTF2\_TYPE\_UINT64 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.21** **OTF2\_ErrorCode** **OTF2\_AttributeList\_AddUint8** ( **OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **uint8\_t** *uint8Value* )

Add an OTF2\_TYPE\_UINT8 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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



## E.4 oftf2/OTF2\_AttributeList.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.22** **OTF2\_ErrorCode** OTF2\_AttributeList\_Delete ( OTF2\_AttributeList \*  
*attributeList* )

Delete an attribute list handle.

Deletes an attribute list handle and releases all associated resources.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.23** **OTF2\_ErrorCode** OTF2\_AttributeList\_GetAttributeByID ( const  
OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*,  
OTF2\_Type \* *type*, OTF2\_AttributeValue \* *attributeValue* )

Get an attribute from an attribute list by attribute ID.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.24** **OTF2\_ErrorCode** OTF2\_AttributeList\_GetAttributeByIndex ( const  
OTF2\_AttributeList \* *attributeList*, uint32\_t *index*, OTF2\_AttributeRef \*  
*attribute*, OTF2\_Type \* *type*, OTF2\_AttributeValue \* *attributeValue* )

Get an attribute from an attribute list by attribute index.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.25** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetAttributeRef** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
**OTF2\_AttributeRef** \* *attributeRef* )

Get an OTF2\_TYPE\_ATTRIBUTE attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.26** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetCommRef** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
**OTF2\_CommRef** \* *commRef* )

Get an OTF2\_TYPE\_COMM attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

## E.4 of2/OTF2\_AttributeList.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.27** `OTF2_ErrorCode OTF2_AttributeList_GetDouble ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute, double  
* float64Value )`

Get an OTF2\_TYPE\_DOUBLE attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.28** `OTF2_ErrorCode OTF2_AttributeList_GetFloat ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute, float *  
float32Value )`

Get an OTF2\_TYPE\_FLOAT attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.4.2.29** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetGroupRef** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
**OTF2\_GroupRef** \* *groupRef* )

Get an OTF2\_TYPE\_GROUP attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.30** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetInt16** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **int16\_t**  
\* *int16Value* )

Get an OTF2\_TYPE\_INT16 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.31** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetInt32** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*, **int32\_t**  
\* *int32Value* )

Get an OTF2\_TYPE\_INT32 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

## E.4 oftf2/OTF2\_AttributeList.h File Reference

---

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.32** `OTF2_ErrorCode OTF2_AttributeList_GetInt64 ( const OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute, int64_t * int64Value )`

Get an OTF2\_TYPE\_INT64 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.33** `OTF2_ErrorCode OTF2_AttributeList_GetInt8 ( const OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute, int8_t * int8Value )`

Get an OTF2\_TYPE\_INT8 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.4.2.34** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetLocationRef** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
**OTF2\_LocationRef** \* *locationRef* )

Get an OTF2\_TYPE\_LOCATION attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.35** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetMetricRef** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
**OTF2\_MetricRef** \* *metricRef* )

Get an OTF2\_TYPE\_METRIC attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.36** **uint32\_t** **OTF2\_AttributeList\_GetNumberOfElements** ( **const**  
**OTF2\_AttributeList** \* *attributeList* )

Get the number of entries in an attribute list.

### Parameters

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

## E.4 of2/OTF2\_AttributeList.h File Reference

---

### Returns

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

**E.4.2.37** `OTF2_ErrorCode OTF2_AttributeList_GetParameterRef ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute,  
OTF2_ParameterRef * parameterRef )`

Get an OTF2\_TYPE\_PARAMETER attribute from an attribute list by attribute ID.  
Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.38** `OTF2_ErrorCode OTF2_AttributeList_GetRegionRef ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute,  
OTF2_RegionRef * regionRef )`

Get an OTF2\_TYPE\_REGION attribute from an attribute list by attribute ID.  
Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.4.2.39** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetRmaWinRef** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
**OTF2\_RmaWinRef** \* *rmaWinRef* )

Get an OTF2\_TYPE\_RMA\_WIN attribute from an attribute list by attribute ID.  
Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.4.2.40** **OTF2\_ErrorCode** **OTF2\_AttributeList\_GetString** ( **const**  
**OTF2\_AttributeList** \* *attributeList*, **OTF2\_AttributeRef** *attribute*,  
**OTF2\_StringRef** \* *stringRef* )

Add an OTF2\_STRING attribute to an attribute list.

### Deprecated

Use [\*OTF2\\_AttributeList\\_GetStringRef\(\)\*](#) instead.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.



## E.4 of2/OTF2\_AttributeList.h File Reference

---

**E.4.2.41** `OTF2_ErrorCode OTF2_AttributeList_GetStringRef ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute,  
OTF2_StringRef * stringRef )`

Get an OTF2\_TYPE\_STRING attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.42** `OTF2_ErrorCode OTF2_AttributeList_GetUint16 ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute,  
uint16_t * uint16Value )`

Get an OTF2\_TYPE\_UINT16 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.43** `OTF2_ErrorCode OTF2_AttributeList_GetUint32 ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute,  
uint32_t * uint32Value )`

Get an OTF2\_TYPE\_UINT32 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.44** `OTF2_ErrorCode OTF2_AttributeList_GetUint64 ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute,  
uint64_t * uint64Value )`

Get an OTF2\_TYPE\_UINT64 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.4.2.45** `OTF2_ErrorCode OTF2_AttributeList_GetUint8 ( const  
OTF2_AttributeList * attributeList, OTF2_AttributeRef attribute, uint8_t  
* uint8Value )`

Get an OTF2\_TYPE\_UINT8 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## E.4 of2/OTF2\_AttributeList.h File Reference

---

### E.4.2.46 OTF2\_AttributeList\* OTF2\_AttributeList.New ( void )

Create a new attribute list handle.

#### Returns

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

### E.4.2.47 OTF2\_ErrorCode OTF2\_AttributeList.PopAttribute ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef \* *attribute*, OTF2\_Type \* *type*, OTF2\_AttributeValue \* *attributeValue* )

Get first attribute from an attribute list and remove it.

Returns the first entry in the attribute list and removes it from the list.

#### Parameters

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

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### E.4.2.48 OTF2\_ErrorCode OTF2\_AttributeList.RemoveAllAttributes ( OTF2\_AttributeList \* *attributeList* )

Remove all attributes from an attribute list.

#### Parameters

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

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

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

Remove an attribute from an attribute list.

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

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

Test if an attribute is in the attribute list.

### Parameters

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

### Returns

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

## E.5 otf2/OTF2\_Callbacks.h File Reference

This header file provides all user callbacks.

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

### Data Structures

- struct [\*OTF2\\_CollectiveCallbacks\*](#)  
*Struct which holds all collective callbacks.*
- struct [\*OTF2\\_FlushCallbacks\*](#)  
*Structure holding the flush callbacks.*

## E.5 otf2/OTF2\_Callbacks.h File Reference

---

- struct [OTF2\\_MemoryCallbacks](#)

*Structure holding the memory callbacks.*

### Typedefs

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_Barrier](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext)  
*Performs an barrier collective on the given communication context.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_Bcast](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext, void \*data, uint32\_t numberElements, [OTF2\\_Type](#) type, uint32\_t root)  
*Performs an broadcast collective on the given communication context.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_CreateLocalComm](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*\*localCommContext, [OTF2\\_CollectiveContext](#) \*globalCommContext, uint32\_t globalRank, uint32\_t globalSize, uint32\_t localRank, uint32\_t localSize, uint32\_t fileNumber, uint32\_t numberOfFiles)  
*Create a new disjoint partitioning of the the globalCommContext communication context. numberOfFiles denotes the number of the partitions. fileNumber denotes in which of the partitions this OTF2\_Archive should belong. localSize is the size of this partition and localRank the rank of this OTF2\_Archive in the partition.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_FreeLocalComm](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*localCommContext)  
*Destroys the communication context previous created by the [OTF2\\_Collectives\\_CreateLocalComm](#) callback.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_Gather](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext, const void \*inData, void \*outData, uint32\_t numberElements, [OTF2\\_Type](#) type, uint32\_t root)  
*Performs an gather collective on the given communication context where each ranks contribute the same number of elements. outData is only valid at rank root.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_Gatherv](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext, const void \*inData, uint32\_t inElements, void \*outData, const uint32\_t \*outElements, [OTF2\\_Type](#) type, uint32\_t root)  
*Performs an gather collective on the given communication context where each ranks contribute different number of elements. outData and outElements are only valid at rank root.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_Collectives\\_GetRank](#))(void \*userData, [OTF2\\_CollectiveContext](#) \*commContext, uint32\_t \*rank)

---

## APPENDIX E. FILE DOCUMENTATION

---

*Returns the rank of this OTF2\_Archive objects in this communication context. A number between 0 and one less of the size of the communication context.*

- typedef OTF2\_CallbackCode(\* OTF2\_Collectives\_GetSize )(void \*userData, OTF2\_CollectiveContext \*commContext, uint32\_t \*size)

*Returns the number of OTF2\_Archive objects operating in this communication context.*

- typedef void(\* OTF2\_Collectives\_Release )(void \*userData, OTF2\_CollectiveContext \*globalCommContext, OTF2\_CollectiveContext \*localCommContext)

*Optionally called in OTF2\_Archive\_Close or OTF2\_Reader\_Close respectively.*

- typedef OTF2\_CallbackCode(\* OTF2\_Collectives\_Scatter )(void \*userData, OTF2\_CollectiveContext \*commContext, const void \*inData, void \*outData, uint32\_t numberElements, OTF2\_Type type, uint32\_t root)

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

- typedef OTF2\_CallbackCode(\* OTF2\_Collectives\_Scatterv )(void \*userData, OTF2\_CollectiveContext \*commContext, const void \*inData, const uint32\_t \*inElements, void \*outData, uint32\_t outElements, OTF2\_Type type, uint32\_t root)

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

- typedef void (\* OTF2\_MemoryAllocate )(void \*userData, OTF2\_FileType fileType, OTF2\_LocationRef location, void \*\*perBufferData, uint64\_t chunkSize)

*Function pointer for allocating memory for chunks.*

- typedef void(\* OTF2\_MemoryFreeAll )(void \*userData, OTF2\_FileType fileType, OTF2\_LocationRef location, void \*\*perBufferData, bool final)

*Function pointer to release all allocated chunks.*

- typedef OTF2\_TimeStamp(\* OTF2\_PostFlushCallback )(void \*userData, OTF2\_FileType fileType, OTF2\_LocationRef location)

*Definition for the post flush callback.*

- typedef OTF2\_FlushType(\* OTF2\_PreFlushCallback )(void \*userData, OTF2\_FileType fileType, OTF2\_LocationRef location, void \*callerData, bool final)

*Definition for the pre flush callback.*

### E.5.1 Detailed Description

This header file provides all user callbacks.

### E.6 otf2/OTF2\_Definitions.h File Reference

Data types used in the definition records.

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

#### Typedefs

- typedef uint8\_t [OTF2\\_CartPeriodicity](#)  
*Wrapper for enum [OTF2\\_CartPeriodicity\\_enum](#).*
- typedef uint32\_t [OTF2\\_GroupFlag](#)  
*Wrapper for enum [OTF2\\_GroupFlag\\_enum](#).*
- typedef uint8\_t [OTF2\\_GroupType](#)  
*Wrapper for enum [OTF2\\_GroupType\\_enum](#).*
- typedef uint8\_t [OTF2\\_LocationGroupType](#)  
*Wrapper for enum [OTF2\\_LocationGroupType\\_enum](#).*
- typedef uint8\_t [OTF2\\_LocationType](#)  
*Wrapper for enum [OTF2\\_LocationType\\_enum](#).*
- typedef uint8\_t [OTF2\\_MetricBase](#)  
*Wrapper for enum [OTF2\\_MetricBase\\_enum](#).*
- typedef uint8\_t [OTF2\\_MetricMode](#)  
*Wrapper for enum [OTF2\\_MetricMode\\_enum](#).*
- typedef uint8\_t [OTF2\\_MetricOccurrence](#)  
*Wrapper for enum [OTF2\\_MetricOccurrence\\_enum](#).*
- typedef uint8\_t [OTF2\\_MetricScope](#)  
*Wrapper for enum [OTF2\\_MetricScope\\_enum](#).*
- typedef uint8\_t [OTF2\\_MetricTiming](#)  
*Wrapper for enum [OTF2\\_MetricTiming\\_enum](#).*
- typedef uint8\_t [OTF2\\_MetricType](#)  
*Wrapper for enum [OTF2\\_MetricType\\_enum](#).*
- typedef uint8\_t [OTF2\\_MetricValueProperty](#)  
*Wrapper for enum [OTF2\\_MetricValueProperty\\_enum](#).*
- typedef uint8\_t [OTF2\\_ParameterType](#)  
*Wrapper for enum [OTF2\\_ParameterType\\_enum](#).*
- typedef uint8\_t [OTF2\\_RecorderKind](#)  
*Wrapper for enum [OTF2\\_RecorderKind\\_enum](#).*
- typedef uint32\_t [OTF2\\_RegionFlag](#)  
*Wrapper for enum [OTF2\\_RegionFlag\\_enum](#).*

- typedef uint8\_t OTF2\_RegionRole  
*Wrapper for enum OTF2\_RegionRole\_enum.*
- typedef uint8\_t OTF2\_SystemTreeDomain  
*Wrapper for enum OTF2\_SystemTreeDomain\_enum.*

## Enumerations

- enum OTF2\_CartPeriodicity\_enum {  
OTF2\_CART\_PERIODIC\_FALSE = 0,  
OTF2\_CART\_PERIODIC\_TRUE = 1 }  
*Periodicity types of a cartesian topology dimension.*
- enum OTF2\_GroupFlag\_enum {  
OTF2\_GROUP\_FLAG\_NONE = 0,  
OTF2\_GROUP\_FLAG\_GLOBAL\_MEMBERS = ( 1 << 0 ) }  
*List of possible flags to specify special characteristics of a Group.*
- enum OTF2\_GroupType\_enum {  
OTF2\_GROUP\_TYPE\_UNKNOWN = 0,  
OTF2\_GROUP\_TYPE\_LOCATIONS = 1,  
OTF2\_GROUP\_TYPE\_REGIONS = 2,  
OTF2\_GROUP\_TYPE\_METRIC = 3,  
OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS = 4,  
OTF2\_GROUP\_TYPE\_COMM\_GROUP = 5,  
OTF2\_GROUP\_TYPE\_COMM\_SELF = 6 }  
*List of available group types.*
- enum OTF2\_LocationGroupType\_enum {  
OTF2\_LOCATION\_GROUP\_TYPE\_UNKNOWN = 0,  
OTF2\_LOCATION\_GROUP\_TYPE\_PROCESS = 1 }  
*List of possible definitions of type LocationGroup.*
- enum OTF2\_LocationType\_enum {  
OTF2\_LOCATION\_TYPE\_UNKNOWN = 0,  
OTF2\_LOCATION\_TYPE\_CPU\_THREAD = 1,  
OTF2\_LOCATION\_TYPE\_GPU = 2,  
OTF2\_LOCATION\_TYPE\_METRIC = 3 }  
*List of possible definitions of type Location.*



## E.6 otf2/OTF2\_Definitions.h File Reference

---

- enum `OTF2_MetricBase_enum` {  
    `OTF2_BASE_BINARY` = 0,  
    `OTF2_BASE_DECIMAL` = 1 }

*Metric base types.*

- enum `OTF2_MetricMode_enum` {  
    `OTF2_METRIC_ACCUMULATED_START` = `OTF2_METRIC_VALUE_-`  
    `ACCUMULATED` | `OTF2_METRIC_TIMING_START`,  
    `OTF2_METRIC_ACCUMULATED_POINT` = `OTF2_METRIC_VALUE_-`  
    `ACCUMULATED` | `OTF2_METRIC_TIMING_POINT`,  
    `OTF2_METRIC_ACCUMULATED_LAST` = `OTF2_METRIC_VALUE_ACCUMULATED`  
    | `OTF2_METRIC_TIMING_LAST`,  
    `OTF2_METRIC_ACCUMULATED_NEXT` = `OTF2_METRIC_VALUE_-`  
    `ACCUMULATED` | `OTF2_METRIC_TIMING_NEXT`,  
    `OTF2_METRIC_ABSOLUTE_POINT` = `OTF2_METRIC_VALUE_ABSOLUTE`  
    | `OTF2_METRIC_TIMING_POINT`,  
    `OTF2_METRIC_ABSOLUTE_LAST` = `OTF2_METRIC_VALUE_ABSOLUTE`  
    | `OTF2_METRIC_TIMING_LAST`,  
    `OTF2_METRIC_ABSOLUTE_NEXT` = `OTF2_METRIC_VALUE_ABSOLUTE`  
    | `OTF2_METRIC_TIMING_NEXT`,  
    `OTF2_METRIC_RELATIVE_POINT` = `OTF2_METRIC_VALUE_RELATIVE`  
    | `OTF2_METRIC_TIMING_POINT`,  
    `OTF2_METRIC_RELATIVE_LAST` = `OTF2_METRIC_VALUE_RELATIVE`  
    | `OTF2_METRIC_TIMING_LAST`,  
    `OTF2_METRIC_RELATIVE_NEXT` = `OTF2_METRIC_VALUE_RELATIVE`  
    | `OTF2_METRIC_TIMING_NEXT` }

*Metric mode is a combination of value property and timing information.*

- enum `OTF2_MetricOccurrence_enum` {  
    `OTF2_METRIC_SYNCHRONOUS_STRICT` = 0,  
    `OTF2_METRIC_SYNCHRONOUS` = 1,  
    `OTF2_METRIC_ASYNCHRONOUS` = 2 }

*Metric occurrence.*

- enum `OTF2_MetricScope_enum` {  
    `OTF2_SCOPE_LOCATION` = 0,  
    `OTF2_SCOPE_LOCATION_GROUP` = 1,  
    `OTF2_SCOPE_SYSTEM_TREE_NODE` = 2,  
    `OTF2_SCOPE_GROUP` = 3 }

*List of available metric scopes.*

- enum OTF2\_MetricTiming\_enum {  
OTF2\_METRIC\_TIMING\_START = 0,  
OTF2\_METRIC\_TIMING\_POINT = 1 << 4,  
OTF2\_METRIC\_TIMING\_LAST = 2 << 4,  
OTF2\_METRIC\_TIMING\_NEXT = 3 << 4,  
OTF2\_METRIC\_TIMING\_MASK = 240 }  
*Determines when the values have been collected or for which interval of time they are valid. Used for the upper half-byte of OTF2\_MetricMode.*
- enum OTF2\_MetricType\_enum {  
OTF2\_METRIC\_TYPE\_OTHER = 0,  
OTF2\_METRIC\_TYPE\_PAPI = 1,  
OTF2\_METRIC\_TYPE\_RUSAGE = 2,  
OTF2\_METRIC\_TYPE\_USER = 3 }  
*List of available metric types.*
- enum OTF2\_MetricValueProperty\_enum {  
OTF2\_METRIC\_VALUE\_ACCUMULATED = 0,  
OTF2\_METRIC\_VALUE\_ABSOLUTE = 1,  
OTF2\_METRIC\_VALUE\_RELATIVE = 2,  
OTF2\_METRIC\_VALUE\_MASK = 15 }  
*Information about whether the metric value is accumulated, absolute, or relative. Used for the lower half-byte of OTF2\_MetricMode.*
- enum OTF2\_ParameterType\_enum {  
OTF2\_PARAMETER\_TYPE\_STRING = 0,  
OTF2\_PARAMETER\_TYPE\_INT64 = 1,  
OTF2\_PARAMETER\_TYPE\_UINT64 = 2 }  
*List of possible for definitions of type Parameter.*
- enum OTF2\_RecorderKind\_enum {  
OTF2\_RECORDER\_KIND\_UNKNOWN = 0,  
OTF2\_RECORDER\_KIND\_ABSTRACT = 1,  
OTF2\_RECORDER\_KIND\_CPU = 2,  
OTF2\_RECORDER\_KIND\_GPU = 3 }  
*List of possible kinds a MetricClass can be recorded by.*
- enum OTF2\_RegionFlag\_enum {  
OTF2\_REGION\_FLAG\_NONE = 0,  
OTF2\_REGION\_FLAG\_DYNAMIC = ( 1 << 0 ),  
OTF2\_REGION\_FLAG\_PHASE = ( 1 << 1 ) }

## E.6 of2/OTF2\_Definitions.h File Reference

---

*List of possible flags to specify special characteristics of a Region.*

- enum OTF2\_RegionRole\_enum {  
OTF2\_REGION\_ROLE\_UNKNOWN = 0,  
OTF2\_REGION\_ROLE\_FUNCTION = 1,  
OTF2\_REGION\_ROLE\_WRAPPER = 2,  
OTF2\_REGION\_ROLE\_LOOP = 3,  
OTF2\_REGION\_ROLE\_CODE = 4,  
OTF2\_REGION\_ROLE\_PARALLEL = 5,  
OTF2\_REGION\_ROLE\_SECTIONS = 6,  
OTF2\_REGION\_ROLE\_SECTION = 7,  
OTF2\_REGION\_ROLE\_WORKSHARE = 8,  
OTF2\_REGION\_ROLE\_SINGLE = 9,  
OTF2\_REGION\_ROLE\_SINGLE\_SBLOCK = 10,  
OTF2\_REGION\_ROLE\_MASTER = 11,  
OTF2\_REGION\_ROLE\_CRITICAL = 12,  
OTF2\_REGION\_ROLE\_CRITICAL\_SBLOCK = 13,  
OTF2\_REGION\_ROLE\_ATOMIC = 14,  
OTF2\_REGION\_ROLE\_BARRIER = 15,  
OTF2\_REGION\_ROLE\_IMPLICIT\_BARRIER = 16,  
OTF2\_REGION\_ROLE\_FLUSH = 17,  
OTF2\_REGION\_ROLE\_ORDERED = 18,  
OTF2\_REGION\_ROLE\_ORDERED\_SBLOCK = 19,  
OTF2\_REGION\_ROLE\_TASK = 20,  
OTF2\_REGION\_ROLE\_TASK\_CREATE = 21,  
OTF2\_REGION\_ROLE\_TASK\_WAIT = 22,  
OTF2\_REGION\_ROLE\_COLL\_ONE2ALL = 23,  
OTF2\_REGION\_ROLE\_COLL\_ALL2ONE = 24,  
OTF2\_REGION\_ROLE\_COLL\_ALL2ALL = 25,  
OTF2\_REGION\_ROLE\_COLL\_OTHER = 26,  
OTF2\_REGION\_ROLE\_FILE\_IO = 27,  
OTF2\_REGION\_ROLE\_POINT2POINT = 28,  
OTF2\_REGION\_ROLE\_RMA = 29,  
OTF2\_REGION\_ROLE\_DATA\_TRANSFER = 30,

```
OTF2_REGION_ROLE_ARTIFICIAL = 31,  
OTF2_REGION_ROLE_THREAD_CREATE = 32,  
OTF2_REGION_ROLE_THREAD_WAIT = 33 }
```

*List of possible roles of a Region.*

- `enum OTF2_SystemTreeDomain_enum` {  
    `OTF2_SYSTEM_TREE_DOMAIN_MACHINE` = 0,  
    `OTF2_SYSTEM_TREE_DOMAIN_SHARED_MEMORY` = 1,  
    `OTF2_SYSTEM_TREE_DOMAIN_NUMA` = 2,  
    `OTF2_SYSTEM_TREE_DOMAIN_SOCKET` = 3,  
    `OTF2_SYSTEM_TREE_DOMAIN_CACHE` = 4,  
    `OTF2_SYSTEM_TREE_DOMAIN_CORE` = 5,  
    `OTF2_SYSTEM_TREE_DOMAIN_PU` = 6 }

*List of available system tree node domains.*

### **E.6.1 Detailed Description**

Data types used in the definition records.

#### **Source Template:**

*templates/OTF2\_Definitions.tmpl.h*

### **E.6.2 Enumeration Type Documentation**

#### **E.6.2.1 `enum OTF2_CartPeriodicity_enum`**

Periodicity types of a cartesian topology dimension.

#### **Since**

Version 1.0

#### **Enumerator:**

***OTF2\_CART\_PERIODIC\_FALSE*** Dimension is not periodic.

***OTF2\_CART\_PERIODIC\_TRUE*** Dimension is periodic.

## E.6 otf2/OTF2\_Definitions.h File Reference

---

### E.6.2.2 enum OTF2\_GroupFlag\_enum

List of possible flags to specify special characteristics of a Group.

Since

Version 1.2

Enumerator:

**OTF2\_GROUP\_FLAG\_NONE** A group without special characterization.

**OTF2\_GROUP\_FLAG\_GLOBAL\_MEMBERS** No translation of ranks in event records needs to be done when a group of type [OTF2\\_GROUP\\_TYPE\\_COMM\\_GROUP](#) has this flag. I.e., the ranks are indexes into the [OTF2\\_GROUP\\_TYPE\\_COMM\\_LOCATIONS](#) group.

### E.6.2.3 enum OTF2\_GroupType\_enum

List of available group types.

Since

Version 1.2

Enumerator:

**OTF2\_GROUP\_TYPE\_UNKNOWN** Group of unknown type.

**OTF2\_GROUP\_TYPE\_LOCATIONS** Group of locations.

**OTF2\_GROUP\_TYPE\_REGIONS** Group of regions.

**OTF2\_GROUP\_TYPE\_METRIC** Group of metrics.

**OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS** List of locations which participated in the paradigm specified by the group definition. For example: In case of MPI, the size of this group should match the size of `MPI_COMM_WORLD`. Each entry in the list is a [Location](#) reference, where the index of the entry is equal to the rank in `MPI_COMM_WORLD` (i.e., rank *i* corresponds to location `members[i]`).

Also, if this definition is present, the location group ids of locations with type [OTF2\\_LOCATION\\_TYPE\\_CPU\\_THREAD](#) should match the MPI rank.

This group needs to be defined, before any group of type [OTF2\\_GROUP\\_TYPE\\_COMM\\_GROUP](#) and the same paradigm.

Note: This does not makes sense in local definitions.

***OTF2\_GROUP\_TYPE\_COMM\_GROUP*** A sub-group of the corresponding group definition with type *OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS* and the same paradigm. The sub-group is formed by listing the indexes of the *OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS* group.

***OTF2\_GROUP\_TYPE\_COMM\_SELF*** Special group type to efficiently handle self-like communicators (i.e., MPI\_COMM\_SELF and friends). At most one of this definition is allowed to exist per paradigm.

#### **E.6.2.4 enum OTF2\_LocationGroupType\_enum**

List of possible definitions of type LocationGroup.

##### **Since**

Version 1.0

##### **Enumerator:**

***OTF2\_LOCATION\_GROUP\_TYPE\_UNKNOWN*** A location group of unknown type.

***OTF2\_LOCATION\_GROUP\_TYPE\_PROCESS*** A process.

#### **E.6.2.5 enum OTF2\_LocationType\_enum**

List of possible definitions of type Location.

##### **Since**

Version 1.0

##### **Enumerator:**

***OTF2\_LOCATION\_TYPE\_UNKNOWN*** A location of unknown type.

***OTF2\_LOCATION\_TYPE\_CPU\_THREAD*** A CPU thread.

***OTF2\_LOCATION\_TYPE\_GPU*** A GPU location.

***OTF2\_LOCATION\_TYPE\_METRIC*** A metric only location e.g. an external device.

## E.6 otf2/OTF2\_Definitions.h File Reference

---

### E.6.2.6 enum OTF2\_MetricBase\_enum

Metric base types.

#### Since

Version 1.0

#### Enumerator:

***OTF2\_BASE\_BINARY*** Binary base.

***OTF2\_BASE\_DECIMAL*** Decimal base.

### E.6.2.7 enum OTF2\_MetricMode\_enum

Metric mode is a combination of value property and timing information.

#### Since

Version 1.0

#### Enumerator:

***OTF2\_METRIC\_ACCUMULATED\_START*** Accumulated metric, 'START' timing.

***OTF2\_METRIC\_ACCUMULATED\_POINT*** Accumulated metric, 'POINT' timing.

***OTF2\_METRIC\_ACCUMULATED\_LAST*** Accumulated metric, 'LAST' timing.

***OTF2\_METRIC\_ACCUMULATED\_NEXT*** Accumulated metric, 'NEXT' timing.

***OTF2\_METRIC\_ABSOLUTE\_POINT*** Absolute metric, 'POINT' timing.

***OTF2\_METRIC\_ABSOLUTE\_LAST*** Absolute metric, 'LAST' timing.

***OTF2\_METRIC\_ABSOLUTE\_NEXT*** Absolute metric, 'NEXT' timing.

***OTF2\_METRIC\_RELATIVE\_POINT*** Relative metric, 'POINT' timing.

***OTF2\_METRIC\_RELATIVE\_LAST*** Relative metric, 'LAST' timing.

***OTF2\_METRIC\_RELATIVE\_NEXT*** Relative metric, 'NEXT' timing.

#### E.6.2.8 enum OTF2\_MetricOccurrence\_enum

Metric occurrence.

##### Since

Version 1.0

##### Enumerator:

***OTF2\_METRIC\_SYNCHRONOUS\_STRICT*** Metric occurs at every region enter and leave.

***OTF2\_METRIC\_SYNCHRONOUS*** Metric occurs only at a region enter and leave, but does not need to occur at every enter/leave.

***OTF2\_METRIC\_ASYNCHRONOUS*** Metric can occur at any place i.e. it is not related to region enter and leaves.

#### E.6.2.9 enum OTF2\_MetricScope\_enum

List of available metric scopes.

##### Since

Version 1.0

##### Enumerator:

***OTF2\_SCOPE\_LOCATION*** Scope of a metric is another location.

***OTF2\_SCOPE\_LOCATION\_GROUP*** Scope of a metric is a location group.

***OTF2\_SCOPE\_SYSTEM\_TREE\_NODE*** Scope of a metric is a system tree node.

***OTF2\_SCOPE\_GROUP*** Scope of a metric is a generic group of locations.

#### E.6.2.10 enum OTF2\_MetricTiming\_enum

Determines when the values have been collected or for which interval of time they are valid. Used for the upper half-byte of OTF2\_MetricMode.

##### Since

Version 1.0



## E.6 oftf2/OTF2\_Definitions.h File Reference

---

### Enumerator:

- OTF2\_METRIC\_TIMING\_START*** Metric value belongs to the time interval since the beginning of the measurement.
- OTF2\_METRIC\_TIMING\_POINT*** Metric value is only valid at a point in time but not necessarily for any interval of time.
- OTF2\_METRIC\_TIMING\_LAST*** Metric value is related to the time interval since the last counter sample of the same metric, i.e. the immediate past.
- OTF2\_METRIC\_TIMING\_NEXT*** Metric value is valid from now until the next counter sample, i.e. the future right ahead.
- OTF2\_METRIC\_TIMING\_MASK*** This mask can be used to get the upper half-byte in OTF2\_MetricMode that is used to indicate metric timing information.

### E.6.2.11 enum OTF2\_MetricType\_enum

List of available metric types.

#### Since

Version 1.0

### Enumerator:

- OTF2\_METRIC\_TYPE\_OTHER*** Any metric of a type not explicitly listed below.
- OTF2\_METRIC\_TYPE\_PAPI*** PAPI counter.
- OTF2\_METRIC\_TYPE\_RUSAGE*** Resource usage counter.
- OTF2\_METRIC\_TYPE\_USER*** User metrics.

### E.6.2.12 enum OTF2\_MetricValueProperty\_enum

Information about whether the metric value is accumulated, absolute, or relative. Used for the lower half-byte of OTF2\_MetricMode.

#### Since

Version 1.0

**Enumerator:**

***OTF2\_METRIC\_VALUE\_ACCUMULATED*** Accumulated metric is monotonously increasing (i.e., PAPI counter for number of executed floating point operations).

***OTF2\_METRIC\_VALUE\_ABSOLUTE*** Absolute metric (i.e., temperature, rate, mean value, etc.).

***OTF2\_METRIC\_VALUE\_RELATIVE*** Relative metric.

***OTF2\_METRIC\_VALUE\_MASK*** This mask can be used to get lower half-byte in *OTF2\_MetricMode* that is used to indicate metric value property.

**E.6.2.13 enum OTF2\_ParameterType\_enum**

List of possible for definitions of type *Parameter*.

**Since**

Version 1.0

**Enumerator:**

***OTF2\_PARAMETER\_TYPE\_STRING*** Parameter is of type string.

***OTF2\_PARAMETER\_TYPE\_INT64*** Parameter is of type signed 8-byte integer.

***OTF2\_PARAMETER\_TYPE\_UINT64*** Parameter is of type unsigned 8-byte integer.

**E.6.2.14 enum OTF2\_RecorderKind\_enum**

List of possible kinds a *MetricClass* can be recorded by.

**Since**

Version 1.2

**Enumerator:**

***OTF2\_RECORDER\_KIND\_UNKNOWN*** No specific kind of recorder.

***OTF2\_RECORDER\_KIND\_ABSTRACT*** The metric class will only be recorded via a *MetricInstance* definitions.

***OTF2\_RECORDER\_KIND\_CPU*** This metric class will only be recored by locations of type *OTF2\_LOCATION\_TYPE\_CPU\_THREAD*.

***OTF2\_RECORDER\_KIND\_GPU*** This metric class will only be recored by locations of type *OTF2\_LOCATION\_TYPE\_GPU*.

## E.6 otf2/OTF2\_Definitions.h File Reference

---

### E.6.2.15 enum OTF2\_RegionFlag\_enum

List of possible flags to specify special characteristics of a Region.

**Since**

Version 1.1

**Enumerator:**

***OTF2\_REGION\_FLAG\_NONE*** A region without special characterization.

***OTF2\_REGION\_FLAG\_DYNAMIC*** Each time this region is entered it will get an individual call path in the profile.

***OTF2\_REGION\_FLAG\_PHASE*** Each time this region is entered it will get an individual root node in the profile.

### E.6.2.16 enum OTF2\_RegionRole\_enum

List of possible roles of a Region.

**Since**

Version 1.1

**Enumerator:**

***OTF2\_REGION\_ROLE\_UNKNOWN*** A region of unknown role.

***OTF2\_REGION\_ROLE\_FUNCTION*** An entire function/subroutine.

***OTF2\_REGION\_ROLE\_WRAPPER*** An API function wrapped by Score-P.

***OTF2\_REGION\_ROLE\_LOOP*** A loop in the code.

***OTF2\_REGION\_ROLE\_CODE*** An arbitrary section of code.

***OTF2\_REGION\_ROLE\_PARALLEL*** E.g. OpenMP "parallel" construct (structured block)

***OTF2\_REGION\_ROLE\_SECTIONS*** E.g. OpenMP "sections" construct.

***OTF2\_REGION\_ROLE\_SECTION*** Individual "section" inside an OpenMP "sections" construct.

***OTF2\_REGION\_ROLE\_WORKSHARE*** E.g. OpenMP "workshare" construct.

***OTF2\_REGION\_ROLE\_SINGLE*** E.g. OpenMP "single" construct.

---

## APPENDIX E. FILE DOCUMENTATION

---

**OTF2\_REGION\_ROLE\_SINGLE\_SBLOCK** E.g. OpenMP "single" construct (structured block)

**OTF2\_REGION\_ROLE\_MASTER** E.g. OpenMP "master" construct.

**OTF2\_REGION\_ROLE\_CRITICAL** E.g. OpenMP "critical" construct.

**OTF2\_REGION\_ROLE\_CRITICAL\_SBLOCK** E.g. OpenMP "critical" construct (structured block)

**OTF2\_REGION\_ROLE\_ATOMIC** E.g. OpenMP "atomic" construct.

**OTF2\_REGION\_ROLE\_BARRIER** Explicit barrier.

**OTF2\_REGION\_ROLE\_IMPLICIT\_BARRIER** Implicit barrier.

**OTF2\_REGION\_ROLE\_FLUSH** E.g. OpenMP "flush" construct.

**OTF2\_REGION\_ROLE\_ORDERED** E.g. OpenMP "ordered" construct.

**OTF2\_REGION\_ROLE\_ORDERED\_SBLOCK** E.g. OpenMP "ordered" construct (structured block)

**OTF2\_REGION\_ROLE\_TASK** "task" construct (structured block)

**OTF2\_REGION\_ROLE\_TASK\_CREATE** "task" construct (creation)

**OTF2\_REGION\_ROLE\_TASK\_WAIT** "taskwait" construct

**OTF2\_REGION\_ROLE\_COLL\_ONE2ALL** Collective 1:N communication operation.

**OTF2\_REGION\_ROLE\_COLL\_ALL2ONE** Collective N:1 communication operation.

**OTF2\_REGION\_ROLE\_COLL\_ALL2ALL** Collective N:N communication operation.

**OTF2\_REGION\_ROLE\_COLL\_OTHER** Collective M:N communication operation.

**OTF2\_REGION\_ROLE\_FILE\_IO** Any file I/O operation.

**OTF2\_REGION\_ROLE\_POINT2POINT** A point-to-point communication function.

**OTF2\_REGION\_ROLE\_RMA** A remote memory access communication operation.

**OTF2\_REGION\_ROLE\_DATA\_TRANSFER** A data transfer operation in memory.

**OTF2\_REGION\_ROLE\_ARTIFICIAL** An artificial region, mostly used by the monitor software.

**Since**

Version 1.2.

**OTF2\_REGION\_ROLE\_THREAD\_CREATE** A function which creates one thread.

## E.7 otf2/OTF2\_DefReader.h File Reference

---

### Since

Version 1.3.

***OTF2\_REGION\_ROLE\_THREAD\_WAIT*** A function which waits for the completion of one thread.

### Since

Version 1.3.

### E.6.2.17 enum OTF2\_SystemTreeDomain\_enum

List of available system tree node domains.

### Since

Version 1.2

### Enumerator:

***OTF2\_SYSTEM\_TREE\_DOMAIN\_MACHINE*** All nodes below a node with this attribute encompass a tightly coupled HPC system.

***OTF2\_SYSTEM\_TREE\_DOMAIN\_SHARED\_MEMORY*** All nodes below a node with this attribute encompass a system where processes can communicate via hardware shared memory.

***OTF2\_SYSTEM\_TREE\_DOMAIN\_NUMA*** A numa domain. A set of processors around memory which the processors can directly access.

***OTF2\_SYSTEM\_TREE\_DOMAIN\_SOCKET*** Socket, physical package, or chip. In the physical meaning, i.e. that you can add or remove physically.

***OTF2\_SYSTEM\_TREE\_DOMAIN\_CACHE*** Cache. Can be L1i, L1d, L2, L3, ...

***OTF2\_SYSTEM\_TREE\_DOMAIN\_CORE*** Core. A computation unit (may be shared by several logical processors).

***OTF2\_SYSTEM\_TREE\_DOMAIN\_PU*** Processing Unit (An non-shared ALU, FPU, ...)

## E.7 otf2/OTF2\_DefReader.h File Reference

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

---

## APPENDIX E. FILE DOCUMENTATION

---

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

### Functions

- [OTF2\\_ErrorCode OTF2\\_DefReader\\_GetLocationID](#) (const [OTF2\\_DefReader](#) \*reader, [OTF2\\_LocationRef](#) \*location)  
*Get the location ID of this reader object.*
- [OTF2\\_ErrorCode OTF2\\_DefReader\\_ReadDefinitions](#) ([OTF2\\_DefReader](#) \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)  
*Reads the given number of records from the definition reader.*
- [OTF2\\_ErrorCode OTF2\\_DefReader\\_SetCallbacks](#) ([OTF2\\_DefReader](#) \*reader, const [OTF2\\_DefReaderCallbacks](#) \*callbacks, void \*userData)  
*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### E.7.1 Detailed Description

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

### E.7.2 Function Documentation

#### E.7.2.1 [OTF2\\_ErrorCode OTF2\\_DefReader\\_GetLocationID](#) ( const [OTF2\\_DefReader](#) \* reader, [OTF2\\_LocationRef](#) \* location )

Get the location ID of this reader object.

#### Parameters

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

#### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.7 otf2/OTF2\_DefReader.h File Reference

---

### E.7.2.2 OTF2\_ErrorCode OTF2.DefReader\_ReadDefinitions ( OTF2\_DefReader \* *reader*, uint64\_t *recordsToRead*, uint64\_t \* *recordsRead* )

Reads the given number of records from the definition reader.

#### Parameters

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

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK** if an user supplied callback returned OTF2\_CALLBACK\_INTERRUPT

**OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE** if an duplicate mapping table definition was read

**otherwise** the error code

### E.7.2.3 OTF2\_ErrorCode OTF2.DefReader\_SetCallbacks ( OTF2\_DefReader \* *reader*, const OTF2\_DefReaderCallbacks \* *callbacks*, void \* *userData* )

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

#### Parameters

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

## Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.8 otF2/OTF2\_DefReaderCallbacks.h File Reference

This defines the callbacks for the definition reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_IdMap.h>
```

## Typedefs

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Attribute](#) )(void \*userData, [OTF2\\_AttributeRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) description, [OTF2\\_Type](#) type)  
*Function pointer definition for the callback which is triggered by a [Attribute](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Callpath](#) )(void \*userData, [OTF2\\_CallpathRef](#) self, [OTF2\\_CallpathRef](#) parent, [OTF2\\_RegionRef](#) region)  
*Function pointer definition for the callback which is triggered by a [Callpath](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Callsite](#) )(void \*userData, [OTF2\\_CallsiteRef](#) self, [OTF2\\_StringRef](#) sourceFile, uint32\_t lineNumber, [OTF2\\_RegionRef](#) enteredRegion, [OTF2\\_RegionRef](#) leftRegion)  
*Function pointer definition for the callback which is triggered by a [Callsite](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_CartCoordinate](#) )(void \*userData, [OTF2\\_CartTopologyRef](#) cartTopology, uint32\_t rank, uint8\_t numberOfDimensions, const uint32\_t \*coordinates)  
*Function pointer definition for the callback which is triggered by a [CartCoordinate](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_CartDimension](#) )(void \*userData, [OTF2\\_CartDimensionRef](#) self, [OTF2\\_StringRef](#) name, uint32\_t size, [OTF2\\_CartPeriodicity](#) cartPeriodicity)  
*Function pointer definition for the callback which is triggered by a [CartDimension](#) definition record.*



## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_CartTopology](#))(void \*userData, [OTF2\\_CartTopologyRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_CommRef](#) communicator, uint8\_t numberOfDimensions, const [OTF2\\_CartDimensionRef](#) \*cartDimensions)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_ClockOffset](#))(void \*userData, [OTF2\\_TimeStamp](#) time, int64\_t offset, double standardDeviation)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Comm](#))(void \*userData, [OTF2\\_CommRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_GroupRef](#) group, [OTF2\\_CommRef](#) parent)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Group](#))(void \*userData, [OTF2\\_GroupRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_GroupType](#) groupType, [OTF2\\_Paradigm](#) paradigm, [OTF2\\_GroupFlag](#) groupFlags, uint32\_t numberOfMembers, const uint64\_t \*members)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Location](#))(void \*userData, [OTF2\\_LocationRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_LocationType](#) locationType, uint64\_t numberOfEvents, [OTF2\\_LocationGroupRef](#) locationGroup)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_LocationGroup](#))(void \*userData, [OTF2\\_LocationGroupRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_LocationGroupType](#) locationGroupType, [OTF2\\_SystemTreeNodeRef](#) systemTreeParent)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_LocationGroupProperty](#))(void \*userData, [OTF2\\_LocationGroupRef](#) locationGroup, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_LocationProperty](#))(void \*userData, [OTF2\\_LocationRef](#) location, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

---

## APPENDIX E. FILE DOCUMENTATION

---

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_MappingTable](#))(void \*userData, [OTF2\\_MappingType](#) mappingType, const [OTF2\\_IdMap](#) \*idMap)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_MetricClass](#))(void \*userData, [OTF2\\_MetricRef](#) self, uint8\_t numberOfMetrics, const [OTF2\\_MetricMemberRef](#) \*metricMembers, [OTF2\\_MetricOccurrence](#) metricOccurrence, [OTF2\\_RecorderKind](#) recorderKind)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_MetricClassRecorder](#))(void \*userData, [OTF2\\_MetricRef](#) metricClass, [OTF2\\_LocationRef](#) recorder)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_MetricInstance](#))(void \*userData, [OTF2\\_MetricRef](#) self, [OTF2\\_MetricRef](#) metricClass, [OTF2\\_LocationRef](#) recorder, [OTF2\\_MetricScope](#) metricScope, uint64\_t scope)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_MetricMember](#))(void \*userData, [OTF2\\_MetricMemberRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) description, [OTF2\\_MetricType](#) metricType, [OTF2\\_MetricMode](#) metricMode, [OTF2\\_Type](#) valueType, [OTF2\\_MetricBase](#) metricBase, int64\_t exponent, [OTF2\\_StringRef](#) unit)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Parameter](#))(void \*userData, [OTF2\\_ParameterRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_ParameterType](#) parameterType)

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

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Region](#))(void \*userData, [OTF2\\_RegionRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) canonicalName, [OTF2\\_StringRef](#) description, [OTF2\\_RegionRole](#) regionRole, [OTF2\\_Paradigm](#) paradigm, [OTF2\\_RegionFlag](#) regionFlags, [OTF2\\_StringRef](#) sourceFile, uint32\_t beginLineNumber, uint32\_t endLineNumber)

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

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_RmaWin](#) )(void \*userData, [OTF2\\_RmaWinRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_CommRef](#) comm)  
*Function pointer definition for the callback which is triggered by a [RmaWin](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_String](#) )(void \*userData, [OTF2\\_StringRef](#) self, const char \*string)  
*Function pointer definition for the callback which is triggered by a [String](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_SystemTreeNode](#) )(void \*userData, [OTF2\\_SystemTreeNodeRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) className, [OTF2\\_SystemTreeNodeRef](#) parent)  
*Function pointer definition for the callback which is triggered by a [SystemTreeNode](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_SystemTreeNodeDomain](#) )(void \*userData, [OTF2\\_SystemTreeNodeRef](#) systemTreeNode, [OTF2\\_SystemTreeDomain](#) systemTreeDomain)  
*Function pointer definition for the callback which is triggered by a [SystemTreeNodeDomain](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_SystemTreeNodeProperty](#) )(void \*userData, [OTF2\\_SystemTreeNodeRef](#) systemTreeNode, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)  
*Function pointer definition for the callback which is triggered by a [SystemTreeNodeProperty](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_DefReaderCallback\\_Unknown](#) )(void \*userData)  
*Function pointer definition for the callback which is triggered for an unknown definition.*
- typedef struct [OTF2\\_DefReaderCallbacks\\_struct](#) [OTF2\\_DefReaderCallbacks](#)  
*Opaque struct which holds all definition record callbacks.*

## Functions

- void [OTF2\\_DefReaderCallbacks\\_Clear](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks)  
*Clears a struct for the definition callbacks.*
- void [OTF2\\_DefReaderCallbacks\\_Delete](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks)  
*Deallocates a struct for the definition callbacks.*
- [OTF2\\_DefReaderCallbacks](#) \* [OTF2\\_DefReaderCallbacks\\_New](#) (void)

*Allocates a new struct for the definition callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetAttributeCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Attribute](#) attributeCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetCallpathCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Callpath](#) callpathCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetCallsiteCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Callsite](#) callsiteCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetCartCoordinateCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_CartCoordinate](#) cartCoordinateCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetCartDimensionCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_CartDimension](#) cartDimensionCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetCartTopologyCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_CartTopology](#) cartTopologyCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetClockOffsetCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_ClockOffset](#) clockOffsetCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetCommCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Comm](#) commCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetGroupCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Group](#) groupCallback)

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

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetLocationCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Location](#) locationCallback)

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

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetLocationGroupCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_LocationGroup](#) locationGroupCallback)  
*Registers the callback for the [LocationGroup](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetLocationGroupPropertyCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_LocationGroupProperty](#) locationGroupPropertyCallback)  
*Registers the callback for the [LocationGroupProperty](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetLocationPropertyCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_LocationProperty](#) locationPropertyCallback)  
*Registers the callback for the [LocationProperty](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetMappingTableCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_MappingTable](#) mappingTableCallback)  
*Registers the callback for the [MappingTable](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetMetricClassCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_MetricClass](#) metricClassCallback)  
*Registers the callback for the [MetricClass](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetMetricClassRecorderCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_MetricClassRecorder](#) metricClassRecorderCallback)  
*Registers the callback for the [MetricClassRecorder](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetMetricInstanceCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_MetricInstance](#) metricInstanceCallback)  
*Registers the callback for the [MetricInstance](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetMetricMemberCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_MetricMember](#) metricMemberCallback)  
*Registers the callback for the [MetricMember](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetParameterCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Parameter](#) parameterCallback)  
*Registers the callback for the [Parameter](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetRegionCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Region](#) regionCallback)  
*Registers the callback for the [Region](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetRmaWinCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_RmaWin](#) rmaWinCallback)  
*Registers the callback for the [RmaWin](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetStringCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_String](#) stringCallback)  
*Registers the callback for the [String](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetSystemTreeNodeCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_SystemTreeNode](#) systemTreeNodeCallback)  
*Registers the callback for the [SystemTreeNode](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetSystemTreeNodeDomainCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_SystemTreeNodeDomain](#) systemTreeNodeDomainCallback)  
*Registers the callback for the [SystemTreeNodeDomain](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetSystemTreeNodePropertyCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_SystemTreeNodeProperty](#) systemTreeNodePropertyCallback)  
*Registers the callback for the [SystemTreeNodeProperty](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_DefReaderCallbacks\\_SetUnknownCallback](#) ([OTF2\\_DefReaderCallbacks](#) \*defReaderCallbacks, [OTF2\\_DefReaderCallback\\_Unknown](#) unknownCallback)  
*Registers the callback for an unknown definition.*

### **E.8.1 Detailed Description**

This defines the callbacks for the definition reader.

#### **Source Template:**

*templates/OTF2\_DefReaderCallbacks.tmpl.h*

### **E.8.2 Typedef Documentation**

**E.8.2.1** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_Attribute)(void *userData, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_Type type)`

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

The attribute definition.

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.2** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
Callpath)(void *userData, OTF2_CallpathRef self, OTF2_CallpathRef  
parent, OTF2_RegionRef region)`

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

The callpath definition.

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.8.2.3** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
Callsite)(void *userData, OTF2_CallsiteRef self, OTF2_StringRef  
sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion,  
OTF2_RegionRef leftRegion)`

Function pointer definition for the callback which is triggered by a *Callsite* definition record.

The callsite definition.

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.8.2.4** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
CartCoordinate)(void *userData, OTF2_CartTopologyRef cartTopology,  
uint32_t rank, uint8_t numberOfDimensions, const uint32_t *coordinates)`

Function pointer definition for the callback which is triggered by a *CartCoordinate* definition record.

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

### Parameters

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



## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

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

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.8.2.5** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_CartDimension)(void *userData, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_t size, OTF2_CartPeriodicity cartPeriodicity)`

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

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

### Parameters

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

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.8.2.6** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
CartTopology)(void *userData, OTF2_CartTopologyRef self,  
OTF2_StringRef name, OTF2_CommRef communicator, uint8_t  
numberOfDimensions, const OTF2_CartDimensionRef *cartDimensions)`

Function pointer definition for the callback which is triggered by a *CartTopology* definition record.

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

#### Parameters

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

#### Since

Version 1.3

#### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.8.2.7** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
ClockOffset)(void *userData, OTF2_TimeStamp time, int64_t offset, double  
standardDeviation)`

Function pointer definition for the callback which is triggered by a *ClockOffset* definition record.

Clock offsets are used for clock corrections.

#### Parameters

## E.8 of2/OTF2\_DefReaderCallbacks.h File Reference

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
<i>time</i>	Time when this offset was determined.
<i>offset</i>	The offset to the global clock which was determined at <i>time</i> .
<i>standard-Deviation</i>	A possible standard deviation, which can be used as a metric for the quality of the offset.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.8** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
Comm)(void *userData, OTF2_CommRef self, OTF2_StringRef name,  
OTF2_GroupRef group, OTF2_CommRef parent)`

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

The communicator definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">Comm</a> 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 <a href="#">String</a> definition.
<i>group</i>	The describing MPI group of this MPI communicator The group needs to be of type <a href="#">OTF2_GROUP_TYPE_COMM_GROUP</a> or <a href="#">OTF2_GROUP_TYPE_COMM_SELF</a> . References a <a href="#">Group</a> definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <a href="#">OTF2_UNDEFINED_COMM</a> to indicate no parent. References a <a href="#">Comm</a> definition.

### Since

Version 1.0

## Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.9** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_ - Group)(void *userData, OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_t numberOfMembers, const uint64_t *members)`

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

The group definition.

## Parameters

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

## Since

Version 1.0

## Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.10** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_ - Location)(void *userData, OTF2_LocationRef self, OTF2_StringRef name, OTF2_LocationType locationType, uint64_t numberOfEvents, OTF2_LocationGroupRef locationGroup)`

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

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

The location definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">Location</a> definition.
<i>name</i>	Name of the location References a <a href="#">String</a> 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 <a href="#">Location-Group</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.11** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
LocationGroup)(void *userData, OTF2_LocationGroupRef self,  
OTF2_StringRef name, OTF2_LocationGroupType locationGroupType,  
OTF2_SystemTreeNodeRef systemTreeParent)`

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

The location group definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">LocationGroup</a> definition.
<i>name</i>	Name of the group. References a <a href="#">String</a> definition.
<i>location-GroupType</i>	Type of this group.
<i>systemTreeParent</i>	Parent of this location group in the system tree. References a <a href="#">SystemTreeNode</a> definition.

**Since**

Version 1.0

**Returns**

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.8.2.12** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
LocationGroupProperty)(void *userData, OTF2_LocationGroupRef  
locationGroup, OTF2_StringRef name, OTF2_StringRef value)`

Function pointer definition for the callback which is triggered by a [\*LocationGroup-Property\*](#) definition record.

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

**Parameters**

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

**Since**

Version 1.3

**Returns**

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.8.2.13** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
LocationProperty)(void *userData, OTF2_LocationRef location,  
OTF2_StringRef name, OTF2_StringRef value)`

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

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

**Parameters**


---

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

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

### Since

Version 1.3

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.14** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_MappingTable)(void *userData, OTF2_MappingType mappingType, const OTF2_IdMap *idMap)`

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

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

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.15** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_MetricClass)(void *userData, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_MetricMemberRef *metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)`

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

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

#### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">MetricClass</a> definition.
<i>numberOfMetrics</i>	Number of metrics within the set.
<i>metricMembers</i>	List of metric members. References a <a href="#">MetricMember</a> definition.
<i>metricOccurrence</i>	Defines occurrence of a metric set.
<i>recorderKind</i>	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.16** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_MetricClassRecorder)(void *userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)`

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

The metric class recorder definition.

#### Parameters

---



## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
<i>metricClass</i>	Parent <a href="#">MetricClass</a> definition to which this one is a supplementary definition. References a <a href="#">MetricClass</a> definition.
<i>recorder</i>	The location which recorded the referenced metric class. References a <a href="#">Location</a> definition.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.17** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
MetricInstance)(void *userData, OTF2_MetricRef self,  
OTF2_MetricRef metricClass, OTF2_LocationRef recorder,  
OTF2_MetricScope metricScope, uint64_t scope)`

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

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type [OTF2\\_METRIC\\_ASYNCHRONOUS](#).

### Parameters

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

### Since

Version 1.0

## Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.18** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_ - MetricMember)(void *userData, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2_StringRef unit)`

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

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

## Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">MetricMember</a> definition.
<i>name</i>	Name of the metric. References a <a href="#">String</a> definition.
<i>description</i>	Description of the metric. References a <a href="#">String</a> definition.
<i>metricType</i>	Metric type: PAPI, etc.
<i>metricMode</i>	Metric mode: accumulative, fix, relative, etc.
<i>valueType</i>	Type of the value. Only <a href="#">OTF2_TYPE_INT64</a> , <a href="#">OTF2_TYPE_UINT64</a> , and <a href="#">OTF2_TYPE_DOUBLE</a> are valid types. If this metric member is recorded in an <a href="#">Metric</a> event, than this type and the type in the event must match.
<i>metricBase</i>	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$ , to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be <a href="#">OTF2_BASE_BINARY</a> 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 <a href="#">String</a> definition.

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.19** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_ -  
Parameter)(void *userData, OTF2_ParameterRef self, OTF2_StringRef  
name, OTF2_ParameterType parameterType)`

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

The parameter definition.

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.20** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_ -  
Region)(void *userData, OTF2_RegionRef self, OTF2_StringRef  
name, OTF2_StringRef canonicalName, OTF2_StringRef description,  
OTF2_RegionRole regionRole, OTF2_Paradigm paradigm,  
OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t  
beginLineNumber, uint32_t endLineNumber)`

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

The region definition.

## APPENDIX E. FILE DOCUMENTATION

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.21** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
RmaWin)(void *userData, OTF2_RmaWinRef self, OTF2_StringRef  
name, OTF2_CommRef comm)`

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

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

### Parameters

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

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

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

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.22** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
String)(void *userData, OTF2_StringRef self, const char  
*string)`

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

The string definitions.

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.23** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
SystemTreeNode)(void *userData, OTF2_SystemTreeNodeRef  
self, OTF2_StringRef name, OTF2_StringRef className,  
OTF2_SystemTreeNodeRef parent)`

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

The system tree node definition.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">SystemTreeNode</a> definition.
<i>name</i>	Free form instance name of this node. References a <a href="#">String</a> definition.
<i>className</i>	Free form class name of this node References a <a href="#">String</a> definition.
<i>parent</i>	Parent id of this node. May be <a href="#">OTF2_UNDEFINED_SYSTEM_TREE_NODE</a> to indicate that there is no parent. References a <a href="#">SystemTreeNode</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.8.2.24** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
SystemTreeNodeDomain)(void *userData, OTF2_SystemTreeNodeRef  
systemTreeNode, OTF2_SystemTreeDomain systemTreeDomain)`

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

The system tree node domain definition.

### Parameters

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

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

**E.8.2.25** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
SystemTreeNodeProperty)(void *userData, OTF2_SystemTreeNodeRef  
systemTreeNode, OTF2_StringRef name, OTF2_StringRef value)`

Function pointer definition for the callback which is triggered by a *SystemTreeNodeProperty* definition record.

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

### Parameters

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

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

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

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

### Parameters

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

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

### E.8.3 Function Documentation

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

Clears a struct for the definition callbacks.

#### Parameters

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

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

Deallocates a struct for the definition callbacks.

#### Parameters

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

**E.8.3.3** `OTF2_DefReaderCallbacks* OTF2_DefReaderCallbacks_New ( void )`

Allocates a new struct for the definition callbacks.

#### Returns

A newly allocated struct of type [OTF2\\_DefReaderCallbacks](#).

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

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

#### Parameters

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

---



## E.8 of2/OTF2\_DefReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful  
[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid `defReaderCallbacks` argument

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

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

### Parameters

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

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful  
[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid `defReaderCallbacks` argument

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

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

### Parameters

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

---

## APPENDIX E. FILE DOCUMENTATION

---

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.8.3.7** **OTF2\_ErrorCode** **OTF2\_DefReaderCallbacks\_SetCartCoordinateCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_CartCoordinate** *cartCoordinateCallback* )

Registers the callback for the *CartCoordinate* definition.

### Parameters

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

### Since

Version 1.3

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.8.3.8** **OTF2\_ErrorCode** **OTF2\_DefReaderCallbacks\_SetCartDimensionCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_CartDimension** *cartDimensionCallback* )

Registers the callback for the *CartDimension* definition.

### Parameters

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

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

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

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

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

Registers the callback for the *CartTopology* definition.

### Parameters

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

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

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

Registers the callback for the *ClockOffset* definition.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.11** **OTF2\_ErrorCode** **OTF2\_DefReaderCallbacks\_SetCommCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_Comm** *commCallback* )

Registers the callback for the *Comm* definition.

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.12** **OTF2\_ErrorCode** **OTF2\_DefReaderCallbacks\_SetGroupCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_Group** *groupCallback* )

Registers the callback for the *Group* definition.

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.13** **OTF2\_ErrorCode** **OTF2\_DefReaderCallbacks\_SetLocationCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_Location** *locationCallback* )

Registers the callback for the *Location* definition.

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.14** **OTF2\_ErrorCode** **OTF2\_DefReaderCallbacks\_SetLocationGroupCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_LocationGroup** *locationGroupCallback* )

Registers the callback for the *LocationGroup* definition.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.15** **OTF2\_StatusCode** **OTF2\_DefReaderCallbacks\_-**  
**SetLocationGroupPropertyCallback (** **OTF2\_DefReaderCallbacks**  
**\*** *defReaderCallbacks***,** **OTF2\_DefReaderCallback\_-**  
**LocationGroupProperty** *locationGroupPropertyCallback*  
**)**

Registers the callback for the *LocationGroupProperty* definition.

### Parameters

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

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

**E.8.3.16** **OTF2\_ErrorCode** **OTF2\_DefReaderCallbacks\_SetLocationPropertyCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_LocationProperty** *locationPropertyCallback*  
)

Registers the callback for the *LocationProperty* definition.

### Parameters

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

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.17** **OTF2\_ErrorCode** **OTF2\_DefReaderCallbacks\_SetMappingTableCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_MappingTable** *mappingTableCallback* )

Registers the callback for the *MappingTable* definition.

### Parameters

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

### Since

Version 1.0

## APPENDIX E. FILE DOCUMENTATION

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.8.3.18** **OTF2\_StatusCode** **OTF2\_DefReaderCallbacks\_SetMetricClassCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_MetricClass** *metricClassCallback* )

Registers the callback for the *MetricClass* definition.

### Parameters

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

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.8.3.19** **OTF2\_StatusCode** **OTF2\_DefReaderCallbacks\_SetMetricClassRecorderCallback**  
( **OTF2\_DefReaderCallbacks** \* *defReaderCallbacks*,  
**OTF2\_DefReaderCallback\_MetricClassRecorder**  
*metricClassRecorderCallback* )

Registers the callback for the *MetricClassRecorder* definition.

### Parameters

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



## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid `defReaderCallbacks` argument

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

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

### Parameters

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

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid `defReaderCallbacks` argument

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

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

### Parameters

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

---

## APPENDIX E. FILE DOCUMENTATION

---

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.22** *OTF2\_StatusCode* *OTF2.DefReaderCallbacks.SetParameterCallback*  
( *OTF2.DefReaderCallbacks* \* *defReaderCallbacks*,  
*OTF2.DefReaderCallback\_Parameter* *parameterCallback* )

Registers the callback for the *Parameter* definition.

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.23** *OTF2\_StatusCode* *OTF2.DefReaderCallbacks.SetRegionCallback*  
( *OTF2.DefReaderCallbacks* \* *defReaderCallbacks*,  
*OTF2.DefReaderCallback\_Region* *regionCallback* )

Registers the callback for the *Region* definition.

## E.8 of2/OTF2\_DefReaderCallbacks.h File Reference

---

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.24** *OTF2\_StatusCode* *OTF2\_DefReaderCallbacks\_SetRmaWinCallback*  
( *OTF2\_DefReaderCallbacks* \* *defReaderCallbacks*,  
*OTF2\_DefReaderCallback\_RmaWin* *rmaWinCallback* )

Registers the callback for the *RmaWin* definition.

### Parameters

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

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.25** *OTF2\_StatusCode* *OTF2\_DefReaderCallbacks\_SetStringCallback*  
( *OTF2\_DefReaderCallbacks* \* *defReaderCallbacks*,  
*OTF2\_DefReaderCallback\_String* *stringCallback* )

Registers the callback for the *String* definition.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.8.3.26** *OTF2\_*ErrorCode *OTF2\_DefReaderCallbacks\_SetSystemTreeNodeCallback*  
( *OTF2\_DefReaderCallbacks* \* *defReaderCallbacks*,  
*OTF2\_DefReaderCallback\_SystemTreeNode* *systemTreeNodeCallback* )

Registers the callback for the *SystemTreeNode* definition.

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

## E.8 otf2/OTF2\_DefReaderCallbacks.h File Reference

---

**E.8.3.27** **OTF2\_ErrorCode** **OTF2.DefReaderCallbacks\_-**  
**SetSystemTreeNodeDomainCallback** ( **OTF2\_DefReaderCallbacks**  
\* **defReaderCallbacks**, **OTF2\_DefReaderCallback\_-**  
**SystemTreeNodeDomain** **systemTreeNodeDomainCallback**  
)

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

### Parameters

<i>defReader- Callbacks</i>	Struct for all callbacks.
<i>sys- temTreeN- odeDo- mainCall- back</i>	Function which should be called for all <a href="#">SystemTreeNodeDomain</a> definitions.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid `defReaderCallbacks` argument

**E.8.3.28** **OTF2\_ErrorCode** **OTF2.DefReaderCallbacks\_-**  
**SetSystemTreeNodePropertyCallback** ( **OTF2\_DefReaderCallbacks**  
\* **defReaderCallbacks**, **OTF2\_DefReaderCallback\_-**  
**SystemTreeNodeProperty** **systemTreeNodePropertyCallback**  
)

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

### Parameters

<i>defReader- Callbacks</i>	Struct for all callbacks.
<i>sys- temTreeN- odeProp- ertyCallback</i>	Function which should be called for all <a href="#">SystemTreeNodeProperty</a> definitions.

**Since**

Version 1.2

**Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.8.3.29** **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

**E.9 otF2/OTF2\_DefWriter.h File Reference**

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

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

**Typedefs**

- typedef struct OTF2\_DefWriter\_struct **OTF2\_DefWriter**  
*Handle definition for the external definition writer.*

## E.9 otf2/OTF2\_DefWriter.h File Reference

---

### Functions

- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_GetLocationID](#) (const [OTF2\\_DefWriter](#) \*writer, [OTF2\\_LocationRef](#) \*location)  
*Returns the location ID of the location which is related to the writer object.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteAttribute](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_AttributeRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) description, [OTF2\\_Type](#) type)  
*Writes a Attribute definition record into the DefWriter.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteCallpath](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_CallpathRef](#) self, [OTF2\\_CallpathRef](#) parent, [OTF2\\_RegionRef](#) region)  
*Writes a Callpath definition record into the DefWriter.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteCallsite](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_CallsiteRef](#) self, [OTF2\\_StringRef](#) sourceFile, [uint32\\_t](#) lineNumber, [OTF2\\_RegionRef](#) enteredRegion, [OTF2\\_RegionRef](#) leftRegion)  
*Writes a Callsite definition record into the DefWriter.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteCartCoordinate](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_CartTopologyRef](#) cartTopology, [uint32\\_t](#) rank, [uint8\\_t](#) numberOfDimensions, const [uint32\\_t](#) \*coordinates)  
*Writes a CartCoordinate definition record into the DefWriter.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteCartDimension](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_CartDimensionRef](#) self, [OTF2\\_StringRef](#) name, [uint32\\_t](#) size, [OTF2\\_CartPeriodicity](#) cartPeriodicity)  
*Writes a CartDimension definition record into the DefWriter.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteCartTopology](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_CartTopologyRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_CommRef](#) communicator, [uint8\\_t](#) numberOfDimensions, const [OTF2\\_CartDimensionRef](#) \*cartDimensions)  
*Writes a CartTopology definition record into the DefWriter.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteClockOffset](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_TimeStamp](#) time, [int64\\_t](#) offset, double standardDeviation)  
*Writes a ClockOffset definition record into the DefWriter.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteComm](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_CommRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_GroupRef](#) group, [OTF2\\_CommRef](#) parent)  
*Writes a Comm definition record into the DefWriter.*
- [OTF2\\_ErrorCode OTF2\\_DefWriter\\_WriteGroup](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_GroupRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_GroupType](#) groupType, [OTF2\\_Paradigm](#) paradigm, [OTF2\\_GroupFlag](#) groupFlags, [uint32\\_t](#) numberOfMembers, const [uint64\\_t](#) \*members)

*Writes a Group definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteLocation](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_LocationRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_LocationType](#) locationType, [uint64\\_t](#) numberOfEvents, [OTF2\\_LocationGroupRef](#) locationGroup)

*Writes a Location definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteLocationGroup](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_LocationGroupRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_LocationGroupType](#) locationGroupType, [OTF2\\_SystemTreeNodeRef](#) systemTreeParent)

*Writes a LocationGroup definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteLocationGroupProperty](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_LocationGroupRef](#) locationGroup, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

*Writes a LocationGroupProperty definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteLocationProperty](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_LocationRef](#) location, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

*Writes a LocationProperty definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteMappingTable](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_MappingType](#) mappingType, const [OTF2\\_IdMap](#) \*idMap)

*Writes a MappingTable definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteMetricClass](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_MetricRef](#) self, [uint8\\_t](#) numberOfMetrics, const [OTF2\\_MetricMemberRef](#) \*metricMembers, [OTF2\\_MetricOccurrence](#) metricOccurrence, [OTF2\\_RecorderKind](#) recorderKind)

*Writes a MetricClass definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteMetricClassRecorder](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_MetricRef](#) metricClass, [OTF2\\_LocationRef](#) recorder)

*Writes a MetricClassRecorder definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteMetricInstance](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_MetricRef](#) self, [OTF2\\_MetricRef](#) metricClass, [OTF2\\_LocationRef](#) recorder, [OTF2\\_MetricScope](#) metricScope, [uint64\\_t](#) scope)

*Writes a MetricInstance definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteMetricMember](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_MetricMemberRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) description, [OTF2\\_MetricType](#) metricType, [OTF2\\_MetricMode](#) metricMode, [OTF2\\_Type](#) valueType, [OTF2\\_MetricBase](#) metricBase, [int64\\_t](#) exponent, [OTF2\\_StringRef](#) unit)

*Writes a MetricMember definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteParameter](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_ParameterRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_ParameterType](#) parameterType)



## E.9 otf2/OTF2\_DefWriter.h File Reference

---

*Writes a Parameter definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteRegion](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_RegionRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) canonicalName, [OTF2\\_StringRef](#) description, [OTF2\\_RegionRole](#) regionRole, [OTF2\\_Paradigm](#) paradigm, [OTF2\\_RegionFlag](#) regionFlags, [OTF2\\_StringRef](#) sourceFile, [uint32\\_t](#) beginLineNumber, [uint32\\_t](#) endLineNumber)

*Writes a Region definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteRmaWin](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_RmaWinRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_CommRef](#) comm)

*Writes a RmaWin definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteString](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_StringRef](#) self, const char \*string)

*Writes a String definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteSystemTreeNode](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_SystemTreeNodeRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) className, [OTF2\\_SystemTreeNodeRef](#) parent)

*Writes a SystemTreeNode definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteSystemTreeNodeDomain](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_SystemTreeNodeRef](#) systemTreeNode, [OTF2\\_SystemTreeDomain](#) systemTreeDomain)

*Writes a SystemTreeNodeDomain definition record into the DefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_WriteSystemTreeNodeProperty](#) ([OTF2\\_DefWriter](#) \*writer, [OTF2\\_SystemTreeNodeRef](#) systemTreeNode, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

*Writes a SystemTreeNodeProperty definition record into the DefWriter.*

### E.9.1 Detailed Description

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

#### Source Template:

*templates/OTF2\_DefWriter.templ.h*

### E.9.2 Function Documentation

#### E.9.2.1 [OTF2\\_ErrorCode](#) [OTF2\\_DefWriter\\_GetLocationID](#) ( const [OTF2\\_DefWriter](#) \* writer, [OTF2\\_LocationRef](#) \* location )

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

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.9.2.2** **OTF2\_ErrorCode** **OTF2.DefWriter.WriteAttribute (** **OTF2.DefWriter**  
\* **writer**, **OTF2\_AttributeRef** **self**, **OTF2\_StringRef** **name**,  
**OTF2\_StringRef** **description**, **OTF2\_Type** **type** )

Writes a Attribute definition record into the DefWriter.

The attribute definition.

### Parameters

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

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.9.2.3** **OTF2\_ErrorCode** **OTF2.DefWriter.WriteCallpath (** **OTF2.DefWriter**  
\* **writer**, **OTF2\_CallpathRef** **self**, **OTF2\_CallpathRef** **parent**,  
**OTF2\_RegionRef** **region** )

Writes a Callpath definition record into the DefWriter.

The callpath definition.

### Parameters

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

## E.9 otF2/OTF2\_DefWriter.h File Reference

---

<i>self</i>	The unique identifier for this <a href="#">Callpath</a> definition.
<i>parent</i>	The parent of this callpath. References a <a href="#">Callpath</a> definition.
<i>region</i>	The region of this callpath. References a <a href="#">Region</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.4** `OTF2_ErrorCode OTF2_DefWriter_WriteCallsite ( OTF2_DefWriter * writer, OTF2_CallsiteRef self, OTF2_StringRef sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion, OTF2_RegionRef leftRegion )`

Writes a Callsite definition record into the DefWriter.

The callsite definition.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <a href="#">Callsite</a> definition.
<i>sourceFile</i>	The source file where this call was made. References a <a href="#">String</a> 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 <a href="#">Region</a> definition.
<i>leftRegion</i>	The region which made the call. References a <a href="#">Region</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.5** `OTF2_ErrorCode OTF2_DefWriter_WriteCartCoordinate ( OTF2_DefWriter * writer, OTF2_CartTopologyRef cartTopology, uint32_t rank, uint8_t numberOfDimensions, const uint32_t * coordinates )`

Writes a CartCoordinate definition record into the DefWriter.

## APPENDIX E. FILE DOCUMENTATION

---

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

### Parameters

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

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.6** `OTF2_ErrorCode OTF2_DefWriter_WriteCartDimension ( OTF2_DefWriter * writer, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_t size, OTF2_CartPeriodicity cartPeriodicity )`

Writes a CartDimension definition record into the DefWriter.

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

### Parameters

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

### Since

Version 1.3

## E.9 otf2/OTF2\_DefWriter.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.9.2.7** `OTF2_ErrorCode OTF2_DefWriter_WriteCartTopology ( OTF2_DefWriter * writer, OTF2_CartTopologyRef self, OTF2_StringRef name, OTF2_CommRef communicator, uint8_t numberOfDimensions, const OTF2_CartDimensionRef * cartDimensions )`

Writes a CartTopology definition record into the DefWriter.

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

### Parameters

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

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.9.2.8** `OTF2_ErrorCode OTF2_DefWriter_WriteClockOffset ( OTF2_DefWriter * writer, OTF2_TimeStamp time, int64_t offset, double standardDeviation )`

Writes a ClockOffset definition record into the DefWriter.

Clock offsets are used for clock corrections.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>writer</i>	Writer object.
<i>time</i>	Time when this offset was determined.
<i>offset</i>	The offset to the global clock which was determined at <i>time</i> .
<i>standard-Deviation</i>	A possible standard deviation, which can be used as a metric for the quality of the offset.

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.9.2.9** `OTF2_ErrorCode OTF2_DefWriter_WriteComm ( OTF2_DefWriter * writer,  
OTF2_CommRef self, OTF2_StringRef name, OTF2_GroupRef group,  
OTF2_CommRef parent )`

Writes a Comm definition record into the DefWriter.

The communicator definition.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <a href="#"><i>Comm</i></a> 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 <a href="#"><i>String</i></a> definition.
<i>group</i>	The describing MPI group of this MPI communicator The group needs to be of type <a href="#"><i>OTF2_GROUP_TYPE_COMM_GROUP</i></a> or <a href="#"><i>OTF2_GROUP_TYPE_COMM_SELF</i></a> . References a <a href="#"><i>Group</i></a> definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <a href="#"><i>OTF2_UNDEFINED_COMM</i></a> to indicate no parent. References a <a href="#"><i>Comm</i></a> definition.

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

## E.9 otf2/OTF2\_DefWriter.h File Reference

---

**E.9.2.10** `OTF2_ErrorCode OTF2_DefWriter_WriteGroup ( OTF2_DefWriter * writer,  
OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType  
groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags,  
uint32_t numberOfMembers, const uint64_t * members )`

Writes a Group definition record into the DefWriter.

The group definition.

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.11** `OTF2_ErrorCode OTF2_DefWriter_WriteLocation ( OTF2_DefWriter  
* writer, OTF2_LocationRef self, OTF2_StringRef name,  
OTF2_LocationType locationType, uint64_t numberOfEvents,  
OTF2_LocationGroupRef locationGroup )`

Writes a Location definition record into the DefWriter.

The location definition.

### Parameters

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

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>numberOfEvents</i>	Number of events this location has recorded.
<i>location-Group</i>	Location group which includes this location. References a <a href="#">Location-Group</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.12** **OTF2\_ErrorCode** **OTF2\_DefWriter\_WriteLocationGroup** ( **OTF2\_DefWriter**  
\* *writer*, **OTF2\_LocationGroupRef** *self*, **OTF2\_StringRef**  
*name*, **OTF2\_LocationGroupType** *locationGroupType*,  
**OTF2\_SystemTreeNodeRef** *systemTreeParent* )

Writes a LocationGroup definition record into the DefWriter.

The location group definition.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <a href="#">LocationGroup</a> definition.
<i>name</i>	Name of the group. References a <a href="#">String</a> definition.
<i>location-GroupType</i>	Type of this group.
<i>systemTreeParent</i>	Parent of this location group in the system tree. References a <a href="#">SystemTreeNode</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.



## E.9 otf2/OTF2\_DefWriter.h File Reference

---

**E.9.2.13** **OTF2\_ErrorCode** **OTF2\_DefWriter\_WriteLocationGroupProperty** (  
    **OTF2\_DefWriter** \* *writer*, **OTF2\_LocationGroupRef** *locationGroup*,  
    **OTF2\_StringRef** *name*, **OTF2\_StringRef** *value* )

Writes a LocationGroupProperty definition record into the DefWriter.

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

### Parameters

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

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.9.2.14** **OTF2\_ErrorCode** **OTF2\_DefWriter\_WriteLocationProperty** (  
    **OTF2\_DefWriter** \* *writer*, **OTF2\_LocationRef** *location*,  
    **OTF2\_StringRef** *name*, **OTF2\_StringRef** *value* )

Writes a LocationProperty definition record into the DefWriter.

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

### Parameters

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

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## APPENDIX E. FILE DOCUMENTATION

**E.9.2.15** `OTF2_ErrorCode OTF2_DefWriter_WriteMappingTable ( OTF2_DefWriter * writer, OTF2_MappingType mappingType, const OTF2_IdMap * idMap )`

Writes a MappingTable definition record into the DefWriter.

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

### Parameters

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

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.9.2.16** `OTF2_ErrorCode OTF2_DefWriter_WriteMetricClass ( OTF2_DefWriter * writer, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_MetricMemberRef * metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind )`

Writes a MetricClass definition record into the DefWriter.

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

### Parameters

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

## E.9 otf2/OTF2\_DefWriter.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.9.2.17** `OTF2_ErrorCode OTF2_DefWriter_WriteMetricClassRecorder ( OTF2_DefWriter * writer, OTF2_MetricRef metricClass, OTF2_LocationRef recorder )`

Writes a MetricClassRecorder definition record into the DefWriter.

The metric class recorder definition.

### Parameters

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

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.9.2.18** `OTF2_ErrorCode OTF2_DefWriter_WriteMetricInstance ( OTF2_DefWriter * writer, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope )`

Writes a MetricInstance definition record into the DefWriter.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type [\*OTF2\\_METRIC\\_ASYNCHRONOUS\*](#).

### Parameters

---

## APPENDIX E. FILE DOCUMENTATION

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.9.2.19** *OTF2\_*ErrorCode *OTF2\_DefWriter\_WriteMetricMember* ( *OTF2\_DefWriter* \* *writer*, *OTF2\_MetricMemberRef* *self*, *OTF2\_StringRef* *name*, *OTF2\_StringRef* *description*, *OTF2\_MetricType* *metricType*, *OTF2\_MetricMode* *metricMode*, *OTF2\_Type* *valueType*, *OTF2\_MetricBase* *metricBase*, *int64\_t* *exponent*, *OTF2\_StringRef* *unit* )

Writes a MetricMember definition record into the DefWriter.

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

### Parameters

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

## E.9 otf2/OTF2\_DefWriter.h File Reference

---

<i>metricBase</i>	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$ , to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be <a href="#">OTF2_BASE_BINARY</a> 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 <a href="#">String</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.20** `OTF2_ErrorCode OTF2_DefWriter_WriteParameter ( OTF2_DefWriter  
* writer, OTF2_ParameterRef self, OTF2_StringRef name,  
OTF2_ParameterType parameterType )`

Writes a Parameter definition record into the DefWriter.

The parameter definition.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <a href="#">Parameter</a> definition.
<i>name</i>	Name of the parameter (variable name etc.) References a <a href="#">String</a> definition.
<i>parameter-Type</i>	Type of the parameter, <a href="#">OTF2_ParameterType</a> for possible types.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.9.2.21** `OTF2_ErrorCode OTF2_DefWriter_WriteRegion ( OTF2_DefWriter *  
writer, OTF2_RegionRef self, OTF2_StringRef name, OTF2_StringRef  
canonicalName, OTF2_StringRef description, OTF2_RegionRole  
regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags,  
OTF2_StringRef sourceFile, uint32_t beginLineNumber, uint32_t  
endLineNumber )`

Writes a Region definition record into the DefWriter.

The region definition.

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.9.2.22** `OTF2_ErrorCode OTF2_DefWriter_WriteRmaWin ( OTF2_DefWriter  
* writer, OTF2_RmaWinRef self, OTF2_StringRef name,  
OTF2_CommRef comm )`

Writes a RmaWin definition record into the DefWriter.

## E.9 oftf2/OTF2\_DefWriter.h File Reference

---

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

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <a href="#">RmaWin</a> definition.
<i>name</i>	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a <a href="#">String</a> definition.
<i>comm</i>	Communicator object used to create the window. References a <a href="#">Comm</a> definition.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.23** `OTF2_ErrorCode OTF2_DefWriter_WriteString ( OTF2_DefWriter * writer,  
OTF2_StringRef self, const char * string )`

Writes a String definition record into the DefWriter.

The string definitions.

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.24** **OTF2\_ErrorCode** **OTF2.DefWriter.WriteSystemTreeNode** (  
**OTF2\_DefWriter** \* *writer*, **OTF2\_SystemTreeNodeRef**  
*self*, **OTF2\_StringRef** *name*, **OTF2\_StringRef** *className*,  
**OTF2\_SystemTreeNodeRef** *parent* )

Writes a SystemTreeNode definition record into the DefWriter.

The system tree node definition.

#### Parameters

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

#### Since

Version 1.0

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.9.2.25** **OTF2\_ErrorCode** **OTF2.DefWriter.WriteSystemTreeNodeDomain** (  
**OTF2\_DefWriter** \* *writer*, **OTF2\_SystemTreeNodeRef** *systemTreeNode*,  
**OTF2\_SystemTreeDomain** *systemTreeDomain* )

Writes a SystemTreeNodeDomain definition record into the DefWriter.

The system tree node domain definition.

#### Parameters

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



## E.10 otf2/OTF2\_Events.h File Reference

---

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.9.2.26** `OTF2_ErrorCode OTF2_DefWriter_WriteSystemTreeNodeProperty (`  
`OTF2_DefWriter * writer, OTF2_SystemTreeNodeRef systemTreeNode,`  
`OTF2_StringRef name, OTF2_StringRef value )`

Writes a SystemTreeNodeProperty definition record into the DefWriter.

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

### Parameters

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

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.10 otf2/OTF2\_Events.h File Reference

Enums and types used in event records.

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

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

### Data Structures

- union [OTF2\\_MetricValue](#)  
*Metric value.*

**Typedefs**

- typedef uint8\_t [OTF2\\_CollectiveOp](#)  
*Wrapper for enum [OTF2\\_CollectiveOp\\_enum](#).*
- typedef uint8\_t [OTF2\\_LockType](#)  
*Wrapper for enum [OTF2\\_LockType\\_enum](#).*
- typedef uint8\_t [OTF2\\_MeasurementMode](#)  
*Wrapper for enum [OTF2\\_MeasurementMode\\_enum](#).*
- typedef uint8\_t [OTF2\\_RmaAtomicType](#)  
*Wrapper for enum [OTF2\\_RmaAtomicType\\_enum](#).*
- typedef uint32\_t [OTF2\\_RmaSyncLevel](#)  
*Wrapper for enum [OTF2\\_RmaSyncLevel\\_enum](#).*
- typedef uint8\_t [OTF2\\_RmaSyncType](#)  
*Wrapper for enum [OTF2\\_RmaSyncType\\_enum](#).*

**Enumerations**

- enum [OTF2\\_CollectiveOp\\_enum](#) {  
[OTF2\\_COLLECTIVE\\_OP\\_BARRIER](#) = 0,  
[OTF2\\_COLLECTIVE\\_OP\\_BCAST](#) = 1,  
[OTF2\\_COLLECTIVE\\_OP\\_GATHER](#) = 2,  
[OTF2\\_COLLECTIVE\\_OP\\_GATHERV](#) = 3,  
[OTF2\\_COLLECTIVE\\_OP\\_SCATTER](#) = 4,  
[OTF2\\_COLLECTIVE\\_OP\\_SCATTERV](#) = 5,  
[OTF2\\_COLLECTIVE\\_OP\\_ALLGATHER](#) = 6,  
[OTF2\\_COLLECTIVE\\_OP\\_ALLGATHERV](#) = 7,  
[OTF2\\_COLLECTIVE\\_OP\\_ALLTOALL](#) = 8,  
[OTF2\\_COLLECTIVE\\_OP\\_ALLTOALLV](#) = 9,  
[OTF2\\_COLLECTIVE\\_OP\\_ALLTOALLW](#) = 10,  
[OTF2\\_COLLECTIVE\\_OP\\_ALLREDUCE](#) = 11,  
[OTF2\\_COLLECTIVE\\_OP\\_REDUCE](#) = 12,  
[OTF2\\_COLLECTIVE\\_OP\\_REDUCE\\_SCATTER](#) = 13,  
[OTF2\\_COLLECTIVE\\_OP\\_SCAN](#) = 14,  
[OTF2\\_COLLECTIVE\\_OP\\_EXSCAN](#) = 15,  
[OTF2\\_COLLECTIVE\\_OP\\_REDUCE\\_SCATTER\\_BLOCK](#) = 16,  
[OTF2\\_COLLECTIVE\\_OP\\_CREATE\\_HANDLE](#) = 17,  
}

## E.10 otf2/OTF2\_Events.h File Reference

---

OTF2\_COLLECTIVE\_OP\_DESTROY\_HANDLE = 18,  
OTF2\_COLLECTIVE\_OP\_ALLOCATE = 19,  
OTF2\_COLLECTIVE\_OP\_DEALLOCATE = 20,  
OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE\_AND\_ALLOCATE = 21,  
OTF2\_COLLECTIVE\_OP\_DESTROY\_HANDLE\_AND\_DEALLOCATE =  
22 }

*Types of collective operations.*

- enum OTF2\_LockType\_enum {  
OTF2\_LOCK\_EXCLUSIVE = 0,  
OTF2\_LOCK\_SHARED = 1 }

*General Lock Type.*

- enum OTF2\_MeasurementMode\_enum {  
OTF2\_MEASUREMENT\_ON = 1,  
OTF2\_MEASUREMENT\_OFF = 2 }

*Types for use in the MeasurementOnOff event.*

- enum OTF2\_RmaAtomicType\_enum {  
OTF2\_RMA\_ATOMIC\_TYPE\_ACCUMULATE = 0,  
OTF2\_RMA\_ATOMIC\_TYPE\_INCREMENT = 1,  
OTF2\_RMA\_ATOMIC\_TYPE\_TEST\_AND\_SET = 2,  
OTF2\_RMA\_ATOMIC\_TYPE\_COMPARE\_AND\_SWAP = 3,  
OTF2\_RMA\_ATOMIC\_TYPE\_SWAP = 4,  
OTF2\_RMA\_ATOMIC\_TYPE\_FETCH\_AND\_ADD = 5,  
OTF2\_RMA\_ATOMIC\_TYPE\_FETCH\_AND\_INCREMENT = 6 }

*RMA Atomic Operation Type.*

- enum OTF2\_RmaSyncLevel\_enum {  
OTF2\_RMA\_SYNC\_LEVEL\_NONE = 0,  
OTF2\_RMA\_SYNC\_LEVEL\_PROCESS = ( 1 << 0 ),  
OTF2\_RMA\_SYNC\_LEVEL\_MEMORY = ( 1 << 1 ) }

*Synchronization level used in RMA synchronization records.*

- enum OTF2\_RmaSyncType\_enum {  
OTF2\_RMA\_SYNC\_TYPE\_MEMORY = 0,  
OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_IN = 1,  
OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_OUT = 2 }

*Type of direct RMA synchronization call.*

**E.10.1 Detailed Description**

Enums and types used in event records.

**Source Template:**

*templates/OTF2\_Events.tmpl.h*

**E.10.2 Enumeration Type Documentation****E.10.2.1 enum OTF2\_CollectiveOp\_enum**

Types of collective operations.

**Since**

Version 1.0

**Enumerator:**

- OTF2\_COLLECTIVE\_OP\_BARRIER*** Barrier synchronization.
- OTF2\_COLLECTIVE\_OP\_BCAST*** Broadcast data from one member to all group members.
- OTF2\_COLLECTIVE\_OP\_GATHER*** Gather data from all group members to one member.
- OTF2\_COLLECTIVE\_OP\_GATHERV*** Gather data from all group members to one member, varying count of data from each member.
- OTF2\_COLLECTIVE\_OP\_SCATTER*** Scatter data from one member to all group members.
- OTF2\_COLLECTIVE\_OP\_SCATTERV*** Scatter data from one member to all group members, varying count of data from each member.
- OTF2\_COLLECTIVE\_OP\_ALLGATHER*** Gather data from all group members, all members of the group will receive the result.
- OTF2\_COLLECTIVE\_OP\_ALLGATHERV*** Gather data from all group members, varying count of data from each member, all members of the group will receive the result.
- OTF2\_COLLECTIVE\_OP\_ALLTOALL*** Collective scatter/gather operation (complete exchange)
- OTF2\_COLLECTIVE\_OP\_ALLTOALLV*** Collective scatter/gather operation (complete exchange), varying count of data from each member.
- OTF2\_COLLECTIVE\_OP\_ALLTOALLW*** Collective scatter/gather operation (complete exchange), varying count of data from each member, varying data type from each member.

## E.10 of2/OTF2\_Events.h File Reference

---

- OTF2\_COLLECTIVE\_OP\_ALLREDUCE*** Collective reduction operation, all members of the group will receive the result.
- OTF2\_COLLECTIVE\_OP\_REDUCE*** Collective reduction operation.
- OTF2\_COLLECTIVE\_OP\_REDUCE\_SCATTER*** Collective reduce/scatter operation, varying size of scattered blocks.
- OTF2\_COLLECTIVE\_OP\_SCAN*** Collective scan operation across all members of a group.
- OTF2\_COLLECTIVE\_OP\_EXSCAN*** Collective exclusive scan operation across all members of a group.
- OTF2\_COLLECTIVE\_OP\_REDUCE\_SCATTER\_BLOCK*** Collective reduce/scatter operation.
- OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE*** Collectively create a handle (ie. MPI\_Win, MPI\_Comm, MPI\_File).
- OTF2\_COLLECTIVE\_OP\_DESTROY\_HANDLE*** Collectively destroy a handle.
- OTF2\_COLLECTIVE\_OP\_ALLOCATE*** Collectively allocate memory.
- OTF2\_COLLECTIVE\_OP\_DEALLOCATE*** Collectively deallocate memory.
- OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE\_AND\_ALLOCATE*** Collectively create a handle and allocate memory.
- OTF2\_COLLECTIVE\_OP\_DESTROY\_HANDLE\_AND\_DEALLOCATE*** Collectively destroy a handle and deallocate memory.

### E.10.2.2 enum OTF2\_LockType\_enum

General Lock Type.

Since

Version 1.2

Enumerator:

- OTF2\_LOCK\_EXCLUSIVE*** Exclusive lock. No other lock will be granted.
- OTF2\_LOCK\_SHARED*** Shared lock. Other shared locks will be granted, but no exclusive locks.

**E.10.2.3 enum OTF2\_MeasurementMode\_enum**

Types for use in the MeasurementOnOff event.

**Since**

Version 1.0

**Enumerator:**

***OTF2\_MEASUREMENT\_ON*** The measurement resumed with event recording.

***OTF2\_MEASUREMENT\_OFF*** The measurement suspended with event recording.

**E.10.2.4 enum OTF2\_RmaAtomicType\_enum**

RMA Atomic Operation Type.

**Since**

Version 1.2

**Enumerator:**

***OTF2\_RMA\_ATOMIC\_TYPE\_ACCUMULATE*** Atomic accumulate operation.

***OTF2\_RMA\_ATOMIC\_TYPE\_INCREMENT*** Atomic increment operation.

***OTF2\_RMA\_ATOMIC\_TYPE\_TEST\_AND\_SET*** Atomic test-and-set operation.

***OTF2\_RMA\_ATOMIC\_TYPE\_COMPARE\_AND\_SWAP*** Atomic compare-and-swap operation.

***OTF2\_RMA\_ATOMIC\_TYPE\_SWAP*** Atomic swap operation.

**Since**

Version 1.4.

***OTF2\_RMA\_ATOMIC\_TYPE\_FETCH\_AND\_ADD*** Atomic fetch-and-add operation.

**Since**

Version 1.4.

***OTF2\_RMA\_ATOMIC\_TYPE\_FETCH\_AND\_INCREMENT*** Atomic fetch-and-increment operation.

## E.11 otf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.4.

### E.10.2.5 enum OTF2\_RmaSyncLevel\_enum

Synchronization level used in RMA synchronization records.

### Since

Version 1.2

### Enumerator:

**OTF2\_RMA\_SYNC\_LEVEL\_NONE** No process synchronization or access completion (e.g., MPI\_Win\_post, MPI\_Win\_start).

**OTF2\_RMA\_SYNC\_LEVEL\_PROCESS** Synchronize processes (e.g., MPI\_Win\_create/free).

**OTF2\_RMA\_SYNC\_LEVEL\_MEMORY** Complete memory accesses (e.g., MPI\_Win\_complete, MPI\_Win\_wait).

### E.10.2.6 enum OTF2\_RmaSyncType\_enum

Type of direct RMA synchronization call.

### Since

Version 1.2

### Enumerator:

**OTF2\_RMA\_SYNC\_TYPE\_MEMORY** Synchronize memory copy.

**OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_IN** Incoming remote notification.

**OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_OUT** Outgoing remote notification.

## E.11 otf2/OTF2\_EventSizeEstimator.h File Reference

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

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

### Typedefs

- typedef struct [OTF2\\_EventSizeEstimator](#) [OTF2\\_EventSizeEstimator](#)  
*Keeps all necessary information about the event size estimator. See [OTF2\\_EventSizeEstimator\\_struct](#) for detailed information.*

### Functions

- [OTF2\\_ErrorCode](#) [OTF2\\_EventSizeEstimator\\_Delete](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)  
*Deletes an [OTF2\\_EventSizeEstimator](#) object.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfAttributeList](#) (const [OTF2\\_EventSizeEstimator](#) \*estimator, const [OTF2\\_AttributeList](#) \*attributeList)  
*Returns the size estimate for an attribute list.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfBufferFlushEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)  
*Calculates the size estimate for the BufferFlush event.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfEnterEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)  
*Calculates the size estimate for the Enter event.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfLeaveEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)  
*Calculates the size estimate for the Leave event.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfMeasurementOnOffEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)  
*Calculates the size estimate for the MeasurementOnOff event.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfMetricEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator, [uint8\\_t](#) numberOfMetrics)  
*Calculates the size estimate for the Metric event.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfMpiCollectiveBeginEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)  
*Calculates the size estimate for the MpiCollectiveBegin event.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfMpiCollectiveEndEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)  
*Calculates the size estimate for the MpiCollectiveEnd event.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfMpiIrecvEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)  
*Calculates the size estimate for the MpiIrecv event.*
- [size\\_t](#) [OTF2\\_EventSizeEstimator\\_GetSizeOfMpiIrecvRequestEvent](#) ([OTF2\\_EventSizeEstimator](#) \*estimator)



## E.11 otf2/OTF2\_EventSizeEstimator.h File Reference

---

*Calculates the size estimate for the `MpiRecvRequest` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfMpiIsendCompleteEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `MpiIsendComplete` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfMpiIsendEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `MpiIsend` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfMpiRecvEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `MpiRecv` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfMpiRequestCancelledEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `MpiRequestCancelled` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfMpiRequestTestEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `MpiRequestTest` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfMpiSendEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `MpiSend` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfOmpAcquireLockEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `OmpAcquireLock` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfOmpForkEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `OmpFork` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfOmpJoinEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `OmpJoin` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfOmpReleaseLockEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `OmpReleaseLock` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskCompleteEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `OmpTaskComplete` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskCreateEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `OmpTaskCreate` event.*

- `size_t OTF2_EventSizeEstimator_GetSizeOfOmpTaskSwitchEvent (OTF2_EventSizeEstimator *estimator)`

*Calculates the size estimate for the `OmpTaskSwitch` event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- `size_t OTF2_EventSizeEstimator_GetSizeOfParameterIntEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ParameterInt event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfParameterStringEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ParameterString event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfParameterUnsignedIntEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ParameterUnsignedInt event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaAcquireLockEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaAcquireLock event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaAtomicEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaAtomic event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveBeginEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaCollectiveBegin event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveEndEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaCollectiveEnd event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaGetEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaGet event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaGroupSyncEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaGroupSync event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteBlockingEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaOpCompleteBlocking event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteNonBlockingEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaOpCompleteNonBlocking event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteRemoteEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaOpCompleteRemote event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpTestEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaOpTest event.*

## E.11 oftf2/OTF2\_EventSizeEstimator.h File Reference

---

- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaPutEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaPut event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaReleaseLockEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaReleaseLock event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaRequestLockEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaRequestLock event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaSyncEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaSync event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaTryLockEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaTryLock event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWaitChangeEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaWaitChange event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinCreateEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaWinCreate event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinDestroyEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the RmaWinDestroy event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadAcquireLockEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadAcquireLock event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadBeginEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadBegin event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadCreateEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadCreate event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadEnd event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadForkEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadFork event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadJoinEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadJoin event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadReleaseLockEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadReleaseLock event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCompleteEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadTaskComplete event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCreateEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadTaskCreate event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskSwitchEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadTaskSwitch event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamBeginEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadTeamBegin event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamEndEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadTeamEnd event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfThreadWaitEvent (OTF2_EventSizeEstimator *estimator)`  
*Calculates the size estimate for the ThreadWait event.*
- `size_t OTF2_EventSizeEstimator_GetSizeOfTimestamp (OTF2_EventSizeEstimator *estimator)`  
*Returns the size for an timestamp record.*
- `OTF2_EventSizeEstimator * OTF2_EventSizeEstimator_New (void)`  
*Creates a new OTF2\_EventSizeEstimator object.*
- `OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfAttributeDefinitions (OTF2_EventSizeEstimator *estimator, uint32_t numberOfAttributeDefinitions)`  
*Sets the number of Attribute definitions used.*
- `OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfCommDefinitions (OTF2_EventSizeEstimator *estimator, uint32_t numberOfCommDefinitions)`  
*Sets the number of Comm definitions used.*
- `OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfGroupDefinitions (OTF2_EventSizeEstimator *estimator, uint32_t numberOfGroupDefinitions)`

## E.11 otf2/OTF2\_EventSizeEstimator.h File Reference

---

*Sets the number of Group definitions used.*

- [OTF2\\_ErrorCode OTF2\\_EventSizeEstimator\\_SetNumberOfLocationDefinitions](#) ([OTF2\\_EventSizeEstimator](#) \*estimator, uint64\_t numberOfLocationDefinitions)

*Sets the number of Location definitions used.*

- [OTF2\\_ErrorCode OTF2\\_EventSizeEstimator\\_SetNumberOfMetricDefinitions](#) ([OTF2\\_EventSizeEstimator](#) \*estimator, uint32\_t numberOfMetricDefinitions)

*Sets the number of Metric definitions used.*

- [OTF2\\_ErrorCode OTF2\\_EventSizeEstimator\\_SetNumberOfParameterDefinitions](#) ([OTF2\\_EventSizeEstimator](#) \*estimator, uint32\_t numberOfParameterDefinitions)

*Sets the number of Parameter definitions used.*

- [OTF2\\_ErrorCode OTF2\\_EventSizeEstimator\\_SetNumberOfRegionDefinitions](#) ([OTF2\\_EventSizeEstimator](#) \*estimator, uint32\_t numberOfRegionDefinitions)

*Sets the number of Region definitions used.*

- [OTF2\\_ErrorCode OTF2\\_EventSizeEstimator\\_SetNumberOfRmaWinDefinitions](#) ([OTF2\\_EventSizeEstimator](#) \*estimator, uint32\_t numberOfRmaWinDefinitions)

*Sets the number of RmaWin definitions used.*

- [OTF2\\_ErrorCode OTF2\\_EventSizeEstimator\\_SetNumberOfStringDefinitions](#) ([OTF2\\_EventSizeEstimator](#) \*estimator, uint32\_t numberOfStringDefinitions)

*Sets the number of String definitions used.*

### E.11.1 Detailed Description

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

#### Source Template:

*templates/OTF2\_EventSizeEstimator.tmpl.h*

### E.11.2 Function Documentation

#### E.11.2.1 OTF2\_ErrorCode OTF2\_EventSizeEstimator\_Delete ( OTF2\_EventSizeEstimator \* estimator )

Deletes an OTF2\_EventSizeEstimator object.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.11.2.2**   `size_t OTF2_EventSizeEstimator_GetSizeOfAttributeList ( const  
OTF2_EventSizeEstimator * estimator, const OTF2_AttributeList *  
attributeList )`

Returns the size estimate for an attribute list.

The attribute list should be filled with the used types. The attribute references are taken from the number of attribute definitions and the values are the upper bounds for integral and floating point types, and the estimates for definition references.

### Parameters

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

### Returns

The estimated size.

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

Calculates the size estimate for the BufferFlush event.

### Parameters

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

### Since

Version 1.0

### Returns

The estimated size.

## E.11 of2/OTF2\_EventSizeEstimator.h File Reference

---

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

Calculates the size estimate for the Enter event.

#### Parameters

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

#### Since

Version 1.0

#### Returns

The estimated size.

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

Calculates the size estimate for the Leave event.

#### Parameters

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

#### Since

Version 1.0

#### Returns

The estimated size.

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

Calculates the size estimate for the MeasurementOnOff event.

#### Parameters

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

**Since**

Version 1.0

**Returns**

The estimated size.

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

Calculates the size estimate for the Metric event.

**Parameters**

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

**Since**

Version 1.0

**Returns**

The estimated size.

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

Calculates the size estimate for the MpiCollectiveBegin event.

**Parameters**

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

**Since**

Version 1.0

**Returns**

The estimated size.



## E.11 of2/OTF2\_EventSizeEstimator.h File Reference

---

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

Calculates the size estimate for the MpiCollectiveEnd event.

### Parameters

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

### Since

Version 1.0

### Returns

The estimated size.

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

Calculates the size estimate for the MpiIrecv event.

### Parameters

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

### Since

Version 1.0

### Returns

The estimated size.

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

Calculates the size estimate for the MpiIrecvRequest event.

### Parameters

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

**Since**

Version 1.0

**Returns**

The estimated size.

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

Calculates the size estimate for the MpilsendComplete event.

**Parameters**

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

**Since**

Version 1.0

**Returns**

The estimated size.

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

Calculates the size estimate for the Mpilsend event.

**Parameters**

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

**Since**

Version 1.0

**Returns**

The estimated size.

## E.11 of2/OTF2\_EventSizeEstimator.h File Reference

---

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

Calculates the size estimate for the MpiRecv event.

### Parameters

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

### Since

Version 1.0

### Returns

The estimated size.

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

Calculates the size estimate for the MpiRequestCancelled event.

### Parameters

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

### Since

Version 1.0

### Returns

The estimated size.

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

Calculates the size estimate for the MpiRequestTest event.

### Parameters

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

### Since

Version 1.0

### Returns

The estimated size.

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

Calculates the size estimate for the MpiSend event.

### Parameters

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

### Since

Version 1.0

### Returns

The estimated size.

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

Calculates the size estimate for the OmpAcquireLock event.

### Parameters

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

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

The estimated size.

## E.11 of2/OTF2\_EventSizeEstimator.h File Reference

---

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

Calculates the size estimate for the OmpFork event.

### Parameters

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

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

The estimated size.

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

Calculates the size estimate for the OmpJoin event.

### Parameters

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

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

The estimated size.

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

Calculates the size estimate for the OmpReleaseLock event.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

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

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

The estimated size.

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

Calculates the size estimate for the OmpTaskComplete event.

### Parameters

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

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

The estimated size.

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

Calculates the size estimate for the OmpTaskCreate event.

### Parameters

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

## E.11 of2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

The estimated size.

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

Calculates the size estimate for the OmpTaskSwitch event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

The estimated size.

**E.11.2.25**   `size_t OTF2_EventSizeEstimator_GetSizeOfParameterIntEvent (`  
                  `OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ParameterInt event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.0

### Returns

The estimated size.

**E.11.2.26** `size_t OTF2_EventSizeEstimator_GetSizeOfParameterStringEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ParameterString event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.0

### Returns

The estimated size.

**E.11.2.27** `size_t OTF2_EventSizeEstimator_GetSizeOfParameterUnsignedIntEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ParameterUnsignedInt event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.0

### Returns

The estimated size.

**E.11.2.28** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaAcquireLockEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaAcquireLock event.



## E.11 of2/OTF2\_EventSizeEstimator.h File Reference

---

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

#### E.11.2.29 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaAtomicEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaAtomic event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

#### E.11.2.30 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveBeginEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaCollectiveBegin event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.11.2.31** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaCollectiveEndEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaCollectiveEnd event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.32** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaGetEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaGet event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.33** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaGroupSyncEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaGroupSync event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

## E.11 oftf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.2

### Returns

The estimated size.

#### E.11.2.34 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteBlockingEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaOpCompleteBlocking event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

#### E.11.2.35 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteNonBlockingEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaOpCompleteNonBlocking event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.11.2.36** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpCompleteRemoteEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaOpCompleteRemote event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.37** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaOpTestEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaOpTest event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.38** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaPutEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaPut event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

## E.11 oftf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.39** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaReleaseLockEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaReleaseLock event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.40** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaRequestLockEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaRequestLock event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.11.2.41** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaSyncEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaSync event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.42** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaTryLockEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaTryLock event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.43** `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWaitChangeEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaWaitChangeEvent event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

## E.11 oftf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.2

### Returns

The estimated size.

#### E.11.2.44 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinCreateEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaWinCreate event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

#### E.11.2.45 `size_t OTF2_EventSizeEstimator_GetSizeOfRmaWinDestroyEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the RmaWinDestroy event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.46** `size_t OTF2_EventSizeEstimator_GetSizeOfThreadAcquireLockEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadAcquireLock event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.47** `size_t OTF2_EventSizeEstimator_GetSizeOfThreadBeginEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadBegin event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.3

### Returns

The estimated size.

**E.11.2.48** `size_t OTF2_EventSizeEstimator_GetSizeOfThreadCreateEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadCreate event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------



## E.11 oftf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.3

### Returns

The estimated size.

**E.11.2.49**    `size_t OTF2_EventSizeEstimator_GetSizeOfThreadEndEvent (`  
                  `OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadEnd event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.3

### Returns

The estimated size.

**E.11.2.50**    `size_t OTF2_EventSizeEstimator_GetSizeOfThreadForkEvent (`  
                  `OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadFork event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.11.2.51** `size_t OTF2_EventSizeEstimator_GetSizeOfThreadJoinEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadJoin event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.52** `size_t OTF2_EventSizeEstimator_GetSizeOfThreadReleaseLockEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadReleaseLock event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.53** `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCompleteEvent (`  
`OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadTaskComplete event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

## E.11 oftf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.2

### Returns

The estimated size.

#### E.11.2.54 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskCreateEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadTaskCreate event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

#### E.11.2.55 `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTaskSwitchEvent ( OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadTaskSwitch event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.11.2.56**   `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamBeginEvent (`  
                  `OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadTeamBegin event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.57**   `size_t OTF2_EventSizeEstimator_GetSizeOfThreadTeamEndEvent (`  
                  `OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadTeamEnd event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Since

Version 1.2

### Returns

The estimated size.

**E.11.2.58**   `size_t OTF2_EventSizeEstimator_GetSizeOfThreadWaitEvent (`  
                  `OTF2_EventSizeEstimator * estimator )`

Calculates the size estimate for the ThreadWait event.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

## E.11 otf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.3

### Returns

The estimated size.

**E.11.2.59** `size_t OTF2_EventSizeEstimator_GetSizeOfTimestamp ( OTF2_EventSizeEstimator * estimator )`

Returns the size for an timestamp record.

OTF2 does only store a timestamp, if it changed between two events.

### Parameters

<i>estimator</i>	Estimator object.
------------------	-------------------

### Returns

The estimated size.

**E.11.2.60** `OTF2_EventSizeEstimator* OTF2_EventSizeEstimator_New ( void )`

Creates a new OTF2\_EventSizeEstimator object.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.11.2.61** `OTF2_ErrorCode OTF2_EventSizeEstimator_SetNumberOfAttributeDefinitions ( OTF2_EventSizeEstimator * estimator, uint32_t numberOfAttributeDefinitions )`

Sets the number of Attribute definitions used.

Definition ids are considered to be in the range [0, numberOfAttributeDefinitions).

### Parameters

<i>estimator</i>	Estimator object.
<i>numberOfAttributeDefinitions</i>	The number of definitions.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.11.2.62** **OTF2\_ErrorCode** **OTF2\_EventSizeEstimator\_SetNumberOfCommDefinitions** (  
    **OTF2\_EventSizeEstimator** \* *estimator*, **uint32\_t** *numberOfCommDefinitions*  
)

Sets the number of Comm definitions used.

Definition ids are considered to be in the range [0, numberOfCommDefinitions).

**Parameters**

<i>estimator</i>	Estimator object.
<i>numberOfCommDefinitions</i>	The number of definitions.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.11.2.63** **OTF2\_ErrorCode** **OTF2\_EventSizeEstimator\_SetNumberOfGroupDefinitions** (  
    **OTF2\_EventSizeEstimator** \* *estimator*, **uint32\_t** *numberOfGroupDefinitions*  
)

Sets the number of Group definitions used.

Definition ids are considered to be in the range [0, numberOfGroupDefinitions).

**Parameters**

<i>estimator</i>	Estimator object.
<i>numberOfGroupDefinitions</i>	The number of definitions.

## E.11 otf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.11.2.64 OTF2\_ErrorCode OTF2\_EventSizeEstimator\_-SetNumberOfLocationDefinitions ( OTF2\_EventSizeEstimator \* *estimator*, uint64\_t *numberOfLocationDefinitions* )**

Sets the number of Location definitions used.

Definition ids are considered to be in the range [0, numberOfLocationDefinitions).

### Parameters

<i>estimator</i>	Estimator object.
<i>numberOfLocationDefinitions</i>	The number of definitions.

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.11.2.65 OTF2\_ErrorCode OTF2\_EventSizeEstimator\_SetNumberOfMetricDefinitions ( OTF2\_EventSizeEstimator \* *estimator*, uint32\_t *numberOfMetricDefinitions* )**

Sets the number of Metric definitions used.

Definition ids are considered to be in the range [0, numberOfMetricDefinitions).

### Parameters

<i>estimator</i>	Estimator object.
<i>numberOfMetricDefinitions</i>	The number of definitions.

## APPENDIX E. FILE DOCUMENTATION

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.11.2.66** **OTF2\_ErrorCode** **OTF2\_EventSizeEstimator\_-SetNumberOfParameterDefinitions** ( **OTF2\_EventSizeEstimator** \* *estimator*, uint32\_t *numberOfParameterDefinitions* )

Sets the number of Parameter definitions used.

Definition ids are considered to be in the range [0, numberOfParameterDefinitions).

### Parameters

<i>estimator</i>	Estimator object.
<i>numberOfParameterDefinitions</i>	The number of definitions.

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.11.2.67** **OTF2\_ErrorCode** **OTF2\_EventSizeEstimator\_SetNumberOfRegionDefinitions** ( **OTF2\_EventSizeEstimator** \* *estimator*, uint32\_t *numberOfRegionDefinitions* )

Sets the number of Region definitions used.

Definition ids are considered to be in the range [0, numberOfRegionDefinitions).

### Parameters

<i>estimator</i>	Estimator object.
<i>numberOfRegionDefinitions</i>	The number of definitions.



## E.11 otf2/OTF2\_EventSizeEstimator.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.11.2.68 OTF2\_ErrorCode OTF2\_EventSizeEstimator\_-SetNumberOfRmaWinDefinitions ( OTF2\_EventSizeEstimator \* estimator, uint32\_t numberOfRmaWinDefinitions )**

Sets the number of RmaWin definitions used.

Definition ids are considered to be in the range [0, numberOfRmaWinDefinitions).

### Parameters

<i>estimator</i>	Estimator object.
<i>numberOfRmaWinDefinitions</i>	The number of definitions.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.11.2.69 OTF2\_ErrorCode OTF2\_EventSizeEstimator\_SetNumberOfStringDefinitions ( OTF2\_EventSizeEstimator \* estimator, uint32\_t numberOfStringDefinitions )**

Sets the number of String definitions used.

Definition ids are considered to be in the range [0, numberOfStringDefinitions).

### Parameters

<i>estimator</i>	Estimator object.
<i>numberOfStringDefinitions</i>	The number of definitions.

**Since**

Version 1.0

**Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.12 otf2/OTF2\_EvtReader.h File Reference**

This is the local event reader, which reads events from one location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_EvtReaderCallbacks.h>
```

**Functions**

- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_ApplyClockOffsets](#) ([OTF2\\_EvtReader](#) \*reader, bool action)  
*Enable or disable applying of the clock offset to event timestamps read from this event reader.*
- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_ApplyMappingTables](#) ([OTF2\\_EvtReader](#) \*reader, bool action)  
*Enable or disable applying of the mapping tables to events read from this event reader.*
- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_GetLocationID](#) (const [OTF2\\_EvtReader](#) \*reader, [OTF2\\_LocationRef](#) \*location)  
*Return the location ID of the reading related location.*
- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_GetPos](#) ([OTF2\\_EvtReader](#) \*reader, uint64\_t \*position)  
*The following function can be used to get the position (number of the event in the stream) of last read event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_ReadEvents](#) ([OTF2\\_EvtReader](#) \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)  
*After callback registration, the local events could be read with the following function. Readn reads recordsToRead records. The reader indicates that it reached the end of the trace by just reading less records than requested.*

## E.12 otf2/OTF2\_EvtReader.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_ReadEventsBackward](#) ([OTF2\\_EvtReader](#) \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)  
*This functions reads recordsRead events backwards from the current position.*
- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_Seek](#) ([OTF2\\_EvtReader](#) \*reader, uint64\_t position)  
*Seek jumps to an event position.*
- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_SetCallbacks](#) ([OTF2\\_EvtReader](#) \*reader, const [OTF2\\_EvtReaderCallbacks](#) \*callbacks, void \*userData)  
*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*
- [OTF2\\_ErrorCode OTF2\\_EvtReader\\_TimeStampRewrite](#) ([OTF2\\_EvtReader](#) \*reader, [OTF2\\_TimeStamp](#) time)  
*The following function rewrites the timestamp from the event on the actual reading position if the buffer is in OTF2\_BUFFER\_MODIFY mode. It also modifies the timestamp for all other events in the same timestamp bundle. This function also has to keep track that not only the last timestamp, but all records equal to the last timestamp has to be modified. This is done by a position list, if there has no seek appeared before. In this case a position list can be easily generated because of that the reader has seen all related timestamps before. This not the case if there has a seek appeared before. In this case the related timestamp positions are generated by a linear search.*

### E.12.1 Detailed Description

This is the local event reader, which reads events from one location.

### E.12.2 Function Documentation

#### E.12.2.1 [OTF2\\_ErrorCode OTF2\\_EvtReader.ApplyClockOffsets](#) ( [OTF2\\_EvtReader](#) \* reader, bool action )

Enable or disable applying of the clock offset to event timestamps read from this event reader.

This setting has no effect if the events are read by an global event reader.

#### Parameters

<i>reader</i>	Reader object.
<i>action</i>	Truth value whether the clock offsets should be applied or not.

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.12.2.2** **OTF2\_ErrorCode** **OTF2\_EvtReader.ApplyMappingTables** (  
    **OTF2\_EvtReader** \* *reader*, **bool** *action* )

Enable or disable applying of the mapping tables to events read from this event reader.

This setting has no effect if the events are read by an global event reader.

**Parameters**

<i>reader</i>	Reader object.
<i>action</i>	Truth value whether the mappings should be applied or not.

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.12.2.3** **OTF2\_ErrorCode** **OTF2\_EvtReader.GetLocationID** ( **const**  
    **OTF2\_EvtReader** \* *reader*, **OTF2\_LocationRef** \* *location* )

Return the location ID of the reading related location.

**Parameters**

	<i>reader</i>	Reader object which reads the events from its buffer.
<i>out</i>	<i>location</i>	ID of the location.

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.12.2.4** **OTF2\_ErrorCode** **OTF2\_EvtReader.GetPos** ( **OTF2\_EvtReader** \* *reader*,  
    **uint64\_t** \* *position* )

The following function can be used to get the position (number of the event in the stream) of last read event.

**Parameters**

## E.12 otf2/OTF2\_EvtReader.h File Reference

---

	<i>reader</i>	Reader object which reads the events from its buffer.
out	<i>position</i>	Number of the event in the stream.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.12.2.5 OTF2\_ErrorCode OTF2\_EvtReader.ReadEvents ( OTF2\_EvtReader \* *reader*, uint64\_t *recordsToRead*, uint64\_t \* *recordsRead* )

After callback registration, the local events could be read with the following function. Readn reads *recordsToRead* records. The reader indicates that it reached the end of the trace by just reading less records than requested.

### Parameters

	<i>reader</i>	Reader object which reads the events from its buffer.
	<i>recordsToRead</i>	How many records can be read next.
out	<i>recordsRead</i>	Return how many records where really read.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.12.2.6 OTF2\_ErrorCode OTF2\_EvtReader.ReadEventsBackward ( OTF2\_EvtReader \* *reader*, uint64\_t *recordsToRead*, uint64\_t \* *recordsRead* )

This functions reads *recordsRead* events backwards from the current position.

### Parameters

	<i>reader</i>	Reader object which reads the events from its buffer.
	<i>recordsToRead</i>	How many records can be read next.
out	<i>recordsRead</i>	Return how many records where really read.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.12.2.7 OTF2\_ErrorCode OTF2\_EvtReader.Seek ( OTF2\_EvtReader \* *reader*,  
uint64\_t *position* )**

Seek jumps to an event position.

**Parameters**

<i>reader</i>	Reader object which reads the events from its buffer.
<i>position</i>	Number of the event, where the reader has to jump.

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.12.2.8 OTF2\_ErrorCode OTF2\_EvtReader.SetCallbacks ( OTF2\_EvtReader \*  
*reader*, const OTF2\_EvtReaderCallbacks \* *callbacks*, void \* *userData* )**

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

These callbacks are ignored, if the events are read by an global event reader.

**Parameters**

<i>reader</i>	Reader object which reads the events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#"><i>OTF2_EvtReaderCallbacks_New</i></a> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.12.2.9 OTF2\_ErrorCode OTF2\_EvtReader.TimestampRewrite ( OTF2\_EvtReader  
\* *reader*, OTF2\_TimeStamp *time* )**

The following function rewrites the timestamp from the event on the actual reading position if the buffer is in OTF2\_BUFFER\_MODIFY mode. It also modifies the timestamp for all other events in the same timestamp bundle. This function also has to keep track that not only the last timestamp, but all records equal to the last timestamp has to be modified. This is done by a position list, if there has no seek

### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

appeared before. In this case a position list can be easily generated because of that the reader has seen all related timestamps before. This not the case if there has a seek appeared before. In this case the related timestamp positions are generated by a linear search.

#### Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
<i>time</i>	New timestamp

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

This defines the callbacks for the event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

#### Typedefs

- typedef [\*OTF2\\_CallbackCode\*](#)(\* [\*OTF2\\_EvtReaderCallback\\_BufferFlush\*](#))([\*OTF2\\_LocationRef\*](#) location, [\*OTF2\\_TimeStamp\*](#) time, uint64\_t eventPosition, void \*userData, [\*OTF2\\_AttributeList\*](#) \*attributeList, [\*OTF2\\_TimeStamp\*](#) stopTime)

*Callback for the BufferFlush event record.*

- typedef [\*OTF2\\_CallbackCode\*](#)(\* [\*OTF2\\_EvtReaderCallback\\_Enter\*](#))([\*OTF2\\_LocationRef\*](#) location, [\*OTF2\\_TimeStamp\*](#) time, uint64\_t eventPosition, void \*userData, [\*OTF2\\_AttributeList\*](#) \*attributeList, [\*OTF2\\_RegionRef\*](#) region)

*Callback for the Enter event record.*

- typedef [\*OTF2\\_CallbackCode\*](#)(\* [\*OTF2\\_EvtReaderCallback\\_Leave\*](#))([\*OTF2\\_LocationRef\*](#) location, [\*OTF2\\_TimeStamp\*](#) time, uint64\_t eventPosition, void \*userData, [\*OTF2\\_AttributeList\*](#) \*attributeList, [\*OTF2\\_RegionRef\*](#) region)

*Callback for the Leave event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MeasurementOnOff)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_MeasurementMode measurementMode)

*Callback for the MeasurementOnOff event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_Metric)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_MetricRef metric, uint8\_t numberOfMetrics, const OTF2\_Type \*typeIDs, const OTF2\_MetricValue \*metricValues)

*Callback for the Metric event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiCollectiveBegin)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList)

*Callback for the MpiCollectiveBegin event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiCollectiveEnd)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CollectiveOp collectiveOp, OTF2\_CommRef communicator, uint32\_t root, uint64\_t sizeSent, uint64\_t sizeReceived)

*Callback for the MpiCollectiveEnd event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiIrecv)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t sender, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID)

*Callback for the MpiIrecv event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiIrecvRequest)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)

*Callback for the MpiIrecvRequest event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiIsend)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID)

*Callback for the MpiIsend event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiIsendComplete)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)

*Callback for the MpiIsendComplete event record.*



### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiRecv )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t sender, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)  
*Callback for the MpiRecv event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiRequestCancelled )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)  
*Callback for the MpiRequestCancelled event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiRequestTest )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)  
*Callback for the MpiRequestTest event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_MpiSend )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)  
*Callback for the MpiSend event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_OmpAcquireLock )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t lockID, uint32\_t acquisitionOrder)  
*Callback for the OmpAcquireLock event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_OmpFork )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t numberOfRequestedThreads)  
*Callback for the OmpFork event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_OmpJoin )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList)  
*Callback for the OmpJoin event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_OmpReleaseLock )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t lockID, uint32\_t acquisitionOrder)  
*Callback for the OmpReleaseLock event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_OmpTaskComplete )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)

*Callback for the OmpTaskComplete event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)`

*Callback for the OmpTaskCreate event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)`

*Callback for the OmpTaskSwitch event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, int64_t value)`

*Callback for the ParameterInt event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, OTF2_StringRef string)`

*Callback for the ParameterString event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, uint64_t value)`

*Callback for the ParameterUnsignedInt event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)`

*Callback for the RmaAcquireLock event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaAtomic)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)`

*Callback for the RmaAtomic event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)`

*Callback for the RmaCollectiveBegin event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp`

### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

collectiveOp, [OTF2\\_RmaSyncLevel](#) syncLevel, [OTF2\\_RmaWinRef](#) win, uint32\_t root, uint64\_t bytesSent, uint64\_t bytesReceived)

*Callback for the RmaCollectiveEnd event record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_EvtReaderCallback\\_RmaGet](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) time, uint64\_t eventPosition, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RmaWinRef](#) win, uint32\_t remote, uint64\_t bytes, uint64\_t matchingId)

*Callback for the RmaGet event record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_EvtReaderCallback\\_RmaGroupSync](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) time, uint64\_t eventPosition, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RmaSyncLevel](#) syncLevel, [OTF2\\_RmaWinRef](#) win, [OTF2\\_GroupRef](#) group)

*Callback for the RmaGroupSync event record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_EvtReaderCallback\\_RmaOpCompleteBlocking](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) time, uint64\_t eventPosition, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RmaWinRef](#) win, uint64\_t matchingId)

*Callback for the RmaOpCompleteBlocking event record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_EvtReaderCallback\\_RmaOpCompleteNonBlocking](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) time, uint64\_t eventPosition, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RmaWinRef](#) win, uint64\_t matchingId)

*Callback for the RmaOpCompleteNonBlocking event record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_EvtReaderCallback\\_RmaOpCompleteRemote](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) time, uint64\_t eventPosition, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RmaWinRef](#) win, uint64\_t matchingId)

*Callback for the RmaOpCompleteRemote event record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_EvtReaderCallback\\_RmaOpTest](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) time, uint64\_t eventPosition, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RmaWinRef](#) win, uint64\_t matchingId)

*Callback for the RmaOpTest event record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_EvtReaderCallback\\_RmaPut](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) time, uint64\_t eventPosition, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RmaWinRef](#) win, uint32\_t remote, uint64\_t bytes, uint64\_t matchingId)

*Callback for the RmaPut event record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_EvtReaderCallback\\_RmaReleaseLock](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) time, uint64\_t eventPosition, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RmaWinRef](#) win, uint32\_t remote, uint64\_t lockId)

*Callback for the RmaReleaseLock event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaRequestLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)`

*Callback for the RmaRequestLock event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaSync)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaSyncType syncType)`

*Callback for the RmaSync event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaTryLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)`

*Callback for the RmaTryLock event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaWaitChange)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win)`

*Callback for the RmaWaitChange event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaWinCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win)`

*Callback for the RmaWinCreate event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaWinDestroy)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win)`

*Callback for the RmaWinDestroy event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

*Callback for the ThreadAcquireLock event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

*Callback for the ThreadBegin event record.*

### E.13 otF2/OTF2\_EvtReaderCallbacks.h File Reference

---

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadCreate )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

*Callback for the ThreadCreate event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadEnd )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

*Callback for the ThreadEnd event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadFork )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)`

*Callback for the ThreadFork event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadJoin )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model)`

*Callback for the ThreadJoin event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadReleaseLock )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

*Callback for the ThreadReleaseLock event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadTaskComplete )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskComplete event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadTaskCreate )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskCreate event record.*

- typedef `OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadTaskSwitch )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskSwitch event record.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- typedef `OTF2_CallbackCode`(\* `OTF2_EvtReaderCallback_ThreadTeamBegin`)(`OTF2_LocationRef` location, `OTF2_TimeStamp` time, `uint64_t` eventPosition, void \*userData, `OTF2_AttributeList` \*attributeList, `OTF2_CommRef` threadTeam)  
*Callback for the ThreadTeamBegin event record.*
- typedef `OTF2_CallbackCode`(\* `OTF2_EvtReaderCallback_ThreadTeamEnd`)(`OTF2_LocationRef` location, `OTF2_TimeStamp` time, `uint64_t` eventPosition, void \*userData, `OTF2_AttributeList` \*attributeList, `OTF2_CommRef` threadTeam)  
*Callback for the ThreadTeamEnd event record.*
- typedef `OTF2_CallbackCode`(\* `OTF2_EvtReaderCallback_ThreadWait`)(`OTF2_LocationRef` location, `OTF2_TimeStamp` time, `uint64_t` eventPosition, void \*userData, `OTF2_AttributeList` \*attributeList, `OTF2_CommRef` threadContingent, `uint64_t` sequenceCount)  
*Callback for the ThreadWait event record.*
- typedef `OTF2_CallbackCode`(\* `OTF2_EvtReaderCallback_Unknown`)(`OTF2_LocationRef` location, `OTF2_TimeStamp` time, `uint64_t` eventPosition, void \*userData, `OTF2_AttributeList` \*attributeList)  
*Callback for an unknown event record.*
- typedef struct `OTF2_EvtReaderCallbacks_struct` `OTF2_EvtReaderCallbacks`  
*Opaque struct which holds all event record callbacks.*

### Functions

- void `OTF2_EvtReaderCallbacks_Clear` (`OTF2_EvtReaderCallbacks` \*evtReaderCallbacks)  
*Clears a struct for the event callbacks.*
- void `OTF2_EvtReaderCallbacks_Delete` (`OTF2_EvtReaderCallbacks` \*evtReaderCallbacks)  
*Deallocates a struct for the event callbacks.*
- `OTF2_EvtReaderCallbacks` \* `OTF2_EvtReaderCallbacks_New` (void)  
*Allocates a new struct for the event callbacks.*
- `OTF2_StatusCode` `OTF2_EvtReaderCallbacks_SetBufferFlushCallback` (`OTF2_EvtReaderCallbacks` \*evtReaderCallbacks, `OTF2_EvtReaderCallback_BufferFlush` bufferFlushCallback)  
*Registers the callback for the BufferFlush event.*
- `OTF2_StatusCode` `OTF2_EvtReaderCallbacks_SetEnterCallback` (`OTF2_EvtReaderCallbacks` \*evtReaderCallbacks, `OTF2_EvtReaderCallback_Enter` enterCallback)  
*Registers the callback for the Enter event.*

### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetLeaveCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_Leave leaveCallback\)](#)  
*Registers the callback for the Leave event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMeasurementOnOffCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_MeasurementOnOff measurementOnOffCallback\)](#)  
*Registers the callback for the MeasurementOnOff event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMetricCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_Metric metricCallback\)](#)  
*Registers the callback for the Metric event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMpiCollectiveBeginCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_MpiCollectiveBegin mpiCollectiveBeginCallback\)](#)  
*Registers the callback for the MpiCollectiveBegin event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMpiCollectiveEndCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_MpiCollectiveEnd mpiCollectiveEndCallback\)](#)  
*Registers the callback for the MpiCollectiveEnd event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMpiIrecvCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_MpiIrecv mpiIrecvCallback\)](#)  
*Registers the callback for the MpiIrecv event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMpiIrecvRequestCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_MpiIrecvRequest mpiIrecvRequestCallback\)](#)  
*Registers the callback for the MpiIrecvRequest event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMpiIsendCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_MpiIsend mpiIsendCallback\)](#)  
*Registers the callback for the MpiIsend event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMpiIsendCompleteCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_MpiIsendComplete mpiIsendCompleteCallback\)](#)  
*Registers the callback for the MpiIsendComplete event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetMpiRecvCallback \(OTF2\\_EvtReaderCallbacks \\*evtReaderCallbacks, OTF2\\_EvtReaderCallback\\_MpiRecv mpiRecvCallback\)](#)  
*Registers the callback for the MpiRecv event.*



- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetMpiRequestCancelledCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_MpiRequestCancelled](#) mpiRequestCancelledCallback)  
*Registers the callback for the MpiRequestCancelled event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetMpiRequestTestCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_MpiRequestTest](#) mpiRequestTestCallback)  
*Registers the callback for the MpiRequestTest event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetMpiSendCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_MpiSend](#) mpiSendCallback)  
*Registers the callback for the MpiSend event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetOmpAcquireLockCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_OmpAcquireLock](#) ompAcquireLockCallback)  
*Registers the callback for the OmpAcquireLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetOmpForkCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_OmpFork](#) ompForkCallback)  
*Registers the callback for the OmpFork event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetOmpJoinCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_OmpJoin](#) ompJoinCallback)  
*Registers the callback for the OmpJoin event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetOmpReleaseLockCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_OmpReleaseLock](#) ompReleaseLockCallback)  
*Registers the callback for the OmpReleaseLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetOmpTaskCompleteCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_OmpTaskComplete](#) ompTaskCompleteCallback)  
*Registers the callback for the OmpTaskComplete event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetOmpTaskCreateCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_OmpTaskCreate](#) ompTaskCreateCallback)  
*Registers the callback for the OmpTaskCreate event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetOmpTaskSwitchCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_OmpTaskSwitch](#) ompTaskSwitchCallback)  
*Registers the callback for the OmpTaskSwitch event.*



### E.13 oftf2/OTF2\_EvtReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetParameterIntCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ParameterInt](#) parameterIntCallback)  
*Registers the callback for the ParameterInt event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetParameterStringCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ParameterString](#) parameterStringCallback)  
*Registers the callback for the ParameterString event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetParameterUnsignedIntCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ParameterUnsignedInt](#) parameterUnsignedIntCallback)  
*Registers the callback for the ParameterUnsignedInt event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetRmaAcquireLockCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaAcquireLock](#) rmaAcquireLockCallback)  
*Registers the callback for the RmaAcquireLock event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetRmaAtomicCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaAtomic](#) rmaAtomicCallback)  
*Registers the callback for the RmaAtomic event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetRmaCollectiveBeginCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaCollectiveBegin](#) rmaCollectiveBeginCallback)  
*Registers the callback for the RmaCollectiveBegin event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetRmaCollectiveEndCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaCollectiveEnd](#) rmaCollectiveEndCallback)  
*Registers the callback for the RmaCollectiveEnd event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetRmaGetCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaGet](#) rmaGetCallback)  
*Registers the callback for the RmaGet event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetRmaGroupSyncCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaGroupSync](#) rmaGroupSyncCallback)  
*Registers the callback for the RmaGroupSync event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetRmaOpCompleteBlockingCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaOpCompleteBlocking](#) rmaOpCompleteBlockingCallback)  
*Registers the callback for the RmaOpCompleteBlocking event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaOpCompleteNonBlockingCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaOpCompleteNonBlocking](#) rmaOpCompleteNonBlockingCallback)  
*Registers the callback for the RmaOpCompleteNonBlocking event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaOpCompleteRemoteCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaOpCompleteRemote](#) rmaOpCompleteRemoteCallback)  
*Registers the callback for the RmaOpCompleteRemote event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaOpTestCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaOpTest](#) rmaOpTestCallback)  
*Registers the callback for the RmaOpTest event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaPutCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaPut](#) rmaPutCallback)  
*Registers the callback for the RmaPut event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaReleaseLockCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaReleaseLock](#) rmaReleaseLockCallback)  
*Registers the callback for the RmaReleaseLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaRequestLockCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaRequestLock](#) rmaRequestLockCallback)  
*Registers the callback for the RmaRequestLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaSyncCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaSync](#) rmaSyncCallback)  
*Registers the callback for the RmaSync event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaTryLockCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaTryLock](#) rmaTryLockCallback)  
*Registers the callback for the RmaTryLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaWaitChangeCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaWaitChange](#) rmaWaitChangeCallback)  
*Registers the callback for the RmaWaitChange event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetRmaWinCreateCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaWinCreate](#) rmaWinCreateCallback)  
*Registers the callback for the RmaWinCreate event.*

## E.13 oftf2/OTF2\_EvtReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetRmaWinDestroyCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_RmaWinDestroy](#) rmaWinDestroyCallback)  
*Registers the callback for the RmaWinDestroy event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadAcquireLockCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadAcquireLock](#) threadAcquireLockCallback)  
*Registers the callback for the ThreadAcquireLock event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadBeginCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadBegin](#) threadBeginCallback)  
*Registers the callback for the ThreadBegin event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadCreateCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadCreate](#) threadCreateCallback)  
*Registers the callback for the ThreadCreate event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadEndCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadEnd](#) threadEndCallback)  
*Registers the callback for the ThreadEnd event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadForkCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadFork](#) threadForkCallback)  
*Registers the callback for the ThreadFork event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadJoinCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadJoin](#) threadJoinCallback)  
*Registers the callback for the ThreadJoin event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadReleaseLockCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadReleaseLock](#) threadReleaseLockCallback)  
*Registers the callback for the ThreadReleaseLock event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadTaskCompleteCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadTaskComplete](#) threadTaskCompleteCallback)  
*Registers the callback for the ThreadTaskComplete event.*
- [OTF2\\_ErrorCode OTF2\\_EvtReaderCallbacks\\_SetThreadTaskCreateCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadTaskCreate](#) threadTaskCreateCallback)  
*Registers the callback for the ThreadTaskCreate event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetThreadTaskSwitchCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadTaskSwitch](#) threadTaskSwitchCallback)  
*Registers the callback for the ThreadTaskSwitch event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetThreadTeamBeginCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadTeamBegin](#) threadTeamBeginCallback)  
*Registers the callback for the ThreadTeamBegin event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetThreadTeamEndCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadTeamEnd](#) threadTeamEndCallback)  
*Registers the callback for the ThreadTeamEnd event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetThreadWaitCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_ThreadWait](#) threadWaitCallback)  
*Registers the callback for the ThreadWait event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtReaderCallbacks\\_SetUnknownCallback](#) ([OTF2\\_EvtReaderCallbacks](#) \*evtReaderCallbacks, [OTF2\\_EvtReaderCallback\\_Unknown](#) unknownCallback)  
*Registers the callback for the Unknown event.*

### E.13.1 Detailed Description

This defines the callbacks for the event reader.

#### Source Template:

*templates/OTF2\_EvtReaderCallbacks.tmpl.h*

### E.13.2 Typedef Documentation

**E.13.2.1** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_BufferFlush)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp stopTime)`

Callback for the BufferFlush event record.

This event signals that the internal buffer was flushed at the given time.

#### Parameters

<i>location</i>	The location where this event happened.
-----------------	---

### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>stopTime</i>	The time the buffer flush finished.

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.2** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_Enter)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

Callback for the Enter event record.

An enter record indicates that the program enters a code region.

#### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	Needs to be defined in a definition record References a <a href="#">Region</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_REGION</a> is available.

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.13.2.3** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_Leave)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

Callback for the Leave event record.

A leave record indicates that the program leaves a code region.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	Needs to be defined in a definition record References a <a href="#">Region</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_REGION</a> is available.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.4** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_MeasurementOnOff)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_MeasurementMode measurementMode)`

Callback for the MeasurementOnOff event record.

This event signals where the measurement system turned measurement on or off.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.

### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>measurementMode</i>	Is the measurement turned on ( <a href="#">OTF2_MEASUREMENT_ON</a> ) or off ( <a href="#">OTF2_MEASUREMENT_OFF</a> )?

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.5** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_Metric)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)`

Callback for the Metric event record.

A metric event is always stored at the location that recorded the metric. A metric event can reference a metric class or metric instance. Therefore, metric classes and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

#### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>metric</i>	Could be a metric class or a metric instance. References a <a href="#">MetricClass</a> , or a <a href="#">MetricInstance</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_METRIC</a> is available.
<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding <a href="#">MetricMember</a> definitions.
<i>metricValues</i>	List of metric values.

## Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.6** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList)`

Callback for the MpiCollectiveBegin event record.

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.).

## Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.

## Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.7** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_CollectiveOp collectiveOp, OTF2_CommRef  
communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)`

Callback for the MpiCollectiveEnd event record.

A MpiCollectiveEnd record marks the end of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.). It keeps the necessary information for this event:



## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.8** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_MpiIrecv)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`

Callback for the MpiIrecv event record.

A MpiIrecv record indicates that a MPI message was received (MPI\_IRECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

### Parameters

## APPENDIX E. FILE DOCUMENTATION

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.9** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, uint64_t requestID)`

Callback for the `MpiIrecvRequest` event record.

Signals the request of an receive, which can be completed later.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the requested receive

## E.13 oftf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.10** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
MpiIsend)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t  
msgLength, uint64_t requestID)`

Callback for the MpiIsend event record.

A MpiIsend record indicates that a MPI message send process was initiated (MPI\_ISEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <a href="#"><i>Comm</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_COMM</i></a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.11** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-  
MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, uint64_t requestID)`

Callback for the MpiIsendComplete event record.

Signals the completion of non-blocking send request.

#### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_- EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.12** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-  
MpiRecv)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t  
msgLength)`

Callback for the MpiRecv event record.

A MpiRecv record indicates that a MPI message was received (MPI\_RECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

#### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.

### E.13 oftf2/OTF2\_EvtReaderCallbacks.h File Reference

---

<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.13** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_MpiRequestCancelled)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

Callback for the `MpiRequestCancelled` event record.

This events appears if the program canceled a request.

#### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

#### Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.14** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-  
MpiRequestTest)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, uint64_t requestID)`

Callback for the MpiRequestTest event record.

This events appears if the program tests if a request has already completed but the test failed.

## Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_-EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

## Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.15** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-  
MpiSend)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t  
msgLength)`

Callback for the MpiSend event record.

A MpiSend record indicates that a MPI message send process was initiated (MPI\_SEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.16** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_OmpAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the `OmpAcquireLock` event record.

An `OmpAcquireLock` record marks that a thread acquires an OpenMP lock.

This event record is superseded by the [ThreadAcquireLock](#) event record and should not be used when the [ThreadAcquireLock](#) event record is in use.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.17** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
OmpFork)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
uint32_t numberOfRequestedThreads)`

Callback for the OmpFork event record.

An OmpFork record marks that an OpenMP Thread forks a thread team.

This event record is superseded by the [\*ThreadFork\*](#) event record and should not be used when the [\*ThreadFork\*](#) event record is in use.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>numberOfRequestedThreads</i>	Requested size of the team.

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).



## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.2.18** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
OmpJoin)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)`

Callback for the OmpJoin event record.

An OmpJoin record marks that a team of threads is joint and only the master thread continues execution.

This event record is superseded by the [ThreadJoin](#) event record and should not be used when the [ThreadJoin](#) event record is in use.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.19** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
OmpReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the OmpReleaseLock event record.

An OmpReleaseLock record marks that a thread releases an OpenMP lock.

This event record is superseded by the [ThreadReleaseLock](#) event record and should not be used when the [ThreadReleaseLock](#) event record is in use.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.

## APPENDIX E. FILE DOCUMENTATION

<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.20** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
OmpTaskComplete)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, uint64_t taskID)`

Callback for the OmpTaskComplete event record.

An OmpTaskComplete record indicates that the execution of an OpenMP task has finished.

This event record is superseded by the [ThreadTaskComplete](#) event record and should not be used when the [ThreadTaskComplete](#) event record is in use.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the completed task instance.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.21** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, uint64_t taskID)`

Callback for the OmpTaskCreate event record.

An OmpTaskCreate record marks that an OpenMP Task was/will be created in the current region.

This event record is superseded by the [\*ThreadTaskCreate\*](#) event record and should not be used when the [\*ThreadTaskCreate\*](#) event record is in use.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the newly created task instance.

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.22** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, uint64_t taskID)`

Callback for the OmpTaskSwitch event record.

---

## APPENDIX E. FILE DOCUMENTATION

---

An `OmpTaskSwitch` record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

This event record is superseded by the [ThreadTaskSwitch](#) event record and should not be used when the [ThreadTaskSwitch](#) event record is in use.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the now active task instance.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.23** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
OTF2_ParameterRef parameter, int64_t value)`

Callback for the `ParameterInt` event record.

A `ParameterInt` record marks that in the current region, the specified integer parameter has the specified value.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.24** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_ParameterRef parameter, OTF2_StringRef string)`

Callback for the ParameterString event record.

A ParameterString record marks that in the current region, the specified string parameter has the specified value.

#### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>string</i>	Value: Handle of a string definition References a <a href="#">String</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_STRING</a> is available.

#### Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.25** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ -  
ParameterUnsignedInt)(OTF2_LocationRef location,  
OTF2_TimeStamp time, uint64_t eventPosition, void *userData,  
OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter,  
uint64_t value)`

Callback for the ParameterUnsignedInt event record.

A ParameterUnsignedInt record marks that in the current region, the specified unsigned integer parameter has the specified value.

## Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	Parameter ID. References a <a href="#"><i>Parameter</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_PARAMETER</i></a> is available.
<i>value</i>	Value of the recorded parameter.

## Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.26** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ -  
RmaAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId,  
OTF2_LockType lockType)`

Callback for the RmaAcquireLock event record.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

An RmaAcquireLock record denotes the time a lock was acquired by the process.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.27** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaAtomic)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type,  
uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)`

Callback for the RmaAtomic event record.

An RmaAtomic record denotes the time a atomic operation was issued.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

## APPENDIX E. FILE DOCUMENTATION

---

<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the target process.
<i>type</i>	Type of atomic operation.
<i>bytesSent</i>	Bytes sent to target.
<i>bytesReceived</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.28** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaCollectiveBegin)(OTF2_LocationRef location,  
OTF2_TimeStamp time, uint64_t eventPosition, void *userData,  
OTF2_AttributeList *attributeList)`

Callback for the RmaCollectiveBegin event record.

An RmaCollectiveBegin record denotes the beginning of a collective RMA operation.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).



## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.2.29** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel  
syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t  
bytesReceived)`

Callback for the RmaCollectiveEnd event record.

"An RmaCollectiveEnd record denotes the end of a collective RMA operation.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<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 <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>root</i>	Root process for this operation.
<i>bytesSent</i>	Bytes sent in operation.
<i>bytesReceived</i>	Bytes receives in operation.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.30** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaGet)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)`

Callback for the RmaGet event record.

---

## APPENDIX E. FILE DOCUMENTATION

---

An RmaGet record denotes the time a get operation was issued.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.31** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaGroupSync)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win,  
OTF2_GroupRef group)`

Callback for the RmaGroupSync event record.

An RmaGroupSync record denotes the synchronization with a subgroup of processes on a window.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .

## E.13 oftf2/OTF2\_EvtReaderCallbacks.h File Reference

---

<i>attributeList</i>	Additional attributes for this event.
<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>group</i>	Group of remote processes involved in synchronization. References a <a href="#">Group</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_GROUP</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.32** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaOpCompleteBlocking)(OTF2_LocationRef location,  
OTF2_TimeStamp time, uint64_t eventPosition, void *userData,  
OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t  
matchingId)`

Callback for the RmaOpCompleteBlocking event record.

An RmaOpCompleteBlocking record denotes the local completion of a blocking RMA operation.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

**Since**

Version 1.2

**Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.33** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaOpCompleteNonBlocking)(OTF2_LocationRef location,  
OTF2_TimeStamp time, uint64_t eventPosition, void *userData,  
OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t  
matchingId)`

Callback for the RmaOpCompleteNonBlocking event record.

An RmaOpCompleteNonBlocking record denotes the local completion of a non-blocking RMA operation.

**Parameters**

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

**Since**

Version 1.2

**Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.2.34** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-  
RmaOpCompleteRemote)(OTF2_LocationRef location,  
OTF2_TimeStamp time, uint64_t eventPosition, void *userData,  
OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t  
matchingId)`

Callback for the RmaOpCompleteRemote event record.

An RmaOpCompleteRemote record denotes the local completion of an RMA operation.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.35** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-  
RmaOpTest)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win, uint64_t matchingId)`

Callback for the RmaOpTest event record.

An RmaOpTest record denotes that a non-blocking RMA operation has been tested for completion unsuccessfully.

### Parameters

---

## APPENDIX E. FILE DOCUMENTATION

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.36** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaPut)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)`

Callback for the RmaPut event record.

An RmaPut record denotes the time a put operation was issued.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes sent to target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.37** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)`

Callback for the RmaReleaseLock event record.

An RmaReleaseLock record denotes the time the lock was released.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock released, if multiple locks are defined on a window.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.38** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaRequestLock)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId,  
OTF2_LockType lockType)`

Callback for the RmaRequestLock event record.

An RmaRequestLock record denotes the time a lock was requested and with it the earliest time it could have been granted. It is used to mark (possibly) non-blocking lock request, as defined by the MPI standard.

#### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.39** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaSync)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaSyncType  
syncType)`

Callback for the RmaSync event record.

An RmaSync record denotes the direct synchronization with a possibly remote process.



## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>syncType</i>	Type of synchronization.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.40** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaTryLock)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType  
lockType)`

Callback for the RmaTryLock event record.

An RmaTryLock record denotes the time of an unsuccessful attempt to acquire the lock.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.41** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaWaitChange)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win)`

Callback for the RmaWaitChange event record.

An RmaWaitChange record denotes the change of a window that was waited for.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> is available.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.2.42** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaWinCreate)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win)`

Callback for the RmaWinCreate event record.

An RmaWinCreate record denotes the creation of an RMA window.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window created. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.43** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaWinDestroy)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win)`

Callback for the RmaWinDestroy event record.

An RmaWinDestroy record denotes the destruction of an RMA window.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.

## APPENDIX E. FILE DOCUMENTATION

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_-EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window destructed. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_-MAPPING_RMA_WIN</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.44** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
ThreadAcquireLock)(OTF2_LocationRef location,  
OTF2_TimeStamp time, uint64_t eventPosition, void *userData,  
OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t  
lockID, uint32_t acquisitionOrder)`

Callback for the ThreadAcquireLock event record.

An ThreadAcquireLock record marks that a thread acquires an lock.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_-EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.45** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - ThreadBegin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

Callback for the ThreadBegin event record.

Marks the begin of a thread created by another thread.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#"><i>Comm</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_COMM</i></a> is available.
<i>sequenceCount</i>	A threadContingent unique number. The corresponding <a href="#"><i>ThreadCreate</i></a> event does have the same number.

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.46** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - ThreadCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

Callback for the ThreadCreate event record.

The location created successfully a new thread.

## APPENDIX E. FILE DOCUMENTATION

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>sequenceCount</i>	A <code>threadContingent</code> unique number. The corresponding <a href="#">Thread-Begin</a> event does have the same number.

### Since

Version 1.3

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.47** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - ThreadEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

Callback for the ThreadEnd event record.

Marks the end of a thread.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>sequenceCount</i>	A <code>threadContingent</code> unique number. The corresponding <a href="#">Thread-Wait</a> event does have the same number. <a href="#">OTF2_UNDEFINED_UINT64</a> in case no corresponding <a href="#">ThreadWait</a> event exists.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.48** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - ThreadFork)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)`

Callback for the ThreadFork event record.

An ThreadFork record marks that an thread forks a thread team.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>numberOfRequestedThreads</i>	Requested size of the team.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.13.2.49** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - ThreadJoin)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model)`

Callback for the ThreadJoin event record.

An ThreadJoin record marks that a team of threads is joint and only the master thread continues execution.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.50** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - ThreadReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the ThreadReleaseLock event record.

An ThreadReleaseLock record marks that a thread releases an lock.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.



### E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.51** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
ThreadTaskComplete)(OTF2_LocationRef location,  
OTF2_TimeStamp time, uint64_t eventPosition, void *userData,  
OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam,  
uint32_t creatingThread, uint32_t generationNumber)`

Callback for the ThreadTaskComplete event record.

An ThreadTaskComplete record indicates that the execution of an OpenMP task has finished.

#### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

**Since**

Version 1.2

**Returns**

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.52** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - ThreadTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

Callback for the ThreadTaskCreate event record.

An ThreadTaskCreate record marks that an task in was/will be created and will be processed by the specified thread team.

**Parameters**

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#"><i>Comm</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_COMM</i></a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

**Since**

Version 1.2

**Returns**

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.2.53** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
ThreadTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t  
generationNumber)`

Callback for the ThreadTaskSwitch event record.

An ThreadTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.54** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
ThreadTeamBegin)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_CommRef threadTeam)`

Callback for the ThreadTeamBegin event record.

The current location enters the specified thread team.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_-EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.13.2.55** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
ThreadTeamEnd)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList, OTF2_CommRef threadTeam)`

Callback for the ThreadTeamEnd event record.

The current location leaves the specified thread team.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_-EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.56** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - ThreadWait)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

Callback for the ThreadWait event record.

The location waits for the completion of another thread.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterEvtCallbacks</i></a> or <a href="#"><i>OTF2_EvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#"><i>Comm</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_COMM</i></a> is available.
<i>sequenceCount</i>	A threadContingent unique number. The corresponding <a href="#"><i>Thread-End</i></a> event does have the same number.

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.13.2.57** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_ - Unknown)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown event record.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

### E.13.3 Function Documentation

**E.13.3.1** `void OTF2_EvtReaderCallbacks_Clear ( OTF2_EvtReaderCallbacks *  
                  evtReaderCallbacks )`

Clears a struct for the event callbacks.

#### Parameters

<i>evtReader-Callbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_EvtReaderCallbacks_New</a> .
----------------------------	--

**E.13.3.2** `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 <a href="#">OTF2_EvtReaderCallbacks_New</a> .
----------------------------	--

**E.13.3.3** `OTF2_EvtReaderCallbacks* OTF2_EvtReaderCallbacks_New ( void )`

Allocates a new struct for the event callbacks.

### Returns

A newly allocated struct of type [OTF2\\_EvtReaderCallbacks](#).

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.3.4** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetBufferFlushCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_BufferFlush** *bufferFlushCallback* )

Registers the callback for the BufferFlush event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>buffer-FlushCall-back</i>	Function which should be called for all BufferFlush events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.5** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetEnterCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_Enter** *enterCallback* )

Registers the callback for the Enter event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>enterCall-back</i>	Function which should be called for all Enter events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks`

argument

**E.13.3.6** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetLeaveCallback**  
**( OTF2\_EvtReaderCallbacks \* *evtReaderCallbacks*,**  
**OTF2\_EvtReaderCallback\_Leave *leaveCallback* )**

Registers the callback for the Leave event.

**Parameters**

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>leaveCall-back</i>	Function which should be called for all Leave events.

**Since**

Version 1.0

**Returns**

***OTF2\_SUCCESS*** if successful  
***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks`  
argument

**E.13.3.7** **OTF2\_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>measurementOnOff-Callback</i>	Function which should be called for all MeasurementOnOff events.

**Since**

Version 1.0



## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.8** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetMetricCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_Metric** *metricCallback* )

Registers the callback for the Metric event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.9** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetMpiCollectiveBeginCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*, **OTF2\_EvtReaderCallback\_MpiCollectiveBegin** *mpiCollectiveBeginCallback* )

Registers the callback for the MpiCollectiveBegin event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin events.

**Since**

Version 1.0

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.10** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetMpiCollectiveEndCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*, **OTF2\_-**  
**EvtReaderCallback\_MpiCollectiveEnd** *mpiCollectiveEndCallback*  
)

Registers the callback for the `MpiCollectiveEnd` event.

**Parameters**

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEnd-Callback</i>	Function which should be called for all <code>MpiCollectiveEnd</code> events.

**Since**

Version 1.0

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.11** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetMpiIrecvCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_MpiIrecv** *mpilrecvCallback* )

Registers the callback for the `MpiIrecv` event.

**Parameters**

---

### E.13 oftf2/OTF2\_EvtReaderCallbacks.h File Reference

---

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all MpiIrecv events.

#### Since

Version 1.0

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.12** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetMpiIrecvRequestCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
OTF2\_EvtReaderCallback\_MpiIrecvRequest *mpiIrecvRequestCallback*  
)

Registers the callback for the MpiIrecvRequest event.

#### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all MpiIrecvRequest events.

#### Since

Version 1.0

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX E. FILE DOCUMENTATION

---

**E.13.3.13** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetMpiIsendCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_MpiIsend** *mpiIsendCallback* )

Registers the callback for the MpiIsend event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiIsend-Callback</i>	Function which should be called for all MpiIsend events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.13.3.14** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetMpiIsendCompleteCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_MpiIsendComplete**  
*mpiIsendCompleteCallback* )

Registers the callback for the MpiIsendComplete event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>mpiIsend-Complete-Callback</i>	Function which should be called for all MpiIsendComplete events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.15** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetMpiRecvCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_MpiRecv** *mpiRecvCallback* )

Registers the callback for the `MpiRecv` event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecvCallback</i>	Function which should be called for all <code>MpiRecv</code> events.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.16** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetMpiRequestCancelledCallback** ( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*, **OTF2\_EvtReaderCallback\_MpiRequestCancelled** *mpiRequestCancelledCallback* )

Registers the callback for the `MpiRequestCancelled` event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRequestCancelledCallback</i>	Function which should be called for all <code>MpiRequestCancelled</code> events.

**Since**

Version 1.0

**Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.17** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetMpiRequestTestCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_MpiRequestTest** *mpiRequestTestCallback* )

Registers the callback for the `MpiRequestTest` event.

**Parameters**

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRequestTestCallback</i>	Function which should be called for all <code>MpiRequestTest</code> events.

**Since**

Version 1.0

**Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.18** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetMpiSendCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_MpiSend** *mpiSendCallback* )

Registers the callback for the `MpiSend` event.

**Parameters**

<i>evtReaderCallbacks</i>	Struct for all callbacks.
---------------------------	---------------------------

## E.13 oftf2/OTF2\_EvtReaderCallbacks.h File Reference

---

<i>mpiSend-Callback</i>	Function which should be called for all MpiSend events.
-------------------------	---

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.19 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks.SetOmpAcquireLockCallback**  
( OTF2\_EvtReaderCallbacks \* *evtReaderCallbacks*, OTF2\_-  
EvtReaderCallback\_OmpAcquireLock *ompAcquireLockCallback*  
)

Registers the callback for the OmpAcquireLock event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>ompAc-quireLock-Callback</i>	Function which should be called for all OmpAcquireLock events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.20 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks.SetOmpForkCallback**  
( OTF2\_EvtReaderCallbacks \* *evtReaderCallbacks*,  
OTF2\_EvtReaderCallback\_OmpFork *ompForkCallback* )

Registers the callback for the OmpFork event.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>ompForkCallback</i>	Function which should be called for all OmpFork events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.21 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks.SetOmpJoinCallback**  
( **OTF2\_EvtReaderCallbacks \* *evtReaderCallbacks*,**  
**OTF2\_EvtReaderCallback\_OmpJoin *ompJoinCallback*** )

Registers the callback for the OmpJoin event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>ompJoinCallback</i>	Function which should be called for all OmpJoin events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument



## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.3.22** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetOmpReleaseLockCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_OmpReleaseLock** *ompReleaseLockCallback*  
)

Registers the callback for the OmpReleaseLock event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>ompReleaseLock-Callback</i>	Function which should be called for all OmpReleaseLock events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.13.3.23** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetOmpTaskCompleteCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_OmpTaskComplete**  
*ompTaskCompleteCallback* )

Registers the callback for the OmpTaskComplete event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>ompTaskCompleteCallback</i>	Function which should be called for all OmpTaskComplete events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.24** **OTF2\_StatusCode** **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-TaskCreate-Callback</i>	Function which should be called for all OmpTaskCreate events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.25** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetOmpTaskSwitchCallback**  
 ( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_OmpTaskSwitch** *ompTaskSwitchCallback* )

Registers the callback for the OmpTaskSwitch event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>omp-TaskSwitch-Callback</i>	Function which should be called for all OmpTaskSwitch events.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.26** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetParameterIntCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ParameterInt** *parameterIntCallback* )

Registers the callback for the ParameterInt event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterIntCallback</i>	Function which should be called for all ParameterInt events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.27** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetParameterStringCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ParameterString** *parameterStringCallback* )

Registers the callback for the ParameterString event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
---------------------------	---------------------------

## APPENDIX E. FILE DOCUMENTATION

---

<i>parameter-StringCallback</i>	Function which should be called for all ParameterString events.
---------------------------------	---

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.28 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_-SetParameterUnsignedIntCallback ( OTF2\_EvtReaderCallbacks \* *evtReaderCallbacks*, OTF2\_EvtReaderCallback\_ -ParameterUnsignedInt *parameterUnsignedIntCallback* )**

Registers the callback for the ParameterUnsignedInt event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt events.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.3.29** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaAcquireLockCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaAcquireLock** *rmaAcquireLockCallback*  
)

Registers the callback for the RmaAcquireLock event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaAcquireLock-Callback</i>	Function which should be called for all RmaAcquireLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.13.3.30** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaAtomicCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaAtomic** *rmaAtomicCallback* )

Registers the callback for the RmaAtomic event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaAtomic-Callback</i>	Function which should be called for all RmaAtomic events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

---

## APPENDIX E. FILE DOCUMENTATION

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.31** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaCollectiveBeginCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaCollectiveBegin**  
*rmaCollectiveBeginCallback* )

Registers the callback for the RmaCollectiveBegin event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaCollectiveBegin-Callback</i>	Function which should be called for all RmaCollectiveBegin events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.32** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaCollectiveEndCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaCollectiveEnd**  
*rmaCollectiveEndCallback* )

Registers the callback for the RmaCollectiveEnd event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaCollectiveEnd-Callback</i>	Function which should be called for all RmaCollectiveEnd events.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid `defReaderCallbacks` argument

**E.13.33** `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaGetCallback`  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks`,  
`OTF2_EvtReaderCallback_RmaGet rmaGetCallback` )

Registers the callback for the RmaGet event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaGet-Callback</i>	Function which should be called for all RmaGet events.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid `defReaderCallbacks` argument

**E.13.34** `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback`  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks`,  
`OTF2_EvtReaderCallback_RmaGroupSync rmaGroupSyncCallback` )

Registers the callback for the RmaGroupSync event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
----------------------------	---------------------------

## APPENDIX E. FILE DOCUMENTATION

---

<i>rmaGroup-SyncCallback</i>	Function which should be called for all RmaGroupSync events.
------------------------------	--

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

```
E.13.335  OTF2_ErrorCode OTF2_EvtReaderCallbacks_-  
          SetRmaOpCompleteBlockingCallback ( OTF2_EvtReaderCallbacks  
          * evtReaderCallbacks, OTF2_EvtReaderCallback_-  
          RmaOpCompleteBlocking rmaOpCompleteBlockingCallback  
          )
```

Registers the callback for the RmaOpCompleteBlocking event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpCompleteBlockingCallback</i>	Function which should be called for all RmaOpCompleteBlocking events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument



## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.3.36** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_**-  
**SetRmaOpCompleteNonBlockingCallback** ( **OTF2\_EvtReaderCallbacks**  
**\* *evtReaderCallbacks***, **OTF2\_EvtReaderCallback\_**-  
**RmaOpCompleteNonBlocking** ***rmaOpCompleteNonBlockingCallback***  
**)**

Registers the callback for the RmaOpCompleteNonBlocking event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaOp-CompleteNon-Blocking-Callback</i>	Function which should be called for all RmaOpCompleteNonBlocking events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.37** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_**-  
**SetRmaOpCompleteRemoteCallback** ( **OTF2\_EvtReaderCallbacks**  
**\* *evtReaderCallbacks***, **OTF2\_EvtReaderCallback\_**-  
**RmaOpCompleteRemote** ***rmaOpCompleteRemoteCallback***  
**)**

Registers the callback for the RmaOpCompleteRemote event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaOp-CompleteR-remoteCall-back</i>	Function which should be called for all RmaOpCompleteRemote events.

## APPENDIX E. FILE DOCUMENTATION

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.38** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaOpTestCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaOpTest** *rmaOpTestCallback* )

Registers the callback for the RmaOpTest event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpTestCallback</i>	Function which should be called for all RmaOpTest events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.39** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaPutCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaPut** *rmaPutCallback* )

Registers the callback for the RmaPut event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaPutCallback</i>	Function which should be called for all RmaPut events.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

```
E.13.3.40  OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaReleaseLockCallback
            ( OTF2_EvtReaderCallbacks * evtReaderCallbacks,
              OTF2_EvtReaderCallback_RmaReleaseLock rmaReleaseLockCallback
            )
```

Registers the callback for the RmaReleaseLock event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaReleaseLockCallback</i>	Function which should be called for all RmaReleaseLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

```
E.13.3.41  OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetRmaRequestLockCallback
            ( OTF2_EvtReaderCallbacks * evtReaderCallbacks,
              OTF2_EvtReaderCallback_RmaRequestLock rmaRequestLockCallback
            )
```

Registers the callback for the RmaRequestLock event.

### Parameters

## APPENDIX E. FILE DOCUMENTATION

---

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaRequestLockCallback</i>	Function which should be called for all RmaRequestLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.42** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetRmaSyncCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaSync** *rmaSyncCallback* )

Registers the callback for the RmaSync event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaSyncCallback</i>	Function which should be called for all RmaSync events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.3.43** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaTryLockCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaTryLock** *rmaTryLockCallback* )

Registers the callback for the RmaTryLock event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaTry-LockCall-back</i>	Function which should be called for all RmaTryLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.13.3.44** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaWaitChangeCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaWaitChange** *rmaWaitChangeCallback* )

Registers the callback for the RmaWaitChange event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaWait-Change-Callback</i>	Function which should be called for all RmaWaitChange events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

---

## APPENDIX E. FILE DOCUMENTATION

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.45** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaWinCreateCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaWinCreate** *rmaWinCreateCallback* )

Registers the callback for the RmaWinCreate event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaWin-CreateCall-back</i>	Function which should be called for all RmaWinCreate events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.46** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetRmaWinDestroyCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_RmaWinDestroy** *rmaWinDestroyCallback* )

Registers the callback for the RmaWinDestroy event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>rmaWinDe-destroyCall-back</i>	Function which should be called for all RmaWinDestroy events.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.47** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetThreadAcquireLockCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadAcquireLock**  
*threadAcquireLockCallback* )

Registers the callback for the ThreadAcquireLock event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadAcquireLockCallback</i>	Function which should be called for all ThreadAcquireLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.48** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks\_SetThreadBeginCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadBegin** *threadBeginCallback* )

Registers the callback for the ThreadBegin event.

### Parameters

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadBeginCallback</i>	Function which should be called for all ThreadBegin events.

### Since

Version 1.3

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.49** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetThreadCreateCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadCreate** *threadCreateCallback* )

Registers the callback for the ThreadCreate event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadCreateCallback</i>	Function which should be called for all ThreadCreate events.

### Since

Version 1.3

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.50** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetThreadEndCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadEnd** *threadEndCallback* )

Registers the callback for the ThreadEnd event.



## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadEndCallback</i>	Function which should be called for all ThreadEnd events.

### Since

Version 1.3

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.51** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetThreadForkCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadFork** *threadForkCallback* )

Registers the callback for the ThreadFork event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadForkCallback</i>	Function which should be called for all ThreadFork events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.52** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetThreadJoinCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadJoin** *threadJoinCallback* )

Registers the callback for the ThreadJoin event.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadJoinCallback</i>	Function which should be called for all ThreadJoin events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.13.3.53** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetThreadReleaseLockCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadReleaseLock**  
*threadReleaseLockCallback* )

Registers the callback for the ThreadReleaseLock event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadReleaseLockCallback</i>	Function which should be called for all ThreadReleaseLock events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

**E.13.3.54** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.**  
**SetThreadTaskCompleteCallback** ( **OTF2\_EvtReaderCallbacks** \*  
*evtReaderCallbacks*, **OTF2\_EvtReaderCallback\_ThreadTaskComplete**  
*threadTaskCompleteCallback* )

Registers the callback for the ThreadTaskComplete event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TaskCompleteCallback</i>	Function which should be called for all ThreadTaskComplete events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.13.3.55** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.**  
**SetThreadTaskCreateCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadTaskCreate**  
*threadTaskCreateCallback* )

Registers the callback for the ThreadTaskCreate event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TaskCreate-Callback</i>	Function which should be called for all ThreadTaskCreate events.

### Since

Version 1.2

## APPENDIX E. FILE DOCUMENTATION

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.56** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetThreadTaskSwitchCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadTaskSwitch**  
*threadTaskSwitchCallback* )

Registers the callback for the ThreadTaskSwitch event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TaskSwitch-Callback</i>	Function which should be called for all ThreadTaskSwitch events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.57** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetThreadTeamBeginCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadTeamBegin**  
*threadTeamBeginCallback* )

Registers the callback for the ThreadTeamBegin event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TeamBegin-Callback</i>	Function which should be called for all ThreadTeamBegin events.

## E.13 otf2/OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.58** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetThreadTeamEndCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadTeamEnd** *threadTeamEndCallback* )

Registers the callback for the ThreadTeamEnd event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTeamEndCallback</i>	Function which should be called for all ThreadTeamEnd events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.59** **OTF2\_ErrorCode** **OTF2\_EvtReaderCallbacks.SetThreadWaitCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_ThreadWait** *threadWaitCallback* )

Registers the callback for the ThreadWait event.

### Parameters

<i>evtReaderCallbacks</i>	Struct for all callbacks.
---------------------------	---------------------------

## APPENDIX E. FILE DOCUMENTATION

---

<i>threadWait-Callback</i>	Function which should be called for all ThreadWait events.
----------------------------	--

### Since

Version 1.3

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.13.3.60** **OTF2\_StatusCode** **OTF2\_EvtReaderCallbacks.SetUnknownCallback**  
( **OTF2\_EvtReaderCallbacks** \* *evtReaderCallbacks*,  
**OTF2\_EvtReaderCallback\_Unknown** *unknownCallback* )

Registers the callback for the Unknown event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	Function which should be called for all unknown events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## E.14 otf2/OTF2\_EvtWriter.h File Reference

This lowest user-visible layer provides write routines to write event data of a single location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_AttributeList.h>
```

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

### Typedefs

- typedef struct OTF2\_EvtWriter\_struct [OTF2\\_EvtWriter](#)  
*Keeps all necessary information about the event writer. See OTF2\_EvtWriter\_struct for detailed information.*

### Functions

- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_BufferFlush](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_TimeStamp](#) stopTime)  
*Records an BufferFlush event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_ClearRewindPoint](#) ([OTF2\\_EvtWriter](#) \*writer, [uint32\\_t](#) rewindId)  
*Please give me a documantation.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_Enter](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RegionRef](#) region)  
*Records an Enter event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_GetLocationID](#) (const [OTF2\\_EvtWriter](#) \*writer, [OTF2\\_LocationRef](#) \*locationID)  
*Function to get the location ID of a writer object.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_GetNumberOfEvents](#) ([OTF2\\_EvtWriter](#) \*writer, [uint64\\_t](#) \*numberOfEvents)  
*Get the number of events.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_GetUserData](#) (const [OTF2\\_EvtWriter](#) \*writer, void \*\*userData)  
*Function to get the location of a writer object.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_Leave](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RegionRef](#) region)  
*Records an Leave event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_MeasurementOnOff](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_MeasurementMode](#) measurementMode)  
*Records an MeasurementOnOff event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_Metric](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_MetricRef](#) metric, [uint8\\_t](#) numberOfMetrics, const [OTF2\\_Type](#) \*typeIDs, const [OTF2\\_MetricValue](#) \*metricValues)  
*Records an Metric event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiCollectiveBegin](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time)  
*Records an MpiCollectiveBegin event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiCollectiveEnd](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_CollectiveOp](#) collectiveOp, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) root, [uint64\\_t](#) sizeSent, [uint64\\_t](#) sizeReceived)  
*Records an MpiCollectiveEnd event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiIrecv](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [uint32\\_t](#) sender, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength, [uint64\\_t](#) requestID)  
*Records an MpiIrecv event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiIrecvRequest](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [uint64\\_t](#) requestID)  
*Records an MpiIrecvRequest event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiIsend](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [uint32\\_t](#) receiver, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength, [uint64\\_t](#) requestID)  
*Records an MpiIsend event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiIsendComplete](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [uint64\\_t](#) requestID)  
*Records an MpiIsendComplete event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiRecv](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [uint32\\_t](#) sender, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength)  
*Records an MpiRecv event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiRequestCancelled](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [uint64\\_t](#) requestID)  
*Records an MpiRequestCancelled event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiRequestTest](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [uint64\\_t](#) requestID)  
*Records an MpiRequestTest event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_MpiSend](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [uint32\\_t](#) receiver, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength)



## E.14 of2/OTF2\_EvtWriter.h File Reference

---

*Records an MpiSend event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_OmpAcquireLock](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, uint32\_t lockID, uint32\_t acquisitionOrder)

*Records an OmpAcquireLock event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_OmpFork](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, uint32\_t numberOfRequestedThreads)

*Records an OmpFork event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_OmpJoin](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time)

*Records an OmpJoin event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_OmpReleaseLock](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, uint32\_t lockID, uint32\_t acquisitionOrder)

*Records an OmpReleaseLock event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_OmpTaskComplete](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, uint64\_t taskID)

*Records an OmpTaskComplete event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_OmpTaskCreate](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, uint64\_t taskID)

*Records an OmpTaskCreate event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_OmpTaskSwitch](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, uint64\_t taskID)

*Records an OmpTaskSwitch event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ParameterInt](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_ParameterRef](#) parameter, int64\_t value)

*Records an ParameterInt event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ParameterString](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_ParameterRef](#) parameter, [OTF2\\_StringRef](#) string)

*Records an ParameterString event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ParameterUnsignedInt](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_ParameterRef](#) parameter, uint64\_t value)

*Records an ParameterUnsignedInt event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_Rewind](#) ([OTF2\\_EvtWriter](#) \*writer, [uint32\\_t](#) rewindId)  
*Please give me a documantation.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaAcquireLock](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) remote, [uint64\\_t](#) lockId, [OTF2\\_LockType](#) lockType)  
*Records an RmaAcquireLock event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaAtomic](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) remote, [OTF2\\_RmaAtomicType](#) type, [uint64\\_t](#) bytesSent, [uint64\\_t](#) bytesReceived, [uint64\\_t](#) matchingId)  
*Records an RmaAtomic event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaCollectiveBegin](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time)  
*Records an RmaCollectiveBegin event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaCollectiveEnd](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_CollectiveOp](#) collectiveOp, [OTF2\\_RmaSyncLevel](#) syncLevel, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) root, [uint64\\_t](#) bytesSent, [uint64\\_t](#) bytesReceived)  
*Records an RmaCollectiveEnd event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaGet](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) remote, [uint64\\_t](#) bytes, [uint64\\_t](#) matchingId)  
*Records an RmaGet event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaGroupSync](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaSyncLevel](#) syncLevel, [OTF2\\_RmaWinRef](#) win, [OTF2\\_GroupRef](#) group)  
*Records an RmaGroupSync event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaOpCompleteBlocking](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint64\\_t](#) matchingId)  
*Records an RmaOpCompleteBlocking event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaOpCompleteNonBlocking](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint64\\_t](#) matchingId)  
*Records an RmaOpCompleteNonBlocking event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_RmaOpCompleteRemote](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint64\\_t](#) matchingId)  
*Records an RmaOpCompleteRemote event.*

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaOpTest](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint64\\_t](#) matchingId)  
*Records an RmaOpTest event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaPut](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) remote, [uint64\\_t](#) bytes, [uint64\\_t](#) matchingId)  
*Records an RmaPut event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaReleaseLock](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) remote, [uint64\\_t](#) lockId)  
*Records an RmaReleaseLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaRequestLock](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) remote, [uint64\\_t](#) lockId, [OTF2\\_LockType](#) lockType)  
*Records an RmaRequestLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaSync](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) remote, [OTF2\\_RmaSyncType](#) syncType)  
*Records an RmaSync event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaTryLock](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win, [uint32\\_t](#) remote, [uint64\\_t](#) lockId, [OTF2\\_LockType](#) lockType)  
*Records an RmaTryLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaWaitChange](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win)  
*Records an RmaWaitChange event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaWinCreate](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win)  
*Records an RmaWinCreate event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_RmaWinDestroy](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_RmaWinRef](#) win)  
*Records an RmaWinDestroy event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_SetLocationID](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_LocationRef](#) location)  
*The location ID is not always known on measurment start, and only needed on the first buffer flush to generate the file name. This function enables setting of the location ID after generating the buffer object.*

- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_SetUserData](#) ([OTF2\\_EvtWriter](#) \*writer, void \*userData)  
*Function to set user defined data to a writer object.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_StoreRewindPoint](#) ([OTF2\\_EvtWriter](#) \*writer, uint32\_t rewindId)  
*Please give me a documantation.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ThreadAcquireLock](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_Paradigm](#) model, uint32\_t lockID, uint32\_t acquisitionOrder)  
*Records an ThreadAcquireLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ThreadBegin](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_CommRef](#) threadContingent, uint64\_t sequenceCount)  
*Records an ThreadBegin event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ThreadCreate](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_CommRef](#) threadContingent, uint64\_t sequenceCount)  
*Records an ThreadCreate event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ThreadEnd](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_CommRef](#) threadContingent, uint64\_t sequenceCount)  
*Records an ThreadEnd event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ThreadFork](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_Paradigm](#) model, uint32\_t numberOfRequestedThreads)  
*Records an ThreadFork event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ThreadJoin](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_Paradigm](#) model)  
*Records an ThreadJoin event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ThreadReleaseLock](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_Paradigm](#) model, uint32\_t lockID, uint32\_t acquisitionOrder)  
*Records an ThreadReleaseLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_EvtWriter\\_ThreadTaskComplete](#) ([OTF2\\_EvtWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) time, [OTF2\\_CommRef](#) threadTeam, uint32\_t creatingThread, uint32\_t generationNumber)  
*Records an ThreadTaskComplete event.*

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_ThreadTaskCreate](#) ([OTF2\\_EvtWriter \\*writer](#), [OTF2\\_AttributeList \\*attributeList](#), [OTF2\\_TimeStamp time](#), [OTF2\\_CommRef threadTeam](#), [uint32\\_t creatingThread](#), [uint32\\_t generationNumber](#))  
*Records an ThreadTaskCreate event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_ThreadTaskSwitch](#) ([OTF2\\_EvtWriter \\*writer](#), [OTF2\\_AttributeList \\*attributeList](#), [OTF2\\_TimeStamp time](#), [OTF2\\_CommRef threadTeam](#), [uint32\\_t creatingThread](#), [uint32\\_t generationNumber](#))  
*Records an ThreadTaskSwitch event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_ThreadTeamBegin](#) ([OTF2\\_EvtWriter \\*writer](#), [OTF2\\_AttributeList \\*attributeList](#), [OTF2\\_TimeStamp time](#), [OTF2\\_CommRef threadTeam](#))  
*Records an ThreadTeamBegin event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_ThreadTeamEnd](#) ([OTF2\\_EvtWriter \\*writer](#), [OTF2\\_AttributeList \\*attributeList](#), [OTF2\\_TimeStamp time](#), [OTF2\\_CommRef threadTeam](#))  
*Records an ThreadTeamEnd event.*
- [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_ThreadWait](#) ([OTF2\\_EvtWriter \\*writer](#), [OTF2\\_AttributeList \\*attributeList](#), [OTF2\\_TimeStamp time](#), [OTF2\\_CommRef threadContingent](#), [uint64\\_t sequenceCount](#))  
*Records an ThreadWait event.*

### E.14.1 Detailed Description

This lowest user-visible layer provides write routines to write event data of a single location.

#### Source Template:

*templates/OTF2\_EvtWriter.tmpl.h*

### E.14.2 Function Documentation

- E.14.2.1** [OTF2\\_ErrorCode OTF2\\_EvtWriter\\_BufferFlush](#) ( [OTF2\\_EvtWriter \\*writer](#), [OTF2\\_AttributeList \\* attributeList](#), [OTF2\\_TimeStamp time](#), [OTF2\\_TimeStamp stopTime](#) )

Records an BufferFlush event.

This event signals that the internal buffer was flushed at the given time.

#### Parameters

<i>writer</i>	Writer object.
---------------	----------------

## APPENDIX E. FILE DOCUMENTATION

---

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>stopTime</i>	The time the buffer flush finished.

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.2** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_ClearRewindPoint** ( **OTF2\_EvtWriter \*** *writer*, **uint32\_t** *rewindId* )

Please give me a documantation.

### Parameters

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

### Since

Version 1.1

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.3** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_Enter** ( **OTF2\_EvtWriter \*** *writer*, **OTF2\_AttributeList \*** *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_RegionRef** *region* )

Records an Enter event.

An enter record indicates that the program enters a code region.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

<i>region</i>	Needs to be defined in a definition record References a <a href="#">Region</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_REGION</a> is available.
---------------	--

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.4** `OTF2_ErrorCode OTF2_EvtWriter_GetLocationID ( const OTF2_EvtWriter * writer, OTF2_LocationRef * locationID )`

Function to get the location ID of a writer object.

### Parameters

<i>writer</i>	Writer object which has to be deleted
<i>locationID</i>	Pointer to a variable where the ID is returned in

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.5** `OTF2_ErrorCode OTF2_EvtWriter_GetNumberOfEvents ( OTF2_EvtWriter * writer, uint64_t * numberOfEvents )`

Get the number of events.

Get the number of events written with this event writer. You should call this function right before closing the event writer to get the correct number of stored event records.

### Parameters

	<i>writer</i>	Writer object.
out	<i>numberOfEvents</i>	Return pointer to the number of events.

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.6** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_GetUserData** ( **const** **OTF2\_EvtWriter** \* *writer*, **void** \*\* *userData* )

Function to get the location of a writer object.

**Parameters**

	<i>writer</i>	Writer object.
out	<i>userData</i>	Pointer to a variable where the pointer to the location is returned in.

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.7** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_Leave** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_RegionRef** *region* )

Records an Leave event.

A leave record indicates that the program leaves a code region.

**Parameters**

	<i>writer</i>	Writer object.
	<i>attributeList</i>	Generic attributes for the event.
	<i>time</i>	The time for this event.
	<i>region</i>	Needs to be defined in a definition record References a <a href="#"><i>Region</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_REGION</i></a> is available.

**Since**

Version 1.0

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.8** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_MeasurementOnOff** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_MeasurementMode** *measurementMode* )

Records an MeasurementOnOff event.



## E.14 oftf2/OTF2\_EvtWriter.h File Reference

---

This event signals where the measurement system turned measurement on or off.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>measurementMode</i>	Is the measurement turned on ( <a href="#">OTF2_MEASUREMENT_ON</a> ) or off ( <a href="#">OTF2_MEASUREMENT_OFF</a> )?

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.9** `OTF2_ErrorCode OTF2_EvtWriter_Metric ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *  
typeIDs, const OTF2_MetricValue * metricValues )`

Records an Metric event.

A metric event is always stored at the location that recorded the metric. A metric event can reference a metric class or metric instance. Therefore, metric classes and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>metric</i>	Could be a metric class or a metric instance. References a <a href="#">MetricClass</a> , or a <a href="#">MetricInstance</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_METRIC</a> is available.
<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding <a href="#">MetricMember</a> definitions.
<i>metricValues</i>	List of metric values.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.14.2.10** **OTF2\_ErrorCode** **OTF2\_EvtWriter.MpiCollectiveBegin** ( **OTF2\_EvtWriter**  
\* **writer**, **OTF2\_AttributeList** \* **attributeList**, **OTF2\_TimeStamp** **time** )

Records an MpiCollectiveBegin event.

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.).

**Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.14.2.11** **OTF2\_ErrorCode** **OTF2\_EvtWriter.MpiCollectiveEnd** ( **OTF2\_EvtWriter**  
\* **writer**, **OTF2\_AttributeList** \* **attributeList**, **OTF2\_TimeStamp** **time**,  
**OTF2\_CollectiveOp** **collectiveOp**, **OTF2\_CommRef** **communicator**,  
**uint32\_t** **root**, **uint64\_t** **sizeSent**, **uint64\_t** **sizeReceived** )

Records an MpiCollectiveEnd event.

A MpiCollectiveEnd record marks the end of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.). It keeps the necessary information for this event: type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

**Parameters**

<i>writer</i>	Writer object.
---------------	----------------

## E.14 of2/OTF2\_EvtWriter.h File Reference

---

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>root</i>	MPI rank of root in <code>communicator</code> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.12** `OTF2_ErrorCode OTF2_EvtWriter_Mpilrecv ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t  
msgLength, uint64_t requestID )`

Records an `MpiIrecv` event.

A `MpiIrecv` record indicates that a MPI message was received (`MPI_IRecv`). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.14.2.13** **OTF2\_ErrorCode** **OTF2\_EvtWriter.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.

**E.14.2.14** **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 Mpilsend event.

A Mpilsend record indicates that a MPI message send process was initiated (MPI\_ISEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

**Parameters**

---

## E.14 oftf2/OTF2\_EvtWriter.h File Reference

---

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.15** `OTF2_ErrorCode OTF2_EvtWriter.MpiIsendComplete ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, uint64_t requestID )`

Records an `MpiIsendComplete` event.

Signals the completion of non-blocking send request.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.14.2.16** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_MpiRecv** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **uint32\_t** *sender*, **OTF2\_CommRef** *communicator*, **uint32\_t** *msgTag*, **uint64\_t** *msgLength* )

Records an MpiRecv event.

A MpiRecv record indicates that a MPI message was received (MPI\_RECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>sender</i>	MPI rank of sender in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.17** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_MpiRequestCancelled** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **uint64\_t** *requestID* )

Records an MpiRequestCancelled event.

This events appears if the program canceled a request.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

## E.14 oftf2/OTF2\_EvtWriter.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.18** **OTF2\_ErrorCode** OTF2\_EvtWriter.MpiRequestTest ( OTF2\_EvtWriter \* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*, uint64\_t *requestID* )

Records an MpiRequestTest event.

This events appears if the program tests if a request has already completed but the test failed.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.19** **OTF2\_ErrorCode** OTF2\_EvtWriter.MpiSend ( OTF2\_EvtWriter \* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*, uint32\_t *receiver*, OTF2\_CommRef *communicator*, uint32\_t *msgTag*, uint64\_t *msgLength* )

Records an MpiSend event.

A MpiSend record indicates that a MPI message send process was initiated (MPI\_SEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

### Parameters

## APPENDIX E. FILE DOCUMENTATION

---

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi- cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.20** `OTF2_ErrorCode OTF2_EvtWriter_OmpAcquireLock ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, uint32_t lockID, uint32_t acquisitionOrder )`

Records an OmpAcquireLock event.

An OmpAcquireLock record marks that a thread acquires an OpenMP lock.

This event record is superseded by the [ThreadAcquireLock](#) event record and should not be used when the [ThreadAcquireLock](#) event record is in use.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>lockID</i>	ID of the lock.
<i>acqui- sitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.0



## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

### Deprecated

In version 1.2

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.21** `OTF2_ErrorCode OTF2_EvtWriter_OmpFork ( OTF2_EvtWriter * writer,  
OTF2_AttributeList * attributeList, OTF2_TimeStamp time, uint32_t  
numberOfRequestedThreads )`

Records an OmpFork event.

An OmpFork record marks that an OpenMP Thread forks a thread team.

This event record is superseded by the [\*ThreadFork\*](#) event record and should not be used when the [\*ThreadFork\*](#) event record is in use.

#### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>numberOfRequestedThreads</i>	Requested size of the team.

#### Since

Version 1.0

### Deprecated

In version 1.2

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.22** `OTF2_ErrorCode OTF2_EvtWriter_OmpJoin ( OTF2_EvtWriter * writer,  
OTF2_AttributeList * attributeList, OTF2_TimeStamp time )`

Records an OmpJoin event.

---

## APPENDIX E. FILE DOCUMENTATION

---

An OmpJoin record marks that a team of threads is joint and only the master thread continues execution.

This event record is superseded by the [ThreadJoin](#) event record and should not be used when the [ThreadJoin](#) event record is in use.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.23** `OTF2_StatusCode OTF2_EvtWriter_OmpReleaseLock ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, uint32_t lockID, uint32_t acquisitionOrder )`

Records an OmpReleaseLock event.

An OmpReleaseLock record marks that a thread releases an OpenMP lock.

This event record is superseded by the [ThreadReleaseLock](#) event record and should not be used when the [ThreadReleaseLock](#) event record is in use.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

## E.14 oftf2/OTF2\_EvtWriter.h File Reference

---

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.24** `OTF2_ErrorCode OTF2_EvtWriter_OmpTaskComplete ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, uint64_t taskID )`

Records an OmpTaskComplete event.

An OmpTaskComplete record indicates that the execution of an OpenMP task has finished.

This event record is superseded by the [\*ThreadTaskComplete\*](#) event record and should not be used when the [\*ThreadTaskComplete\*](#) event record is in use.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	Identifier of the completed task instance.

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.25** `OTF2_ErrorCode OTF2_EvtWriter_OmpTaskCreate ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, uint64_t taskID )`

Records an OmpTaskCreate event.

An OmpTaskCreate record marks that an OpenMP Task was/will be created in the current region.

This event record is superseded by the [ThreadTaskCreate](#) event record and should not be used when the [ThreadTaskCreate](#) event record is in use.

#### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	Identifier of the newly created task instance.

#### Since

Version 1.0

#### Deprecated

In version 1.2

#### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.26** `OTF2_ErrorCode OTF2_EvtWriter_OmpTaskSwitch ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, uint64_t taskID )`

Records an OmpTaskSwitch event.

An OmpTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

This event record is superseded by the [ThreadTaskSwitch](#) event record and should not be used when the [ThreadTaskSwitch](#) event record is in use.

#### Parameters

<i>writer</i>	Writer object.
---------------	----------------

## E.14 oftf2/OTF2\_EvtWriter.h File Reference

---

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	Identifier of the now active task instance.

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.27** `OTF2_ErrorCode OTF2_EvtWriter.ParameterInt ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_ParameterRef parameter, int64_t value )`

Records an ParameterInt event.

A ParameterInt record marks that in the current region, the specified integer parameter has the specified value.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>parameter</i>	Parameter ID. References a <a href="#"><i>Parameter</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_PARAMETER</i></a> is available.
<i>value</i>	Value of the recorded parameter.

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.14.2.28** **OTF2\_ErrorCode** **OTF2\_EvtWriter.ParameterString** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_ParameterRef** *parameter*, **OTF2\_StringRef** *string* )

Records an ParameterString event.

A ParameterString record marks that in the current region, the specified string parameter has the specified value.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>string</i>	Value: Handle of a string definition References a <a href="#">String</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_STRING</a> is available.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.29** **OTF2\_ErrorCode** **OTF2\_EvtWriter.ParameterUnsignedInt** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_ParameterRef** *parameter*, **uint64\_t** *value* )

Records an ParameterUnsignedInt event.

A ParameterUnsignedInt record marks that in the current region, the specified unsigned integer parameter has the specified value.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.30** `OTF2_ErrorCode OTF2_EvtWriter.Rewind ( OTF2_EvtWriter * writer,  
uint32_t rewindId )`

Please give me a documantation.

### Parameters

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

### Since

Version 1.1

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.31** `OTF2_ErrorCode OTF2_EvtWriter.RmaAcquireLock ( OTF2_EvtWriter  
* writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp  
time, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId,  
OTF2_LockType lockType )`

Records an RmaAcquireLock event.

An RmaAcquireLock record denotes the time a lock was aquired by the process.

### Parameters

<i>writer</i>	Writer object.
---------------	----------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.32** `OTF2_ErrorCode OTF2_EvtWriter_RmaAtomic ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type,  
uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId )`

Records an RmaAtomic event.

An RmaAtomic record denotes the time a atomic operation was issued.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the target process.
<i>type</i>	Type of atomic operation.
<i>bytesSent</i>	Bytes sent to target.
<i>bytesReceived</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2



## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.33** `OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveBegin ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time )`

Records an RmaCollectiveBegin event.

An RmaCollectiveBegin record denotes the beginnig of a collective RMA operation.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.34** `OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveEnd ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived )`

Records an RmaCollectiveEnd event.

"An RmaCollectiveEnd record denotes the end of a collective RMA operation.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>syncLevel</i>	Synchronization level of this collective operation.

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>root</i>	Root process for this operation.
<i>bytesSent</i>	Bytes sent in operation.
<i>bytesReceived</i>	Bytes receives in operation.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.35** `OTF2_ErrorCode OTF2_EvtWriter.RmaGet ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId  
)`

Records an RmaGet event.

An RmaGet record denotes the time a get operation was issued.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

**E.14.2.36** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_RmaGroupSync** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_RmaSyncLevel** *syncLevel*, **OTF2\_RmaWinRef** *win*, **OTF2\_GroupRef** *group* )

Records an RmaGroupSync event.

An RmaGroupSync record denotes the synchronization with a subgroup of processes on a window.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>group</i>	Group of remote processes involved in synchronization. References a <a href="#">Group</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_GROUP</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.37** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_RmaOpCompleteBlocking** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_RmaWinRef** *win*, **uint64\_t** *matchingId* )

Records an RmaOpCompleteBlocking event.

An RmaOpCompleteBlocking record denotes the local completion of a blocking RMA operation.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

## APPENDIX E. FILE DOCUMENTATION

---

<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.38** `OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteNonBlocking ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint64_t matchingId )`

Records an RmaOpCompleteNonBlocking event.

An RmaOpCompleteNonBlocking record denotes the local completion of a non-blocking RMA operation.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

**E.14.2.39** **OTF2\_ErrorCode** OTF2\_EvtWriter\_RmaOpCompleteRemote (   
 OTF2\_EvtWriter \* *writer*, OTF2\_AttributeList \* *attributeList*,  
 OTF2\_TimeStamp *time*, OTF2\_RmaWinRef *win*, uint64\_t *matchingId* )

Records an RmaOpCompleteRemote event.

An RmaOpCompleteRemote record denotes the local completion of an RMA operation.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.40** **OTF2\_ErrorCode** OTF2\_EvtWriter\_RmaOpTest ( OTF2\_EvtWriter \*   
 *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*,  
 OTF2\_RmaWinRef *win*, uint64\_t *matchingId* )

Records an RmaOpTest event.

An RmaOpTest record denotes that a non-blocking RMA operation has been tested for completion unsuccessfully.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.41** `OTF2_StatusCode 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 <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> is available.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes sent to target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.42** `OTF2_StatusCode OTF2_EvtWriter_RmaReleaseLock ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId )`

Records an RmaReleaseLock event.

An RmaReleaseLock record denotes the time the lock was released.

## E.14 oftf2/OTF2\_EvtWriter.h File Reference

---

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock released, if multiple locks are defined on a window.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.43** `OTF2_ErrorCode OTF2_EvtWriter.RmaRequestLock ( OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType )`

Records an RmaRequestLock event.

An RmaRequestLock record denotes the time a lock was requested and with it the earliest time it could have been granted. It is used to mark (possibly) non-blocking lock request, as defined by the MPI standard.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

### Since

Version 1.2

## Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.44** `OTF2_ErrorCode OTF2_EvtWriter_RmaSync ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaSyncType  
syncType )`

Records an RmaSync event.

An RmaSync record denotes the direct synchronization with a possibly remote process.

## Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>syncType</i>	Type of synchronization.

## Since

Version 1.2

## Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.45** `OTF2_ErrorCode OTF2_EvtWriter_RmaTryLock ( OTF2_EvtWriter  
* writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp  
time, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId,  
OTF2_LockType lockType )`

Records an RmaTryLock event.

An RmaTryLock record denotes the time of an unsuccessful attempt to acquire the lock.

## Parameters



## E.14 oftf2/OTF2\_EvtWriter.h File Reference

---

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.46 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaWaitChange ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_RmaWinRef win )**

Records an RmaWaitChange event.

An RmaWaitChange record denotes the change of a window that was waited for.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.14.2.47** **OTF2\_ErrorCode** **OTF2\_EvtWriter.RmaWinCreate** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_RmaWinRef** *win* )

Records an RmaWinCreate event.

An RmaWinCreate record denotes the creation of an RMA window.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window created. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_-MAPPING_RMA_WIN</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.48** **OTF2\_ErrorCode** **OTF2\_EvtWriter.RmaWinDestroy** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_RmaWinRef** *win* )

Records an RmaWinDestroy event.

An RmaWinDestroy record denotes the destruction of an RMA window.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window destructed. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_-MAPPING_RMA_WIN</a> is available.

### Since

Version 1.2

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.49** **OTF2\_ErrorCode** OTF2\_EvtWriter.SetLocationID ( OTF2\_EvtWriter \*  
*writer*, OTF2\_LocationRef *location* )

The location ID is not always known on measurment start, and only needed on the first buffer flush to generate the file name. This function enables setting of the location ID after generating the buffer object.

### Parameters

<i>writer</i>	Writer object.
<i>location</i>	Location ID.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.50** **OTF2\_ErrorCode** OTF2\_EvtWriter.SetUserData ( OTF2\_EvtWriter \*  
*writer*, void \* *userData* )

Function to set user defined data to a writer object.

### Parameters

<i>writer</i>	Writer object.
<i>userData</i>	User provided data. Can be queried with <a href="#"><i>OTF2_EvtWriter_GetUserData</i></a> .

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.51** **OTF2\_ErrorCode** OTF2\_EvtWriter.StoreRewindPoint ( OTF2\_EvtWriter  
\* *writer*, uint32\_t *rewindId* )

Please give me a documantation.

### Parameters

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

**Since**

Version 1.1

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.14.2.52** **OTF2\_ErrorCode** **OTF2\_EvtWriter.ThreadAcquireLock** (  
**OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*,  
**OTF2\_TimeStamp** *time*, **OTF2\_Paradigm** *model*, **uint32\_t** *lockID*,  
**uint32\_t** *acquisitionOrder* )

Records an ThreadAcquireLock event.

An ThreadAcquireLock record marks that a thread acquires an lock.

**Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.14.2.53** **OTF2\_ErrorCode** **OTF2\_EvtWriter.ThreadBegin** ( **OTF2\_EvtWriter** \*  
*writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*,  
**OTF2\_CommRef** *threadContingent*, **uint64\_t** *sequenceCount* )

Records an ThreadBegin event.

Marks the begin of a thread created by another thread.

**Parameters**

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>sequence-Count</i>	A threadContingent unique number. The corresponding <a href="#">Thread-Create</a> event does have the same number.

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.54** `OTF2_StatusCode OTF2_EvtWriter.ThreadCreate ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_CommRef threadContingent, uint64_t sequenceCount )`

Records an ThreadCreate event.

The location created successfully a new thread.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>sequence-Count</i>	A threadContingent unique number. The corresponding <a href="#">Thread-Begin</a> event does have the same number.

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.14.2.55** **OTF2\_ErrorCode** **OTF2\_EvtWriter.ThreadEnd** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_CommRef** *threadContingent*, **uint64\_t** *sequenceCount* )

Records an ThreadEnd event.

Marks the end of a thread.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>sequenceCount</i>	A <code>threadContingent</code> unique number. The corresponding <a href="#">ThreadWait</a> event does have the same number. <a href="#">OTF2_UNDEFINED_UINT64</a> in case no corresponding <a href="#">ThreadWait</a> event exists.

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.56** **OTF2\_ErrorCode** **OTF2\_EvtWriter.ThreadFork** ( **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*, **OTF2\_Paradigm** *model*, **uint32\_t** *numberOfRequestedThreads* )

Records an ThreadFork event.

An ThreadFork record marks that an thread forks a thread team.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>model</i>	The threading paradigm this event took place.
<i>numberOfRequestedThreads</i>	Requested size of the team.

## E.14 oftf2/OTF2\_EvtWriter.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.57** `OTF2_ErrorCode OTF2_EvtWriter.ThreadJoin ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_Paradigm model )`

Records an ThreadJoin event.

An ThreadJoin record marks that a team of threads is joint and only the master thread continues execution.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>model</i>	The threading paradigm this event took place.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.58** `OTF2_ErrorCode OTF2_EvtWriter.ThreadReleaseLock (  
OTF2_EvtWriter * writer, OTF2_AttributeList * attributeList,  
OTF2_TimeStamp time, OTF2_Paradigm model, uint32_t lockID,  
uint32_t acquisitionOrder )`

Records an ThreadReleaseLock event.

An ThreadReleaseLock record marks that a thread releases an lock.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.14.2.59** **OTF2\_ErrorCode** **OTF2\_EvtWriter\_ThreadTaskComplete** (  
    **OTF2\_EvtWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*,  
    **OTF2\_TimeStamp** *time*, **OTF2\_CommRef** *threadTeam*, **uint32\_t**  
    *creatingThread*, **uint32\_t** *generationNumber* )

Records an ThreadTaskComplete event.

An ThreadTaskComplete record indicates that the execution of an OpenMP task has finished.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <a href="#"><i>Comm</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_COMM</i></a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.



## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

**E.14.2.60** **OTF2\_ErrorCode** **OTF2\_EvtWriter.ThreadTaskCreate** ( **OTF2\_EvtWriter**  
\* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp**  
*time*, **OTF2\_CommRef** *threadTeam*, **uint32\_t** *creatingThread*, **uint32\_t**  
*generationNumber* )

Records an ThreadTaskCreate event.

An ThreadTaskCreate record marks that an task in was/will be created and will be processed by the specified thread team.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.61** **OTF2\_ErrorCode** **OTF2\_EvtWriter.ThreadTaskSwitch** ( **OTF2\_EvtWriter**  
\* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp**  
*time*, **OTF2\_CommRef** *threadTeam*, **uint32\_t** *creatingThread*, **uint32\_t**  
*generationNumber* )

Records an ThreadTaskSwitch event.

An ThreadTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

### Parameters

<i>writer</i>	Writer object.
---------------	----------------

## APPENDIX E. FILE DOCUMENTATION

---

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.62** **OTF2\_StatusCode** **OTF2\_EvtWriter.ThreadTeamBegin** ( **OTF2\_EvtWriter**  
\* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *time*,  
**OTF2\_CommRef** *threadTeam* )

Records an ThreadTeamBegin event.

The current location enters the specified thread team.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.14 otf2/OTF2\_EvtWriter.h File Reference

---

**E.14.2.63** `OTF2_StatusCode OTF2_EvtWriter.ThreadTeamEnd ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_CommRef threadTeam )`

Records an ThreadTeamEnd event.

The current location leaves the specified thread team.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.14.2.64** `OTF2_StatusCode OTF2_EvtWriter.ThreadWait ( OTF2_EvtWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,  
OTF2_CommRef threadContingent, uint64_t sequenceCount )`

Records an ThreadWait event.

The location waits for the completion of another thread.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>sequence-Count</i>	A threadContingent unique number. The corresponding <a href="#">Thread-End</a> event does have the same number.

**Since**

Version 1.3

**Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.15 otf2/OTF2\_GeneralDefinitions.h File Reference**

This header file provides general definitions which should be accessible in all internal and external modules.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
```

**Defines**

- #define [OTF2\\_CHUNK\\_SIZE\\_MAX](#) ( uint64\_t )( 1024 \* 1024 \* 16 )  
*Defines the maximum size of a chunk.*
- #define [OTF2\\_CHUNK\\_SIZE\\_MIN](#) ( uint64\_t )( 256 \* 1024 )  
*Defines the minimum size of a chunk.*
- #define [OTF2\\_UNDEFINED\\_ATTRIBUTE](#) ( ( [OTF2\\_AttributeRef](#) )OTF2\_UNDEFINED\_UINT32 )  
*The invalid value for a reference to a [Attribute](#) definition.*
- #define [OTF2\\_UNDEFINED\\_CALLPATH](#) ( ( [OTF2\\_CallpathRef](#) )OTF2\_UNDEFINED\_UINT32 )  
*The invalid value for a reference to a [Callpath](#) definition.*
- #define [OTF2\\_UNDEFINED\\_CALLSITE](#) ( ( [OTF2\\_CallsiteRef](#) )OTF2\_UNDEFINED\_UINT32 )  
*The invalid value for a reference to a [Callsite](#) definition.*
- #define [OTF2\\_UNDEFINED\\_CART\\_DIMENSION](#) ( ( [OTF2\\_CartDimensionRef](#) )OTF2\_UNDEFINED\_UINT32 )  
*The invalid value for a reference to a [CartDimension](#) definition.*
- #define [OTF2\\_UNDEFINED\\_CART\\_TOPOLOGY](#) ( ( [OTF2\\_CartTopologyRef](#) )OTF2\_UNDEFINED\_UINT32 )  
*The invalid value for a reference to a [CartTopology](#) definition.*
- #define [OTF2\\_UNDEFINED\\_COMM](#) ( ( [OTF2\\_CommRef](#) )OTF2\_UNDEFINED\_UINT32 )  
*The invalid value for a reference to a [Comm](#) definition.*

## E.15 otf2/OTF2\_GeneralDefinitions.h File Reference

---

- #define `OTF2_UNDEFINED_GROUP` ( ( `OTF2_GroupRef` )OTF2\_UNDEFINED\_-  
UINT32 )

*The invalid value for a reference to a [Group](#) definition.*

- #define `OTF2_UNDEFINED_LOCATION` ( ( `OTF2_LocationRef` )OTF2\_-  
UNDEFINED\_UINT64 )

*The invalid value for a reference to a [Location](#) definition.*

- #define `OTF2_UNDEFINED_LOCATION_GROUP` ( ( `OTF2_LocationGroupRef`  
)OTF2\_UNDEFINED\_UINT32 )

*The invalid value for a reference to a [LocationGroup](#) definition.*

- #define `OTF2_UNDEFINED_METRIC` ( ( `OTF2_MetricRef` )OTF2\_UNDEFINED\_-  
UINT32 )

*The invalid value for a reference to a [MetricClass](#), or a [MetricInstance](#) definition.*

- #define `OTF2_UNDEFINED_METRIC_MEMBER` ( ( `OTF2_MetricMemberRef`  
)OTF2\_UNDEFINED\_UINT32 )

*The invalid value for a reference to a [MetricMember](#) definition.*

- #define `OTF2_UNDEFINED_PARAMETER` ( ( `OTF2_ParameterRef` )OTF2\_-  
UNDEFINED\_UINT32 )

*The invalid value for a reference to a [Parameter](#) definition.*

- #define `OTF2_UNDEFINED_REGION` ( ( `OTF2_RegionRef` )OTF2\_UNDEFINED\_-  
UINT32 )

*The invalid value for a reference to a [Region](#) definition.*

- #define `OTF2_UNDEFINED_RMA_WIN` ( ( `OTF2_RmaWinRef` )OTF2\_-  
UNDEFINED\_UINT32 )

*The invalid value for a reference to a [RmaWin](#) definition.*

- #define `OTF2_UNDEFINED_STRING` ( ( `OTF2_StringRef` )OTF2\_UNDEFINED\_-  
UINT32 )

*The invalid value for a reference to a [String](#) definition.*

- #define `OTF2_UNDEFINED_SYSTEM_TREE_NODE` ( ( `OTF2_SystemTreeNodeRef`  
)OTF2\_UNDEFINED\_UINT32 )

*The invalid value for a reference to a [SystemTreeNode](#) definition.*

- #define `OTF2_UNDEFINED_TIMESTAMP` OTF2\_UNDEFINED\_UINT64

*Undefined value for [OTF2\\_TimeStamp](#).*

- #define `OTF2_UNDEFINED_TYPE` OTF2\_UNDEFINED\_UINT8

*Undefined value for enums.*

**OTF2 library version.**

- #define [OTF2\\_VERSION\\_MAJOR](#) 1  
*Major version number of this OTF2 version.*
- #define [OTF2\\_VERSION\\_MINOR](#) 4  
*Minor version number of this OTF2 version.*
- #define [OTF2\\_VERSION\\_BUGFIX](#) 0  
*Bugfix version number of this OTF2 version.*
- #define [OTF2\\_VERSION\\_SUFFIX](#) ""  
*Any string suffix of this OTF2 version.*
- #define [OTF2\\_VERSION](#) "1.4"  
*The OTF2 version as string.*

**Standard undefined values for basic data types.**

- #define [OTF2\\_UNDEFINED\\_UINT8](#) ( ( uint8\_t )( ~( ( uint8\_t )0u ) ) )  
*Undefined value for type uint8\_t.*
- #define [OTF2\\_UNDEFINED\\_UINT16](#) ( ( uint16\_t )( ~( ( uint16\_t )0u ) ) )  
*Undefined value for type uint16\_t.*
- #define [OTF2\\_UNDEFINED\\_UINT32](#) ( ( uint32\_t )( ~( ( uint32\_t )0u ) ) )  
*Undefined value for type uint32\_t.*
- #define [OTF2\\_UNDEFINED\\_UINT64](#) ( ( uint64\_t )( ~( ( uint64\_t )0u ) ) )  
*Undefined value for type uint64\_t.*

**Typedefs**

- typedef uint32\_t [OTF2\\_AttributeRef](#)  
*Type used to indicate a reference to a [Attribute](#) definition.*
- typedef uint32\_t [OTF2\\_CallpathRef](#)  
*Type used to indicate a reference to a [Callpath](#) definition.*
- typedef uint32\_t [OTF2\\_CallsiteRef](#)  
*Type used to indicate a reference to a [Callsite](#) definition.*
- typedef uint32\_t [OTF2\\_CartDimensionRef](#)  
*Type used to indicate a reference to a [CartDimension](#) definition.*
- typedef uint32\_t [OTF2\\_CartTopologyRef](#)  
*Type used to indicate a reference to a [CartTopology](#) definition.*
- typedef uint32\_t [OTF2\\_CommRef](#)  
*Type used to indicate a reference to a [Comm](#) definition.*

## E.15 otf2/OTF2\_GeneralDefinitions.h File Reference

---

- typedef uint8\_t [OTF2\\_Compression](#)  
*Defines which compression is used. Please see [OTF2\\_Compression\\_enum](#) for a detailed description.*
- typedef struct OTF2\_DefReader\_struct [OTF2\\_DefReader](#)  
*OTF2 local definition reader handle.*
- typedef struct OTF2\_EvtReader\_struct [OTF2\\_EvtReader](#)  
*OTF2 local event reader handle.*
- typedef uint8\_t [OTF2\\_FileMode](#)  
*Defines how to interact with files. Please see [OTF2\\_FileMode\\_enum](#) for a detailed description.*
- typedef uint8\_t [OTF2\\_FileSubstrate](#)  
*Wrapper for enum [OTF2\\_FileSubstrate\\_enum](#).*
- typedef uint8\_t [OTF2\\_FileType](#)  
*Wrapper for enum [OTF2\\_FileType\\_enum](#).*
- typedef uint8\_t [OTF2\\_FlushType](#)  
*Defines whether the recorded data is flushed to a file or not. Please see [OTF2\\_FlushType\\_enum](#) for a detailed description.*
- typedef struct OTF2\_GlobalDefReader\_struct [OTF2\\_GlobalDefReader](#)  
*OTF2 global definition reader handle.*
- typedef struct OTF2\_GlobalEvtReader\_struct [OTF2\\_GlobalEvtReader](#)  
*OTF2 global event reader handle.*
- typedef struct OTF2\_GlobalSnapReader\_struct [OTF2\\_GlobalSnapReader](#)  
*OTF2 global snap reader handle.*
- typedef uint32\_t [OTF2\\_GroupRef](#)  
*Type used to indicate a reference to a [Group](#) definition.*
- typedef uint32\_t [OTF2\\_LocationGroupRef](#)  
*Type used to indicate a reference to a [LocationGroup](#) definition.*
- typedef uint64\_t [OTF2\\_LocationRef](#)  
*Type used to indicate a reference to a [Location](#) definition.*
- typedef uint8\_t [OTF2\\_MappingType](#)  
*Wrapper for enum [OTF2\\_MappingType\\_enum](#).*
- typedef struct OTF2\_MarkerReader\_struct [OTF2\\_MarkerReader](#)  
*OTF2 marker reader handle.*
- typedef uint32\_t [OTF2\\_MetricMemberRef](#)  
*Type used to indicate a reference to a [MetricMember](#) definition.*
- typedef uint32\_t [OTF2\\_MetricRef](#)  
*Type used to indicate a reference to a [MetricClass](#), or a [MetricInstance](#) definition.*
- typedef uint8\_t [OTF2\\_Paradigm](#)

Wrapper for enum [OTF2\\_Paradigm\\_enum](#).

- typedef uint32\_t [OTF2\\_ParameterRef](#)  
Type used to indicate a reference to a [Parameter](#) definition.
- typedef uint32\_t [OTF2\\_RegionRef](#)  
Type used to indicate a reference to a [Region](#) definition.
- typedef uint32\_t [OTF2\\_RmaWinRef](#)  
Type used to indicate a reference to a [RmaWin](#) definition.
- typedef struct OTF2\_SnapReader\_struct [OTF2\\_SnapReader](#)  
OTF2 local snap reader handle.
- typedef uint32\_t [OTF2\\_StringRef](#)  
Type used to indicate a reference to a [String](#) definition.
- typedef uint32\_t [OTF2\\_SystemTreeNodeRef](#)  
Type used to indicate a reference to a [SystemTreeNode](#) definition.
- typedef uint8\_t [OTF2\\_ThumbnailType](#)  
Wrapper for enum [OTF2\\_ThumbnailType\\_enum](#).
- typedef uint64\_t [OTF2\\_TimeStamp](#)  
OTF2 time stamp.
- typedef uint8\_t [OTF2\\_Type](#)  
Wrapper for enum [OTF2\\_Type\\_enum](#).

## Enumerations

- enum [OTF2\\_CallbackCode](#) {  
    [OTF2\\_CALLBACK\\_SUCCESS](#) = 0,  
    [OTF2\\_CALLBACK\\_INTERRUPT](#) = ![OTF2\\_CALLBACK\\_SUCCESS](#),  
    [OTF2\\_CALLBACK\\_ERROR](#) = ![OTF2\\_CALLBACK\\_SUCCESS](#) }  
Return value to indicate that the record reading should be interrupted.
- enum [OTF2\\_Compression\\_enum](#) {  
    [OTF2\\_COMPRESSION\\_UNDEFINED](#) = 0,  
    [OTF2\\_COMPRESSION\\_NONE](#) = 1,  
    [OTF2\\_COMPRESSION\\_ZLIB](#) = 2 }  
Defines which compression is used.
- enum [OTF2\\_FileMode\\_enum](#) {  
    [OTF2\\_FILEMODE\\_WRITE](#) = 0,  
    [OTF2\\_FILEMODE\\_READ](#) = 1,  
    [OTF2\\_FILEMODE\\_MODIFY](#) = 2 }  
Defines how to interact with files.



## E.15 otf2/OTF2\_GeneralDefinitions.h File Reference

---

- enum `OTF2_FileSubstrate_enum` {  
    `OTF2_SUBSTRATE_UNDEFINED` = 0,  
    `OTF2_SUBSTRATE_POSIX` = 1,  
    `OTF2_SUBSTRATE_SION` = 2,  
    `OTF2_SUBSTRATE_NONE` = 3 }  
    *Defines which file substrate is used.*
- enum `OTF2_FileType_enum` {  
    `OTF2_FILETYPE_ANCHOR` = 0,  
    `OTF2_FILETYPE_GLOBAL_DEFS` = 1,  
    `OTF2_FILETYPE_LOCAL_DEFS` = 2,  
    `OTF2_FILETYPE_EVENTS` = 3,  
    `OTF2_FILETYPE_SNAPSHOTS` = 4,  
    `OTF2_FILETYPE_THUMBNAIL` = 5,  
    `OTF2_FILETYPE_MARKER` = 6,  
    `OTF2_FILETYPE_SIONRANKMAP` = 7 }  
    *Defines which file type is used.*
- enum `OTF2_FlushType_enum` {  
    `OTF2_NO_FLUSH` = 0,  
    `OTF2_FLUSH` = 1 }  
    *Defines whether the recorded data is flushed to a file or not.*
- enum `OTF2_MappingType_enum` {  
    `OTF2_MAPPING_STRING` = 0,  
    `OTF2_MAPPING_ATTRIBUTE` = 1,  
    `OTF2_MAPPING_LOCATION` = 2,  
    `OTF2_MAPPING_REGION` = 3,  
    `OTF2_MAPPING_GROUP` = 4,  
    `OTF2_MAPPING_METRIC` = 5,  
    `OTF2_MAPPING_COMM` = 6,  
    `OTF2_MAPPING_PARAMETER` = 7,  
    `OTF2_MAPPING_RMA_WIN` = 8,  
    `OTF2_MAPPING_MAX` = 9 }  
    *Possible mappings from local to global identifiers.*

- `enum OTF2_Paradigm_enum {`  
    `OTF2_PARADIGM_UNKNOWN = 0,`  
    `OTF2_PARADIGM_USER = 1,`  
    `OTF2_PARADIGM_COMPILER = 2,`  
    `OTF2_PARADIGM_OPENMP = 3,`  
    `OTF2_PARADIGM_MPI = 4,`  
    `OTF2_PARADIGM_CUDA = 5,`  
    `OTF2_PARADIGM_MEASUREMENT_SYSTEM = 6,`  
    `OTF2_PARADIGM_PTHREAD = 7,`  
    `OTF2_PARADIGM_HMPP = 8,`  
    `OTF2_PARADIGM_OMPSS = 9,`  
    `OTF2_PARADIGM_HARDWARE = 10,`  
    `OTF2_PARADIGM_GASPI = 11,`  
    `OTF2_PARADIGM_UPC = 12,`  
    `OTF2_PARADIGM_SHMEM = 13 }`

*List of known paradigms.*

- `enum OTF2_ThumbnailType_enum {`  
    `OTF2_THUMBNAIL_TYPE_REGION = 0,`  
    `OTF2_THUMBNAIL_TYPE_METRIC = 1,`  
    `OTF2_THUMBNAIL_TYPE_ATTRIBUTES = 2 }`

*Type of definitions used as metric in an thumbnail.*

- `enum OTF2_Type_enum {`  
    `OTF2_TYPE_NONE = 0,`  
    `OTF2_TYPE_UINT8 = 1,`  
    `OTF2_TYPE_UINT16 = 2,`  
    `OTF2_TYPE_UINT32 = 3,`  
    `OTF2_TYPE_UINT64 = 4,`  
    `OTF2_TYPE_INT8 = 5,`  
    `OTF2_TYPE_INT16 = 6,`  
    `OTF2_TYPE_INT32 = 7,`  
    `OTF2_TYPE_INT64 = 8,`  
    `OTF2_TYPE_FLOAT = 9,`  
    `OTF2_TYPE_DOUBLE = 10,`  
    `OTF2_TYPE_STRING = 11,`

## E.15 otf2/OTF2\_GeneralDefinitions.h File Reference

---

```
OTF2_TYPE_ATTRIBUTE = 12,  
OTF2_TYPE_LOCATION = 13,  
OTF2_TYPE_REGION = 14,  
OTF2_TYPE_GROUP = 15,  
OTF2_TYPE_METRIC = 16,  
OTF2_TYPE_COMM = 17,  
OTF2_TYPE_PARAMETER = 18,  
OTF2_TYPE_RMA_WIN = 19 }
```

*OTF2 basic data types.*

### E.15.1 Detailed Description

This header file provides general definitions which should be accessible in all internal and external modules.

#### Source Template:

*templates/OTF2\_GeneralDefinitions.tmpl.h*

### E.15.2 Enumeration Type Documentation

#### E.15.2.1 enum OTF2\_CallbackCode

Return value to indicate that the record reading should be interrupted.

Returning [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#) will stop reading more events, if functions like:

- [\*OTF2\\_Reader\\_ReadLocalEvents\*](#)
- [\*OTF2\\_Reader\\_ReadAllLocalEvents\*](#)
- [\*OTF2\\_Reader\\_ReadLocalEventsBackward\*](#)
- [\*OTF2\\_Reader\\_ReadGlobalEvents\*](#)
- [\*OTF2\\_Reader\\_ReadAllGlobalEvents\*](#)
- [\*OTF2\\_Reader\\_ReadLocalDefinitions\*](#)
- [\*OTF2\\_Reader\\_ReadAllLocalDefinitions\*](#)
- [\*OTF2\\_Reader\\_ReadGlobalDefinitions\*](#)

- *OTF2\_Reader\_ReadAllGlobalDefinitions*

where called. The return value for these functions is *OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK* in this case. It is valid to call any reader functions in such a condition again.

This type is also used as return type in the collective callbacks (see *Operating OTF2 in an collective context*). Any value different than *OTF2\_CALLBACK\_SUCCESS* is treated as an error and the calling function will return *OTF2\_ERROR\_COLLECTIVE\_CALLBACK* to its caller. As the name *OTF2\_CALLBACK\_INTERRUPT* does not really fit, a second alias is named *OTF2\_CALLBACK\_ERROR* provided.

**Enumerator:**

*OTF2\_CALLBACK\_SUCCESS* Record reading can continue.

*OTF2\_CALLBACK\_INTERRUPT* Interrupt record reading. Control returns to the caller of the read function with error *OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK* to signal this. The actual value can be any except *OTF2\_CALLBACK\_SUCCESS*.

*OTF2\_CALLBACK\_ERROR* Signalling an error in the callback.

**E.15.2.2 enum OTF2\_Compression\_enum**

Defines which compression is used.

**Enumerator:**

*OTF2\_COMPRESSION\_UNDEFINED* Undefined.

*OTF2\_COMPRESSION\_NONE* No compression is used.

*OTF2\_COMPRESSION\_ZLIB* Use zlib compression.

**E.15.2.3 enum OTF2\_FileMode\_enum**

Defines how to interact with files.

**Enumerator:**

*OTF2\_FILEMODE\_WRITE* Open a file in write-only mode.

*OTF2\_FILEMODE\_READ* Open a file in read-only mode.

*OTF2\_FILEMODE\_MODIFY* Open a file in write-read mode.

## E.15 otf2/OTF2\_GeneralDefinitions.h File Reference

---

### E.15.2.4 enum OTF2\_FileSubstrate\_enum

Defines which file substrate is used.

#### Since

Version 1.0

#### Enumerator:

**OTF2\_SUBSTRATE\_UNDEFINED** Undefined.

**OTF2\_SUBSTRATE\_POSIX** Use standard posix file interface.

**OTF2\_SUBSTRATE\_SION** Use the interface of the SIONlib to write many logical files into few physical files.

**OTF2\_SUBSTRATE\_NONE** Do not use any file interface. No data is written to a file.

### E.15.2.5 enum OTF2\_FileType\_enum

Defines which file type is used.

#### Since

Version 1.0

#### Enumerator:

**OTF2\_FILETYPE\_ANCHOR** Represents the type for the anchor file (.otf2).

**OTF2\_FILETYPE\_GLOBAL\_DEFS** Represents the type for the global definition file (.def).

**OTF2\_FILETYPE\_LOCAL\_DEFS** Represents the type for a local definition file (.def).

**OTF2\_FILETYPE\_EVENTS** Represents the type for a event file (.evt).

**OTF2\_FILETYPE\_SNAPSHOTS** Represents the type for a snapshot file (.snap).

**OTF2\_FILETYPE\_THUMBNAIL** Represents the type for a thumb file (.thumb).

**OTF2\_FILETYPE\_MARKER** Represents the type for a marker file (.marker).

**OTF2\_FILETYPE\_SIONRANKMAP** Internal file which holds the SION rank map (.srm).

#### E.15.2.6 enum OTF2\_FlushType\_enum

Defines whether the recorded data is flushed to a file or not.

##### Enumerator:

**OTF2\_NO\_FLUSH** Flushing will be suppressed when running out of memory.

**OTF2\_FLUSH** Recorded data is flushed when running out of memory.

#### E.15.2.7 enum OTF2\_MappingType\_enum

Possible mappings from local to global identifiers.

##### Since

Version 1.0

##### Enumerator:

**OTF2\_MAPPING\_STRING** Mapping of *String* identifiers.

**OTF2\_MAPPING\_ATTRIBUTE** Mapping of *Attribute* identifiers.

**OTF2\_MAPPING\_LOCATION** Mapping of *Location* identifiers.

**OTF2\_MAPPING\_REGION** Mapping of *Region* identifiers.

**OTF2\_MAPPING\_GROUP** Mapping of *Group* identifiers.

**OTF2\_MAPPING\_METRIC** Mapping of *Metric* identifiers.

**OTF2\_MAPPING\_COMM** Mapping of *Comm* identifiers.

**OTF2\_MAPPING\_PARAMETER** Mapping of *Parameter* identifiers.

**OTF2\_MAPPING\_RMA\_WIN** Mapping of *RmaWin* identifiers.

##### Since

Version 1.2.

**OTF2\_MAPPING\_MAX** Max entry.

#### E.15.2.8 enum OTF2\_Paradigm\_enum

List of known paradigms.

##### Since

Version 1.1

## E.15 otf2/OTF2\_GeneralDefinitions.h File Reference

---

### Enumerator:

***OTF2\_PARADIGM\_UNKNOWN*** An unknown paradigm.

***OTF2\_PARADIGM\_USER*** User instrumentation.

***OTF2\_PARADIGM\_COMPILER*** Compiler instrumentation.

***OTF2\_PARADIGM\_OPENMP*** OpenMP.

***OTF2\_PARADIGM\_MPI*** MPI.

***OTF2\_PARADIGM\_CUDA*** CUDA.

***OTF2\_PARADIGM\_MEASUREMENT\_SYSTEM*** The measurement software.

**Since**

Version 1.2.

***OTF2\_PARADIGM\_PTHREAD*** POSIX threads.

**Since**

Version 1.3.

***OTF2\_PARADIGM\_HMPP*** HMPP.

**Since**

Version 1.3.

***OTF2\_PARADIGM\_OMPSS*** OmpSs.

**Since**

Version 1.3.

***OTF2\_PARADIGM\_HARDWARE*** Hardware.

**Since**

Version 1.3.

***OTF2\_PARADIGM\_GASPI*** GASPI.

**Since**

Version 1.4.

***OTF2\_PARADIGM\_UPC*** Unified Parallel C (UPC).

**Since**

Version 1.4.

***OTF2\_PARADIGM\_SHMEM*** SGI SHMEM, Cray SHMEM, OpenShmem.

**Since**

Version 1.4.

**E.15.2.9 enum OTF2\_ThumbnailType\_enum**

Type of definitions used as metric in an thumbnail.

**Since**

Version 1.2

**Enumerator:**

**OTF2\_THUMBNAIL\_TYPE\_REGION** The referenced definitions are of type [Region](#).

**OTF2\_THUMBNAIL\_TYPE\_METRIC** The referenced definitions are of type [MetricMember](#).

**OTF2\_THUMBNAIL\_TYPE\_ATTRIBUTES** The referenced definitions are of type [Attribute](#).

**E.15.2.10 enum OTF2\_Type\_enum**

OTF2 basic data types.

**Since**

Version 1.0

**Enumerator:**

**OTF2\_TYPE\_NONE** Undefined type. Type category: None

**OTF2\_TYPE\_UINT8** Unsigned 8-bit integer. Type category: Integer

**OTF2\_TYPE\_UINT16** Unsigned 16-bit integer. Type category: Integer

**OTF2\_TYPE\_UINT32** Unsigned 32-bit integer. Type category: Integer

**OTF2\_TYPE\_UINT64** Unsigned 64-bit integer. Type category: Integer

**OTF2\_TYPE\_INT8** Signed 8-bit integer. Type category: Integer

**OTF2\_TYPE\_INT16** Signed 16-bit integer. Type category: Integer

**OTF2\_TYPE\_INT32** Signed 32-bit integer. Type category: Integer

**OTF2\_TYPE\_INT64** Signed 64-bit integer. Type category: Integer

**OTF2\_TYPE\_FLOAT** 32-bit floating point value Type category: Floating point

**OTF2\_TYPE\_DOUBLE** 64-bit floating point value Type category: Floating point



## E.16 otf2/OTF2\_GlobalDefReader.h File Reference

---

**OTF2\_TYPE\_STRING** Mapping of *String* identifiers. Type category: Definition reference

**OTF2\_TYPE\_ATTRIBUTE** Mapping of *Attribute* identifiers. Type category: Definition reference

**OTF2\_TYPE\_LOCATION** Mapping of *Location* identifiers. Type category: Definition reference

**OTF2\_TYPE\_REGION** Mapping of *Region* identifiers. Type category: Definition reference

**OTF2\_TYPE\_GROUP** Mapping of *Group* identifiers. Type category: Definition reference

**OTF2\_TYPE\_METRIC** Mapping of *Metric* identifiers. Type category: Definition reference

**OTF2\_TYPE\_COMM** Mapping of *Comm* identifiers. Type category: Definition reference

**OTF2\_TYPE\_PARAMETER** Mapping of *Parameter* identifiers. Type category: Definition reference

**OTF2\_TYPE\_RMA\_WIN** Mapping of *RmaWin* identifiers.

**Since**

Version 1.2.

Type category: Definition reference

## E.16 otf2/OTF2\_GlobalDefReader.h File Reference

This is the definition reader.

```
#include <stddef.h>
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_GlobalDefReaderCallbacks.h>
```

### Functions

- **OTF2\_ErrorCode OTF2\_GlobalDefReader\_ReadDefinitions** (OTF2\_GlobalDefReader \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)

*Reads the given number of records from the global definition reader.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReader\\_SetCallbacks](#) ([OTF2\\_GlobalDefReader](#) \*reader, const [OTF2\\_GlobalDefReaderCallbacks](#) \*callbacks, void \*userData)

*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### E.16.1 Detailed Description

This is the definition reader.

### E.16.2 Function Documentation

**E.16.2.1** [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReader\\_ReadDefinitions](#) (  
[OTF2\\_GlobalDefReader](#) \* reader, uint64\_t recordsToRead, uint64\_t \*  
recordsRead )

Reads the given number of records from the global definition reader.

#### Parameters

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.
out	<i>recordsRead</i>	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given recordsToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finished his job by checking recordsRead < recordsToRead.

#### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.16.2.2** [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReader\\_SetCallbacks](#) ( [OTF2\\_GlobalDefReader](#) \* reader, const [OTF2\\_GlobalDefReaderCallbacks](#) \*  
callbacks, void \* userData )

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function.

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

### Parameters

<i>reader</i>	This given reader object will be setted up with new callback functions.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#">OTF2_GlobalDefReaderCallbacks_New</a> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

This defines the callbacks for the global definition reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
```

### Typedefs

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Attribute](#))(void \*userData, [OTF2\\_AttributeRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) description, [OTF2\\_Type](#) type)  
*Function pointer definition for the callback which is triggered by a [Attribute](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Callpath](#))(void \*userData, [OTF2\\_CallpathRef](#) self, [OTF2\\_CallpathRef](#) parent, [OTF2\\_RegionRef](#) region)  
*Function pointer definition for the callback which is triggered by a [Callpath](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Callsite](#))(void \*userData, [OTF2\\_CallsiteRef](#) self, [OTF2\\_StringRef](#) sourceFile, uint32\_t lineNumber, [OTF2\\_RegionRef](#) enteredRegion, [OTF2\\_RegionRef](#) leftRegion)  
*Function pointer definition for the callback which is triggered by a [Callsite](#) definition record.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_CartCoordinate](#))(void \*userData, [OTF2\\_CartTopologyRef](#) cartTopology, uint32\_t rank, uint8\_t numberOfDimensions, const uint32\_t \*coordinates)  
*Function pointer definition for the callback which is triggered by a [CartCoordinate](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_CartDimension](#))(void \*userData, [OTF2\\_CartDimensionRef](#) self, [OTF2\\_StringRef](#) name, uint32\_t size, [OTF2\\_CartPeriodicity](#) cartPeriodicity)  
*Function pointer definition for the callback which is triggered by a [CartDimension](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_CartTopology](#))(void \*userData, [OTF2\\_CartTopologyRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_CommRef](#) communicator, uint8\_t numberOfDimensions, const [OTF2\\_CartDimensionRef](#) \*cartDimensions)  
*Function pointer definition for the callback which is triggered by a [CartTopology](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_ClockProperties](#))(void \*userData, uint64\_t timerResolution, uint64\_t globalOffset, uint64\_t traceLength)  
*Function pointer definition for the callback which is triggered by a [ClockProperties](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Comm](#))(void \*userData, [OTF2\\_CommRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_GroupRef](#) group, [OTF2\\_CommRef](#) parent)  
*Function pointer definition for the callback which is triggered by a [Comm](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Group](#))(void \*userData, [OTF2\\_GroupRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_GroupType](#) groupType, [OTF2\\_Paradigm](#) paradigm, [OTF2\\_GroupFlag](#) groupFlags, uint32\_t numberOfMembers, const uint64\_t \*members)  
*Function pointer definition for the callback which is triggered by a [Group](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Location](#))(void \*userData, [OTF2\\_LocationRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_LocationType](#) locationType, uint64\_t numberOfEvents, [OTF2\\_LocationGroupRef](#) locationGroup)  
*Function pointer definition for the callback which is triggered by a [Location](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_LocationGroup](#))(void \*userData, [OTF2\\_LocationGroupRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_LocationGroupType](#) locationGroupType, [OTF2\\_SystemTreeNodeRef](#) systemTreeParent)

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

*Function pointer definition for the callback which is triggered by a [Location-Group](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_LocationGroupProperty](#))(void \*userData, [OTF2\\_LocationGroupRef](#) locationGroup, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

*Function pointer definition for the callback which is triggered by a [Location-GroupProperty](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_LocationProperty](#))(void \*userData, [OTF2\\_LocationRef](#) location, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

*Function pointer definition for the callback which is triggered by a [Location-Property](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_MetricClass](#))(void \*userData, [OTF2\\_MetricRef](#) self, uint8\_t numberOfMetrics, const [OTF2\\_MetricMemberRef](#) \*metricMembers, [OTF2\\_MetricOccurrence](#) metricOccurrence, [OTF2\\_RecorderKind](#) recorderKind)

*Function pointer definition for the callback which is triggered by a [MetricClass](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_MetricClassRecorder](#))(void \*userData, [OTF2\\_MetricRef](#) metricClass, [OTF2\\_LocationRef](#) recorder)

*Function pointer definition for the callback which is triggered by a [MetricClass-Recorder](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_MetricInstance](#))(void \*userData, [OTF2\\_MetricRef](#) self, [OTF2\\_MetricRef](#) metricClass, [OTF2\\_LocationRef](#) recorder, [OTF2\\_MetricScope](#) metricScope, uint64\_t scope)

*Function pointer definition for the callback which is triggered by a [MetricInstance](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_MetricMember](#))(void \*userData, [OTF2\\_MetricMemberRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) description, [OTF2\\_MetricType](#) metricType, [OTF2\\_MetricMode](#) metricMode, [OTF2\\_Type](#) valueType, [OTF2\\_MetricBase](#) metricBase, int64\_t exponent, [OTF2\\_StringRef](#) unit)

*Function pointer definition for the callback which is triggered by a [MetricMember](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Parameter](#))(void \*userData, [OTF2\\_ParameterRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_ParameterType](#) parameterType)

*Function pointer definition for the callback which is triggered by a [Parameter](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Region](#))(void \*userData, [OTF2\\_RegionRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#)

---

## APPENDIX E. FILE DOCUMENTATION

---

canonicalName, [OTF2\\_StringRef](#) description, [OTF2\\_RegionRole](#) regionRole, [OTF2\\_Paradigm](#) paradigm, [OTF2\\_RegionFlag](#) regionFlags, [OTF2\\_StringRef](#) sourceFile, uint32\_t beginLineNumber, uint32\_t endLineNumber)

*Function pointer definition for the callback which is triggered by a [Region](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_RmaWin](#))(void \*userData, [OTF2\\_RmaWinRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_CommRef](#) comm)

*Function pointer definition for the callback which is triggered by a [RmaWin](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_String](#))(void \*userData, [OTF2\\_StringRef](#) self, const char \*string)

*Function pointer definition for the callback which is triggered by a [String](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_SystemTreeNode](#))(void \*userData, [OTF2\\_SystemTreeNodeRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) className, [OTF2\\_SystemTreeNodeRef](#) parent)

*Function pointer definition for the callback which is triggered by a [SystemTreeNode](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_SystemTreeNodeDomain](#))(void \*userData, [OTF2\\_SystemTreeNodeRef](#) systemTreeNode, [OTF2\\_SystemTreeDomain](#) systemTreeDomain)

*Function pointer definition for the callback which is triggered by a [SystemTreeNodeDomain](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_SystemTreeNodeProperty](#))(void \*userData, [OTF2\\_SystemTreeNodeRef](#) systemTreeNode, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

*Function pointer definition for the callback which is triggered by a [SystemTreeNodeProperty](#) definition record.*

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalDefReaderCallback\\_Unknown](#))(void \*userData)

*Function pointer definition for the callback which is triggered by an unknown definition record.*

- typedef struct [OTF2\\_GlobalDefReaderCallbacks\\_struct](#) [OTF2\\_GlobalDefReaderCallbacks](#)

*Opaque struct which holds all global definition record callbacks.*

### Functions

- void [OTF2\\_GlobalDefReaderCallbacks\\_Clear](#)([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks)

*Clears a struct for the global definition callbacks.*

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

- `void OTF2_GlobalDefReaderCallbacks_Delete (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks)`  
*Deallocates a struct for the global definition callbacks.*
- `OTF2_GlobalDefReaderCallbacks * OTF2_GlobalDefReaderCallbacks_New (void)`  
*Allocates a new struct for the global definition callbacks.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetAttributeCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Attribute attributeCallback)`  
*Registers the callback for the *Attribute* definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCallpathCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Callpath callpathCallback)`  
*Registers the callback for the *Callpath* definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCallsiteCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Callsite callsiteCallback)`  
*Registers the callback for the *Callsite* definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCartCoordinateCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_CartCoordinate cartCoordinateCallback)`  
*Registers the callback for the *CartCoordinate* definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCartDimensionCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_CartDimension cartDimensionCallback)`  
*Registers the callback for the *CartDimension* definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCartTopologyCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_CartTopology cartTopologyCallback)`  
*Registers the callback for the *CartTopology* definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetClockPropertiesCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_ClockProperties clockPropertiesCallback)`  
*Registers the callback for the *ClockProperties* definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetCommCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Comm commCallback)`  
*Registers the callback for the *Comm* definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetGroupCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Group groupCallback)`



---

## APPENDIX E. FILE DOCUMENTATION

---

*Registers the callback for the [Group](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetLocationCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_Location](#) locationCallback)

*Registers the callback for the [Location](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetLocationGroupCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_LocationGroup](#) locationGroupCallback)

*Registers the callback for the [LocationGroup](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetLocationGroupPropertyCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_LocationGroupProperty](#) locationGroupPropertyCallback)

*Registers the callback for the [LocationGroupProperty](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetLocationPropertyCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_LocationProperty](#) locationPropertyCallback)

*Registers the callback for the [LocationProperty](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetMetricClassCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_MetricClass](#) metricClassCallback)

*Registers the callback for the [MetricClass](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetMetricClassRecorderCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_MetricClassRecorder](#) metricClassRecorderCallback)

*Registers the callback for the [MetricClassRecorder](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetMetricInstanceCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_MetricInstance](#) metricInstanceCallback)

*Registers the callback for the [MetricInstance](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetMetricMemberCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_MetricMember](#) metricMemberCallback)

*Registers the callback for the [MetricMember](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetParameterCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_Parameter](#) parameterCallback)

*Registers the callback for the [Parameter](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetRegionCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_Region](#) regionCallback)

*Registers the callback for the [Region](#) definition.*



## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetRmaWinCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_RmaWin](#) rmaWinCallback)  
*Registers the callback for the [RmaWin](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetStringCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_String](#) stringCallback)  
*Registers the callback for the [String](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetSystemTreeNodeCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_SystemTreeNode](#) systemTreeNodeCallback)  
*Registers the callback for the [SystemTreeNode](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetSystemTreeNodeDomainCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_SystemTreeNodeDomain](#) systemTreeNodeDomainCallback)  
*Registers the callback for the [SystemTreeNodeDomain](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetSystemTreeNodePropertyCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_SystemTreeNodeProperty](#) systemTreeNodePropertyCallback)  
*Registers the callback for the [SystemTreeNodeProperty](#) definition.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefReaderCallbacks\\_SetUnknownCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_Unknown](#) unknownCallback)  
*Registers the callback for an unknown definition.*

### E.17.1 Detailed Description

This defines the callbacks for the global definition reader.

#### Source Template:

*templates/OTF2\_GlobalDefReaderCallbacks.tmpl.h*

### E.17.2 Typedef Documentation

- E.17.2.1** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_Attribute)(void *userData, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_Type type)`

Function pointer definition for the callback which is triggered by a [Attribute](#) definition record.

The attribute definition.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">Attribute</a> definition.
<i>name</i>	Name of the attribute. References a <a href="#">String</a> definition.
<i>description</i>	Description of the attribute. References a <a href="#">String</a> definition. Since version 1.4.
<i>type</i>	Type of the attribute value.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.2** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
Callpath)(void *userData, OTF2_CallpathRef self, OTF2_CallpathRef  
parent, OTF2_RegionRef region)`

Function pointer definition for the callback which is triggered by a [Callpath](#) definition record.

The callpath definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">Callpath</a> definition.
<i>parent</i>	The parent of this callpath. References a <a href="#">Callpath</a> definition.
<i>region</i>	The region of this callpath. References a <a href="#">Region</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

**E.17.2.3** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
Callsite)(void *userData, OTF2_CallsiteRef self, OTF2_StringRef  
sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion,  
OTF2_RegionRef leftRegion)`

Function pointer definition for the callback which is triggered by a [Callsite](#) definition record.

The callsite definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">Callsite</a> definition.
<i>sourceFile</i>	The source file where this call was made. References a <a href="#">String</a> 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 <a href="#">Region</a> definition.
<i>leftRegion</i>	The region which made the call. References a <a href="#">Region</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.4** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
CartCoordinate)(void *userData, OTF2_CartTopologyRef cartTopology,  
uint32_t rank, uint8_t numberOfDimensions, const uint32_t *coordinates)`

Function pointer definition for the callback which is triggered by a [CartCoordinate](#) definition record.

Defines the coordinate of the location referenced by the given rank (w.r.t. the communicator associated to the topology) in the referenced topology.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>cartTopology</i>	Parent <a href="#">CartTopology</a> definition to which this one is a supplementary definition. References a <a href="#">CartTopology</a> definition.

## APPENDIX E. FILE DOCUMENTATION

---

<i>rank</i>	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
<i>numberOfDimensions</i>	Number of dimensions.
<i>coordinates</i>	Coordinates, indexed by dimension.

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.5** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_CartDimension)(void *userData, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_t size, OTF2_CartPeriodicity cartPeriodicity)`

Function pointer definition for the callback which is triggered by a [\*CartDimension\*](#) definition record.

Each dimension in a Cartesian topology is composed of a global id, a name, its size, and whether it is periodic or not.

### Parameters

<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalDefCallbacks</i></a> or <a href="#"><i>OTF2_GlobalDefReader_SetCallbacks</i></a> .
<i>self</i>	The unique identifier for this <a href="#"><i>CartDimension</i></a> definition.
<i>name</i>	The name of the cartesian topology dimension. References a <a href="#"><i>String</i></a> definition.
<i>size</i>	The size of the cartesian topology dimension.
<i>cartPeriodicity</i>	Periodicity of the cartesian topology dimension.

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

**E.17.2.6** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
CartTopology)(void *userData, OTF2_CartTopologyRef self,  
OTF2_StringRef name, OTF2_CommRef communicator, uint8_t  
numberOfDimensions, const OTF2_CartDimensionRef *cartDimensions)`

Function pointer definition for the callback which is triggered by a [CartTopology](#) definition record.

Each topology is described by a global id, a reference to its name, a reference to a communicator, the number of dimensions, and references to those dimensions. The topology type is defined by the paradigm of the group referenced by the associated communicator.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">CartTopology</a> definition.
<i>name</i>	The name of the topology. References a <a href="#">String</a> definition.
<i>communi- cator</i>	Communicator object used to create the topology. References a <a href="#">Comm</a> definition.
<i>num- berOfDi- mensions</i>	Number of dimensions.
<i>cartDimen- sions</i>	The dimensions of this topology. References a <a href="#">CartDimension</a> definition.

### Since

Version 1.3

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.7** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
ClockProperties)(void *userData, uint64_t timerResolution, uint64_t  
globalOffset, uint64_t traceLength)`

Function pointer definition for the callback which is triggered by a [ClockProperties](#) definition record.

Defines the timer resolution and time range of this trace. There will be no event with a timestamp less than `globalOffset`, and no event with timestamp greater than `(globalOffset + traceLength)`.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>timerResolution</i>	Ticks per seconds.
<i>globalOffset</i>	A timestamp smaller than all event timestamps.
<i>traceLength</i>	A timespan which includes the timespan between the smallest and greatest timestamp of all event timestamps.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.8** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
Comm)(void *userData, OTF2_CommRef self, OTF2_StringRef name,  
OTF2_GroupRef group, OTF2_CommRef parent)`

Function pointer definition for the callback which is triggered by a [Comm](#) definition record.

The communicator definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">Comm</a> 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 <a href="#">String</a> definition.
<i>group</i>	The describing MPI group of this MPI communicator The group needs to be of type <a href="#">OTF2_GROUP_TYPE_COMM_GROUP</a> or <a href="#">OTF2_GROUP_TYPE_COMM_SELF</a> . References a <a href="#">Group</a> definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <a href="#">OTF2_UNDEFINED_COMM</a> to indicate no parent. References a <a href="#">Comm</a> definition.

### Since

Version 1.0

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.9** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - Group)(void *userData, OTF2_GroupRef self, OTF2_StringRef name, OTF2_GroupType groupType, OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_t numberOfMembers, const uint64_t *members)`

Function pointer definition for the callback which is triggered by a [\*Group\*](#) definition record.

The group definition.

### Parameters

<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalDefCallbacks</i></a> or <a href="#"><i>OTF2_GlobalDefReader_SetCallbacks</i></a> .
<i>self</i>	The unique identifier for this <a href="#"><i>Group</i></a> definition.
<i>name</i>	Name of this group References a <a href="#"><i>String</i></a> definition.
<i>groupType</i>	The type of this group. Since version 1.2.
<i>paradigm</i>	The paradigm of this communication group. Since version 1.2.
<i>groupFlags</i>	Flags for this group. Since version 1.2.
<i>numberOfMembers</i>	The number of members in this group.
<i>members</i>	The identifiers of the group members.

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.10** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - Location)(void *userData, OTF2_LocationRef self, OTF2_StringRef name, OTF2_LocationType locationType, uint64_t numberOfEvents, OTF2_LocationGroupRef locationGroup)`

Function pointer definition for the callback which is triggered by a [\*Location\*](#) definition record.

---

## APPENDIX E. FILE DOCUMENTATION

---

The location definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">Location</a> definition.
<i>name</i>	Name of the location References a <a href="#">String</a> 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 <a href="#">Location-Group</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.11** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
LocationGroup)(void *userData, OTF2_LocationGroupRef self,  
OTF2_StringRef name, OTF2_LocationGroupType locationGroupType,  
OTF2_SystemTreeNodeRef systemTreeParent)`

Function pointer definition for the callback which is triggered by a [LocationGroup](#) definition record.

The location group definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">LocationGroup</a> definition.
<i>name</i>	Name of the group. References a <a href="#">String</a> definition.
<i>location-GroupType</i>	Type of this group.
<i>systemTreeParent</i>	Parent of this location group in the system tree. References a <a href="#">SystemTreeNode</a> definition.



## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.12** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - LocationGroupProperty)(void *userData, OTF2_LocationGroupRef locationGroup, OTF2_StringRef name, OTF2_StringRef value)`

Function pointer definition for the callback which is triggered by a [\*LocationGroup-Property\*](#) definition record.

An arbitrary key/value property for a [\*LocationGroup\*](#) definition.

### Parameters

<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalDefCallbacks</i></a> or <a href="#"><i>OTF2_GlobalDefReader_SetCallbacks</i></a> .
<i>location-Group</i>	Parent <a href="#"><i>LocationGroup</i></a> definition to which this one is a supplementary definition. References a <a href="#"><i>LocationGroup</i></a> definition.
<i>name</i>	Name of the property. References a <a href="#"><i>String</i></a> definition.
<i>value</i>	Property value. References a <a href="#"><i>String</i></a> definition.

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.13** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - LocationProperty)(void *userData, OTF2_LocationRef location, OTF2_StringRef name, OTF2_StringRef value)`

Function pointer definition for the callback which is triggered by a [\*LocationProperty\*](#) definition record.

An arbitrary key/value property for a [\*Location\*](#) definition.

### Parameters

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>location</i>	Parent <a href="#">Location</a> definition to which this one is a supplementary definition. References a <a href="#">Location</a> definition.
<i>name</i>	Name of the property. References a <a href="#">String</a> definition.
<i>value</i>	Property value. References a <a href="#">String</a> definition.

### Since

Version 1.3

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.14** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
MetricClass)(void *userData, OTF2_MetricRef self, uint8_t  
numberOfMetrics, const OTF2_MetricMemberRef *metricMembers,  
OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind  
recorderKind)`

Function pointer definition for the callback which is triggered by a [MetricClass](#) definition record.

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">MetricClass</a> definition.
<i>numberOfMetrics</i>	Number of metrics within the set.
<i>metricMembers</i>	List of metric members. References a <a href="#">MetricMember</a> definition.
<i>metricOccurrence</i>	Defines occurrence of a metric set.
<i>recorderKind</i>	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

### Since

Version 1.0

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.15** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - MetricClassRecorder)(void *userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)`

Function pointer definition for the callback which is triggered by a [\*MetricClassRecorder\*](#) definition record.

The metric class recorder definition.

### Parameters

<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalDefCallbacks</i></a> or <a href="#"><i>OTF2_GlobalDefReader_SetCallbacks</i></a> .
<i>metricClass</i>	Parent <a href="#"><i>MetricClass</i></a> definition to which this one is a supplementary definition. References a <a href="#"><i>MetricClass</i></a> definition.
<i>recorder</i>	The location which recorded the referenced metric class. References a <a href="#"><i>Location</i></a> definition.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.16** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - MetricInstance)(void *userData, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)`

Function pointer definition for the callback which is triggered by a [\*MetricInstance\*](#) definition record.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type [\*OTF2\\_METRIC\\_ASYNCHRONOUS\*](#).

### Parameters

---

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">MetricClass</a> definition.
<i>metricClass</i>	The instanced <a href="#">MetricClass</a> . This metric class must be of kind <a href="#">OTF2_RECORDER_KIND_ABSTRACT</a> . References a <a href="#">MetricClass</a> definition.
<i>recorder</i>	Recorder of the metric: location ID. References a <a href="#">Location</a> definition.
<i>metric-Scope</i>	Defines type of scope: location, location group, system tree node, or a generic group of locations.
<i>scope</i>	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.17** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - MetricMember)(void *userData, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2_StringRef unit)`

Function pointer definition for the callback which is triggered by a [MetricMember](#) definition record.

A metric is defined by a metric member definition. A metric member is always a member of a metric class. Therefore, a single metric is a special case of a metric class with only one member. It is not allowed to reference a metric member id in a metric event, but only metric class IDs.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">MetricMember</a> definition.
<i>name</i>	Name of the metric. References a <a href="#">String</a> definition.
<i>description</i>	Description of the metric. References a <a href="#">String</a> definition.
<i>metricType</i>	Metric type: PAPI, etc.
<i>metricMode</i>	Metric mode: accumulative, fix, relative, etc.

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

<i>valueType</i>	Type of the value. Only <a href="#">OTF2_TYPE_INT64</a> , <a href="#">OTF2_TYPE_UINT64</a> , and <a href="#">OTF2_TYPE_DOUBLE</a> are valid types. If this metric member is recorded in an <a href="#">Metric</a> event, than this type and the type in the event must match.
<i>metricBase</i>	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$ , to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be <a href="#">OTF2_BASE_BINARY</a> 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 <a href="#">String</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.18** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ -  
Parameter)(void *userData, OTF2_ParameterRef self,  
OTF2_StringRef name, OTF2_ParameterType parameterType)`

Function pointer definition for the callback which is triggered by a [Parameter](#) definition record.

The parameter definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">Parameter</a> definition.
<i>name</i>	Name of the parameter (variable name etc.) References a <a href="#">String</a> definition.
<i>parameter-Type</i>	Type of the parameter, <a href="#">OTF2_ParameterType</a> for possible types.

## Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.19** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ -  
Region)(void *userData, OTF2_RegionRef self, OTF2_StringRef  
name, OTF2_StringRef canonicalName, OTF2_StringRef description,  
OTF2_RegionRole regionRole, OTF2_Paradigm paradigm,  
OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t  
beginLineNumber, uint32_t endLineNumber)`

Function pointer definition for the callback which is triggered by a [\*Region\*](#) definition record.

The region definition.

## Parameters

<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalDefCallbacks</i></a> or <a href="#"><i>OTF2_GlobalDefReader_SetCallbacks</i></a> .
<i>self</i>	The unique identifier for this <a href="#"><i>Region</i></a> definition.
<i>name</i>	Name of the region (demangled name if available). References a <a href="#"><i>String</i></a> definition.
<i>canonical-Name</i>	Alternative name of the region (e.g. mangled name). References a <a href="#"><i>String</i></a> definition. Since version 1.1.
<i>description</i>	A more detailed description of this region. References a <a href="#"><i>String</i></a> 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 <a href="#"><i>String</i></a> definition.
<i>beginLineNumber</i>	Starting line number of this region in the source file.
<i>endLineNumber</i>	Ending line number of this region in the source file.

## Since

Version 1.0

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.20** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - RmaWin)(void *userData, OTF2_RmaWinRef self, OTF2_StringRef name, OTF2_CommRef comm)`

Function pointer definition for the callback which is triggered by a [RmaWin](#) definition record.

A window defines the communication context for any remote-memory access operation.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">RmaWin</a> definition.
<i>name</i>	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a <a href="#">String</a> definition.
<i>comm</i>	Communicator object used to create the window. References a <a href="#">Comm</a> definition.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.21** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - String)(void *userData, OTF2_StringRef self, const char *string)`

Function pointer definition for the callback which is triggered by a [String](#) definition record.

The string definitions.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
-----------------	---

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>self</i>	The unique identifier for this <a href="#">String</a> definition.
<i>string</i>	The string, null terminated.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.17.2.22** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
SystemTreeNode)(void *userData, OTF2_SystemTreeNodeRef  
self, OTF2_StringRef name, OTF2_StringRef className,  
OTF2_SystemTreeNodeRef parent)`

Function pointer definition for the callback which is triggered by a [SystemTreeNode](#) definition record.

The system tree node definition.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
<i>self</i>	The unique identifier for this <a href="#">SystemTreeNode</a> definition.
<i>name</i>	Free form instance name of this node. References a <a href="#">String</a> definition.
<i>className</i>	Free form class name of this node References a <a href="#">String</a> definition.
<i>parent</i>	Parent id of this node. May be <a href="#">OTF2_UNDEFINED_SYSTEM_TREE_NODE</a> to indicate that there is no parent. References a <a href="#">SystemTreeNode</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).



## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

**E.17.2.23** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
SystemTreeNodeDomain)(void *userData, OTF2_SystemTreeNodeRef  
systemTreeNode, OTF2_SystemTreeDomain systemTreeDomain)`

Function pointer definition for the callback which is triggered by a *SystemTreeNodeDomain* definition record.

The system tree node domain definition.

### Parameters

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>systemTreeDomain</i>	The domain in which the referenced <i>SystemTreeNode</i> operates in.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.17.2.24** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_  
SystemTreeNodeProperty)(void *userData,  
OTF2_SystemTreeNodeRef systemTreeNode, OTF2_StringRef name,  
OTF2_StringRef value)`

Function pointer definition for the callback which is triggered by a *SystemTreeNodeProperty* definition record.

An arbitrary key/value property for a *SystemTreeNode* definition.

### Parameters

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalDefCallbacks</i> or <i>OTF2_GlobalDefReader_SetCallbacks</i> .
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

**Since**

Version 1.2

**Returns**

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.2.25** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ -  
Unknown)(void *userData)`

Function pointer definition for the callback which is triggered by an unknown definition record.

**Parameters**

<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalDefCallbacks</i></a> or <a href="#"><i>OTF2_GlobalDefReader_SetCallbacks</i></a> .
-----------------	---

**Returns**

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.17.3 Function Documentation**

**E.17.3.1** `void OTF2_GlobalDefReaderCallbacks_Clear ( OTF2_ -  
GlobalDefReaderCallbacks * globalDefReaderCallbacks  
)`

Clears a struct for the global definition callbacks.

**Parameters**

<i>globalDef- Reader- Callbacks</i>	Handle to a struct previously allocated with <a href="#"><i>OTF2_ - GlobalDefReaderCallbacks_New</i></a> .
---	--

**E.17.3.2** `void OTF2_GlobalDefReaderCallbacks_Delete ( OTF2_ -  
GlobalDefReaderCallbacks * globalDefReaderCallbacks  
)`

Deallocates a struct for the global definition callbacks.

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Parameters

<i>globalDef-Reader-Callbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_GlobalDefReaderCallbacks_New</a> .
-----------------------------------	--

### E.17.3.3 OTF2\_GlobalDefReaderCallbacks\* OTF2\_GlobalDefReaderCallbacks\_New ( void )

Allocates a new struct for the global definition callbacks.

### Returns

A newly allocated struct of type [OTF2\\_GlobalDefReaderCallbacks](#).

### E.17.3.4 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetAttributeCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_Attribute *attributeCallback* )

Registers the callback for the [Attribute](#) definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>attribute-Callback</i>	Function which should be called for all <a href="#">Attribute</a> definitions.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid `defReaderCallbacks` argument

**E.17.3.5** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_SetCallpathCallback**  
**( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*,**  
**OTF2\_GlobalDefReaderCallback\_Callpath *callpathCallback* )**

Registers the callback for the *Callpath* definition.

**Parameters**

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>callpath-Callback</i>	Function which should be called for all <i>Callpath</i> definitions.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.17.3.6** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_SetCallsiteCallback**  
**( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*,**  
**OTF2\_GlobalDefReaderCallback\_Callsite *callsiteCallback* )**

Registers the callback for the *Callsite* definition.

**Parameters**

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>callsite-Callback</i>	Function which should be called for all <i>Callsite</i> definitions.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.17.3.7 OTF2\_StatusCode OTF2\_GlobalDefReaderCallbacks\_-SetCartCoordinateCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_ *CartCoordinate cartCoordinateCallback* )**

Registers the callback for the *CartCoordinate* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>cartCoordinateCallback</i>	Function which should be called for all <i>CartCoordinate</i> definitions.

### Since

Version 1.3

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.17.3.8 OTF2\_StatusCode OTF2\_GlobalDefReaderCallbacks\_-SetCartDimensionCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_ *CartDimension cartDimensionCallback* )**

Registers the callback for the *CartDimension* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>cartDimensionCallback</i>	Function which should be called for all <i>CartDimension</i> definitions.

**Since**

Version 1.3

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.17.3.9** `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetCartTopologyCallback ( OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_CartTopology cartTopologyCallback )`

Registers the callback for the *CartTopology* definition.

**Parameters**

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>cartTopologyCallback</i>	Function which should be called for all <i>CartTopology</i> definitions.

**Since**

Version 1.3

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.17.3.10** `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetClockPropertiesCallback ( OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_ClockProperties clockPropertiesCallback )`

Registers the callback for the *ClockProperties* definition.

**Parameters**

---

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>clockPropertiesCallback</i>	Function which should be called for all <i>ClockProperties</i> definitions.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.17.3.11** `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetCommCallback ( OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Comm commCallback )`

Registers the callback for the *Comm* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>commCallback</i>	Function which should be called for all <i>Comm</i> definitions.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.17.3.12** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_SetGroupCallback**  
( **OTF2\_GlobalDefReaderCallbacks** \* *globalDefReaderCallbacks*,  
**OTF2\_GlobalDefReaderCallback\_Group** *groupCallback* )

Registers the callback for the *Group* definition.

**Parameters**

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>groupCallback</i>	Function which should be called for all <i>Group</i> definitions.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.17.3.13** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_SetLocationCallback**  
( **OTF2\_GlobalDefReaderCallbacks** \* *globalDefReaderCallbacks*,  
**OTF2\_GlobalDefReaderCallback\_Location** *locationCallback* )

Registers the callback for the *Location* definition.

**Parameters**

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>location-Callback</i>	Function which should be called for all <i>Location</i> definitions.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful



## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.17.3.14** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_**  
**SetLocationGroupCallback ( OTF2\_GlobalDefReaderCallbacks \***  
***globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_**  
**LocationGroup *locationGroupCallback* )**

Registers the callback for the *LocationGroup* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>location-GroupCallback</i>	Function which should be called for all <i>LocationGroup</i> definitions.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.17.3.15** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_**  
**SetLocationGroupPropertyCallback ( OTF2\_GlobalDefReaderCallbacks**  
**\* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_**  
**LocationGroupProperty *locationGroupPropertyCallback***  
**)**

Registers the callback for the *LocationGroupProperty* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
-----------------------------------	---------------------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>location-GroupPropertyCallback</i>	Function which should be called for all <i>LocationGroupProperty</i> definitions.
---------------------------------------	---

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.17.3.16** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_-SetLocationPropertyCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_-LocationProperty *locationPropertyCallback* )**

Registers the callback for the *LocationProperty* definition.

### Parameters

<i>globalDefReaderCallbacks</i>	Struct for all callbacks.
<i>locationPropertyCallback</i>	Function which should be called for all <i>LocationProperty</i> definitions.

### Since

Version 1.3

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

**E.17.3.17** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_SetMetricClassCallback**  
( **OTF2\_GlobalDefReaderCallbacks** \* *globalDefReaderCallbacks*,  
**OTF2\_GlobalDefReaderCallback\_MetricClass** *metricClassCallback* )

Registers the callback for the *MetricClass* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metric-ClassCallback</i>	Function which should be called for all <i>MetricClass</i> definitions.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.17.3.18** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_-SetMetricClassRecorderCallback** ( **OTF2\_GlobalDefReaderCallbacks** \* *globalDefReaderCallbacks*, **OTF2\_GlobalDefReaderCallback\_-MetricClassRecorder** *metricClassRecorderCallback* )

Registers the callback for the *MetricClassRecorder* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metric-ClassRecorder-Callback</i>	Function which should be called for all <i>MetricClassRecorder</i> definitions.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.17.3.19** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_-SetMetricInstanceCallback** ( **OTF2\_GlobalDefReaderCallbacks \*** *globalDefReaderCallbacks*, **OTF2\_GlobalDefReaderCallback\_-MetricInstance** *metricInstanceCallback* )

Registers the callback for the *MetricInstance* definition.

**Parameters**

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metricInstanceCallback</i>	Function which should be called for all <i>MetricInstance</i> definitions.

**Since**

Version 1.0

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.17.3.20** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_-SetMetricMemberCallback** ( **OTF2\_GlobalDefReaderCallbacks \*** *globalDefReaderCallbacks*, **OTF2\_GlobalDefReaderCallback\_-MetricMember** *metricMemberCallback* )

Registers the callback for the *MetricMember* definition.

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>metricMemberCallback</i>	Function which should be called for all <i>MetricMember</i> definitions.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.17.3.21** `OTF2_StatusCode OTF2_GlobalDefReaderCallbacks_SetParameterCallback ( OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Parameter parameterCallback )`

Registers the callback for the *Parameter* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>parameter-Callback</i>	Function which should be called for all <i>Parameter</i> definitions.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

### E.17.3.22 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetRegionCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_Region *regionCallback* )

Registers the callback for the *Region* definition.

#### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>regionCallback</i>	Function which should be called for all <i>Region</i> definitions.

#### Since

Version 1.0

#### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

### E.17.3.23 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetRmaWinCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_RmaWin *rmaWinCallback* )

Registers the callback for the *RmaWin* definition.

#### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>rmaWin-Callback</i>	Function which should be called for all <i>RmaWin</i> definitions.

#### Since

Version 1.2

#### Returns

*OTF2\_SUCCESS* if successful

## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.17.3.24** **OTF2\_StatusCode** **OTF2\_GlobalDefReaderCallbacks\_SetStringCallback**  
( **OTF2\_GlobalDefReaderCallbacks** \* *globalDefReaderCallbacks*,  
**OTF2\_GlobalDefReaderCallback\_String** *stringCallback* )

Registers the callback for the *String* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>stringCallback</i>	Function which should be called for all <i>String</i> definitions.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.17.3.25** **OTF2\_StatusCode** **OTF2\_GlobalDefReaderCallbacks\_SetSystemTreeNodeCallback** ( **OTF2\_GlobalDefReaderCallbacks** \* *globalDefReaderCallbacks*, **OTF2\_GlobalDefReaderCallback\_SystemTreeNode** *systemTreeNodeCallback* )

Registers the callback for the *SystemTreeNode* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>systemTreeNodeCallback</i>	Function which should be called for all <i>SystemTreeNode</i> definitions.

**Since**

Version 1.0

**Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.17.3.26** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_-SetSystemTreeNodeDomainCallback** ( **OTF2\_GlobalDefReaderCallbacks** \* **globalDefReaderCallbacks**, **OTF2\_GlobalDefReaderCallback\_-SystemTreeNodeDomain** **systemTreeNodeDomainCallback** )

Registers the callback for the *SystemTreeNodeDomain* definition.

**Parameters**

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>systemTreeNodeDomainCallback</i>	Function which should be called for all <i>SystemTreeNodeDomain</i> definitions.

**Since**

Version 1.2

**Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument



## E.17 otf2/OTF2\_GlobalDefReaderCallbacks.h File Reference

---

**E.17.3.27** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_**  
**SetSystemTreeNodePropertyCallback** ( **OTF2\_GlobalDefReaderCallbacks**  
**\*** *globalDefReaderCallbacks*, **OTF2\_GlobalDefReaderCallback\_**  
**SystemTreeNodeProperty** *systemTreeNodePropertyCallback*  
**)**

Registers the callback for the *SystemTreeNodeProperty* definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>systemTreeNodePropertyCallback</i>	Function which should be called for all <i>SystemTreeNodeProperty</i> definitions.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.17.3.28** **OTF2\_ErrorCode** **OTF2\_GlobalDefReaderCallbacks\_SetUnknownCallback**  
( **OTF2\_GlobalDefReaderCallbacks** \* *globalDefReaderCallbacks*,  
**OTF2\_GlobalDefReaderCallback\_Unknown** *unknownCallback* )

Registers the callback for an unknown definition.

### Parameters

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	Function which should be called for all Unknown definitions.

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid defReaderCallbacks argument

**E.18 otF2/OTF2\_GlobalDefWriter.h File Reference**

This layer always writes globally defined OTF2 definition records and is used to write either the global definitions in addition to local definitions or write all definitions as globally valid in combination with OTF2\_GlobalEventWriter. Global definitions are stored in one global definition file, which makes it nearly impossible to write them in a distributed manner. It is therefore only allowed to get such a writer from an OTF2\_Archive which is the master in the collective context.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
```

**Typedefs**

- typedef struct OTF2\_GlobalDefWriter\_struct [\*OTF2\\_GlobalDefWriter\*](#)  
*Typedef of the struct which keeps all necessary information of a global definition writer. Can be used to reference these structs from external.*

**Functions**

- [\*OTF2\\_ErrorCode OTF2\\_GlobalDefWriter\\_GetNumberOfDefinitions \(OTF2\\_GlobalDefWriter \\*writerHandle, uint64\\_t \\*numberOfDefinitions\)\*](#)  
*Returns the current number of written definitions of a global definition writer.*
- [\*OTF2\\_ErrorCode OTF2\\_GlobalDefWriter\\_GetNumberOfLocations \(OTF2\\_GlobalDefWriter \\*writerHandle, uint64\\_t \\*numberOfLocations\)\*](#)  
*Returns the current number of written location definitions of a global definition writer.*
- [\*OTF2\\_ErrorCode OTF2\\_GlobalDefWriter\\_WriteAttribute \(OTF2\\_GlobalDefWriter \\*writerHandle, OTF2\\_AttributeRef self, OTF2\\_StringRef name, OTF2\\_StringRef description, OTF2\\_Type type\)\*](#)  
*Writes a [\*Attribute\*](#) definition record into the GlobalDefWriter.*
- [\*OTF2\\_ErrorCode OTF2\\_GlobalDefWriter\\_WriteCallpath \(OTF2\\_GlobalDefWriter \\*writerHandle, OTF2\\_CallpathRef self, OTF2\\_CallpathRef parent, OTF2\\_RegionRef region\)\*](#)

## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

*Writes a [Callpath](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteCallsite](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_CallsiteRef](#) self, [OTF2\\_StringRef](#) sourceFile, uint32\_t lineNumber, [OTF2\\_RegionRef](#) enteredRegion, [OTF2\\_RegionRef](#) leftRegion)

*Writes a [Callsite](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteCartCoordinate](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_CartTopologyRef](#) cartTopology, uint32\_t rank, uint8\_t numberOfDimensions, const uint32\_t \*coordinates)

*Writes a [CartCoordinate](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteCartDimension](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_CartDimensionRef](#) self, [OTF2\\_StringRef](#) name, uint32\_t size, [OTF2\\_CartPeriodicity](#) cartPeriodicity)

*Writes a [CartDimension](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteCartTopology](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_CartTopologyRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_CommRef](#) communicator, uint8\_t numberOfDimensions, const [OTF2\\_CartDimensionRef](#) \*cartDimensions)

*Writes a [CartTopology](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteClockProperties](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, uint64\_t timerResolution, uint64\_t globalOffset, uint64\_t traceLength)

*Writes a [ClockProperties](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteComm](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_CommRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_GroupRef](#) group, [OTF2\\_CommRef](#) parent)

*Writes a [Comm](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteGroup](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_GroupRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_GroupType](#) groupType, [OTF2\\_Paradigm](#) paradigm, [OTF2\\_GroupFlag](#) groupFlags, uint32\_t numberOfMembers, const uint64\_t \*members)

*Writes a [Group](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteLocation](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_LocationRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_LocationType](#) locationType, uint64\_t numberOfEvents, [OTF2\\_LocationGroupRef](#) locationGroup)

*Writes a [Location](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteLocationGroup](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_LocationGroupRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_LocationGroupType](#) locationGroupType, [OTF2\\_SystemTreeNodeRef](#) systemTreeParent)

---

## APPENDIX E. FILE DOCUMENTATION

---

Writes a *LocationGroup* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocationGroupProperty (OTF2_GlobalDefWriter *writerHandle, OTF2_LocationGroupRef locationGroup, OTF2_StringRef name, OTF2_StringRef value)`

Writes a *LocationGroupProperty* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteLocationProperty (OTF2_GlobalDefWriter *writerHandle, OTF2_LocationRef location, OTF2_StringRef name, OTF2_StringRef value)`

Writes a *LocationProperty* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricClass (OTF2_GlobalDefWriter *writerHandle, OTF2_MetricRef self, uint8_t numberOfMetrics, const OTF2_MetricMemberRef *metricMembers, OTF2_MetricOccurrence metricOccurrence, OTF2_RecorderKind recorderKind)`

Writes a *MetricClass* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricClassRecorder (OTF2_GlobalDefWriter *writerHandle, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)`

Writes a *MetricClassRecorder* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricInstance (OTF2_GlobalDefWriter *writerHandle, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)`

Writes a *MetricInstance* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricMember (OTF2_GlobalDefWriter *writerHandle, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2_StringRef unit)`

Writes a *MetricMember* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteParameter (OTF2_GlobalDefWriter *writerHandle, OTF2_ParameterRef self, OTF2_StringRef name, OTF2_ParameterType parameterType)`

Writes a *Parameter* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteRegion (OTF2_GlobalDefWriter *writerHandle, OTF2_RegionRef self, OTF2_StringRef name, OTF2_StringRef canonicalName, OTF2_StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2_StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)`

Writes a *Region* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteRmaWin (OTF2_GlobalDefWriter *writerHandle, OTF2_RmaWinRef self, OTF2_StringRef name, OTF2_CommRef comm)`

## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

*Writes a [RmaWin](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteString](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_StringRef](#) self, const char \*string)

*Writes a [String](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteSystemTreeNode](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_SystemTreeNodeRef](#) self, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) className, [OTF2\\_SystemTreeNodeRef](#) parent)

*Writes a [SystemTreeNode](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteSystemTreeNodeDomain](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_SystemTreeNodeRef](#) systemTreeNode, [OTF2\\_SystemTreeDomain](#) systemTreeDomain)

*Writes a [SystemTreeNodeDomain](#) definition record into the GlobalDefWriter.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_WriteSystemTreeNodeProperty](#) ([OTF2\\_GlobalDefWriter](#) \*writerHandle, [OTF2\\_SystemTreeNodeRef](#) systemTreeNode, [OTF2\\_StringRef](#) name, [OTF2\\_StringRef](#) value)

*Writes a [SystemTreeNodeProperty](#) definition record into the GlobalDefWriter.*

### E.18.1 Detailed Description

This layer always writes globally defined OTF2 definition records and is used to write either the global definitions in addition to local definitions or write all definitions as globally valid in combination with [OTF2\\_GlobalEventWriter](#). Global definitions are stored in one global definition file, which makes it nearly impossible to write them in a distributed manner. It is therefore only allowed to get such a writer from an [OTF2\\_Archive](#) which is the master in the collective context.

#### Source Template:

*templates/OTF2\_GlobalDefWriter.templ.h*

### E.18.2 Function Documentation

#### E.18.2.1 [OTF2\\_ErrorCode](#) [OTF2\\_GlobalDefWriter\\_GetNumberOfDefinitions](#) ([OTF2\\_GlobalDefWriter](#) \* *writerHandle*, [uint64\\_t](#) \* *numberOfDefinitions* )

Returns the current number of written definitions of a global definition writer.

#### Parameters

	<i>writerHandle</i>	Handle to the global definition writer.
--	---------------------	---

## APPENDIX E. FILE DOCUMENTATION

out	<i>numberOfDefinitions</i>	Storage for the number of definitions.
-----	----------------------------	--

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.18.2.2** `OTF2_StatusCode OTF2_GlobalDefWriter_GetNumberOfLocations ( OTF2_GlobalDefWriter * writerHandle, uint64_t * numberOfLocations )`

Returns the current number of written location definitions of a global definition writer.

### Parameters

	<i>writerHandle</i>	Handle to the global definition writer.
out	<i>numberOfLocations</i>	Storage for the number of locations.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.18.2.3** `OTF2_StatusCode OTF2_GlobalDefWriter_WriteAttribute ( OTF2_GlobalDefWriter * writerHandle, OTF2_AttributeRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_Type type )`

Writes a [\*Attribute\*](#) definition record into the GlobalDefWriter.

The attribute definition.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#"><i>Attribute</i></a> definition.
<i>name</i>	Name of the attribute. References a <a href="#"><i>String</i></a> definition.
<i>description</i>	Description of the attribute. References a <a href="#"><i>String</i></a> definition. Since version 1.4.
<i>type</i>	Type of the attribute value.

## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.4** `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallpath (`  
`OTF2_GlobalDefWriter * writerHandle, OTF2_CallpathRef self,`  
`OTF2_CallpathRef parent, OTF2_RegionRef region )`

Writes a [Callpath](#) definition record into the GlobalDefWriter.

The callpath definition.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">Callpath</a> definition.
<i>parent</i>	The parent of this callpath. References a <a href="#">Callpath</a> definition.
<i>region</i>	The region of this callpath. References a <a href="#">Region</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.5** `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCallsite (`  
`OTF2_GlobalDefWriter * writerHandle, OTF2_CallsiteRef self,`  
`OTF2_StringRef sourceFile, uint32_t lineNumber, OTF2_RegionRef`  
`enteredRegion, OTF2_RegionRef leftRegion )`

Writes a [Callsite](#) definition record into the GlobalDefWriter.

The callsite definition.

### Parameters

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

## APPENDIX E. FILE DOCUMENTATION

---

<i>self</i>	The unique identifier for this <a href="#">Callsite</a> definition.
<i>sourceFile</i>	The source file where this call was made. References a <a href="#">String</a> 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 <a href="#">Region</a> definition.
<i>leftRegion</i>	The region which made the call. References a <a href="#">Region</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.6** `OTF2_StatusCode OTF2_GlobalDefWriter_WriteCartCoordinate (`  
`OTF2_GlobalDefWriter * writerHandle, OTF2_CartTopologyRef`  
`cartTopology, uint32_t rank, uint8_t numberOfDimensions, const uint32_t *`  
`coordinates )`

Writes a [CartCoordinate](#) definition record into the GlobalDefWriter.

Defines the coordinate of the location referenced by the given rank (w.r.t. the communicator associated to the topology) in the referenced topology.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>cartTopology</i>	Parent <a href="#">CartTopology</a> definition to which this one is a supplementary definition. References a <a href="#">CartTopology</a> definition.
<i>rank</i>	The rank w.r.t. the communicator associated to the topology referencing this coordinate.
<i>numberOfDimensions</i>	Number of dimensions.
<i>coordinates</i>	Coordinates, indexed by dimension.

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.



## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

**E.18.2.7** `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCartDimension ( OTF2_GlobalDefWriter * writerHandle, OTF2_CartDimensionRef self, OTF2_StringRef name, uint32_t size, OTF2_CartPeriodicity cartPeriodicity )`

Writes a [CartDimension](#) definition record into the GlobalDefWriter.

Each dimension in a Cartesian topology is composed of a global id, a name, its size, and whether it is periodic or not.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">CartDimension</a> definition.
<i>name</i>	The name of the cartesian topology dimension. References a <a href="#">String</a> definition.
<i>size</i>	The size of the cartesian topology dimension.
<i>cartPeriodicity</i>	Periodicity of the cartesian topology dimension.

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.8** `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteCartTopology ( OTF2_GlobalDefWriter * writerHandle, OTF2_CartTopologyRef self, OTF2_StringRef name, OTF2_CommRef communicator, uint8_t numberOfDimensions, const OTF2_CartDimensionRef * cartDimensions )`

Writes a [CartTopology](#) definition record into the GlobalDefWriter.

Each topology is described by a global id, a reference to its name, a reference to a communicator, the number of dimensions, and references to those dimensions. The topology type is defined by the paradigm of the group referenced by the associated communicator.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">CartTopology</a> definition.

## APPENDIX E. FILE DOCUMENTATION

---

<i>name</i>	The name of the topology. References a <a href="#">String</a> definition.
<i>communicator</i>	Communicator object used to create the topology. References a <a href="#">Comm</a> definition.
<i>numberOfDimensions</i>	Number of dimensions.
<i>cartDimensions</i>	The dimensions of this topology. References a <a href="#">CartDimension</a> definition.

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.9 OTF2\_StatusCode OTF2\_GlobalDefWriter\_WriteClockProperties (**  
**OTF2\_GlobalDefWriter \* writerHandle, uint64\_t timerResolution, uint64\_t**  
**globalOffset, uint64\_t traceLength )**

Writes a [ClockProperties](#) definition record into the GlobalDefWriter.

Defines the timer resolution and time range of this trace. There will be no event with a timestamp less than `globalOffset`, and no event with timestamp greater than `(globalOffset + traceLength)`.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>timerResolution</i>	Ticks per seconds.
<i>globalOffset</i>	A timestamp smaller than all event timestamps.
<i>traceLength</i>	A timespan which includes the timespan between the smallest and greatest timestamp of all event timestamps.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

**E.18.2.10** `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteComm (`  
    `OTF2_GlobalDefWriter * writerHandle, OTF2_CommRef self,`  
    `OTF2_StringRef name, OTF2_GroupRef group, OTF2_CommRef`  
    `parent )`

Writes a [Comm](#) definition record into the GlobalDefWriter.

The communicator definition.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">Comm</a> 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 <a href="#">String</a> definition.
<i>group</i>	The describing MPI group of this MPI communicator The group needs to be of type <a href="#">OTF2_GROUP_TYPE_COMM_GROUP</a> or <a href="#">OTF2_GROUP_TYPE_COMM_SELF</a> . References a <a href="#">Group</a> definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <a href="#">OTF2_UNDEFINED_COMM</a> to indicate no parent. References a <a href="#">Comm</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.11** `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteGroup (`  
    `OTF2_GlobalDefWriter * writerHandle, OTF2_GroupRef`  
    `self, OTF2_StringRef name, OTF2_GroupType groupType,`  
    `OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_t`  
    `numberOfMembers, const uint64_t * members )`

Writes a [Group](#) definition record into the GlobalDefWriter.

The group definition.

### Parameters

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

## APPENDIX E. FILE DOCUMENTATION

---

<i>self</i>	The unique identifier for this <a href="#">Group</a> definition.
<i>name</i>	Name of this group References a <a href="#">String</a> definition.
<i>groupType</i>	The type of this group. Since version 1.2.
<i>paradigm</i>	The paradigm of this communication group. Since version 1.2.
<i>groupFlags</i>	Flags for this group. Since version 1.2.
<i>numberOfMembers</i>	The number of members in this group.
<i>members</i>	The identifiers of the group members.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.12** `OTF2_StatusCode OTF2_GlobalDefWriter_WriteLocation (`  
    `OTF2_GlobalDefWriter * writerHandle, OTF2_LocationRef self,`  
    `OTF2_StringRef name, OTF2_LocationType locationType, uint64_t`  
    `numberOfEvents, OTF2_LocationGroupRef locationGroup )`

Writes a [Location](#) definition record into the GlobalDefWriter.

The location definition.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">Location</a> definition.
<i>name</i>	Name of the location References a <a href="#">String</a> definition.
<i>locationType</i>	Location type.
<i>numberOfEvents</i>	Number of events this location has recorded.
<i>locationGroup</i>	Location group which includes this location. References a <a href="#">Location-Group</a> definition.

### Since

Version 1.0

## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.18.2.13** **OTF2\_ErrorCode** OTF2\_GlobalDefWriter\_WriteLocationGroup (   
 OTF2\_GlobalDefWriter \* *writerHandle*, OTF2\_LocationGroupRef *self*, OTF2\_StringRef *name*, OTF2\_LocationGroupType *locationGroupType*, OTF2\_SystemTreeNodeRef *systemTreeParent* )

Writes a [\*LocationGroup\*](#) definition record into the GlobalDefWriter.

The location group definition.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#"><i>LocationGroup</i></a> definition.
<i>name</i>	Name of the group. References a <a href="#"><i>String</i></a> definition.
<i>locationGroupType</i>	Type of this group.
<i>systemTreeParent</i>	Parent of this location group in the system tree. References a <a href="#"><i>SystemTreeNode</i></a> definition.

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.18.2.14** **OTF2\_ErrorCode** OTF2\_GlobalDefWriter\_WriteLocationGroupProperty (   
 OTF2\_GlobalDefWriter \* *writerHandle*, OTF2\_LocationGroupRef *locationGroup*, OTF2\_StringRef *name*, OTF2\_StringRef *value* )

Writes a [\*LocationGroupProperty\*](#) definition record into the GlobalDefWriter.

An arbitrary key/value property for a [\*LocationGroup\*](#) definition.

### Parameters

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>location-Group</i>	Parent <a href="#">LocationGroup</a> definition to which this one is a supplementary definition. References a <a href="#">LocationGroup</a> definition.
<i>name</i>	Name of the property. References a <a href="#">String</a> definition.
<i>value</i>	Property value. References a <a href="#">String</a> definition.

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.15** **OTF2\_ErrorCode** **OTF2\_GlobalDefWriter\_WriteLocationProperty** (  
    **OTF2\_GlobalDefWriter** \* *writerHandle*, **OTF2\_LocationRef** *location*,  
    **OTF2\_StringRef** *name*, **OTF2\_StringRef** *value* )

Writes a [LocationProperty](#) definition record into the GlobalDefWriter.

An arbitrary key/value property for a [Location](#) definition.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>location</i>	Parent <a href="#">Location</a> definition to which this one is a supplementary definition. References a <a href="#">Location</a> definition.
<i>name</i>	Name of the property. References a <a href="#">String</a> definition.
<i>value</i>	Property value. References a <a href="#">String</a> definition.

### Since

Version 1.3

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

**E.18.2.16** **OTF2\_ErrorCode** **OTF2\_GlobalDefWriter\_WriteMetricClass** (  
    **OTF2\_GlobalDefWriter** \* *writerHandle*, **OTF2\_MetricRef** *self*, **uint8\_t**  
    *numberOfMetrics*, **const** **OTF2\_MetricMemberRef** \* *metricMembers*,  
    **OTF2\_MetricOccurrence** *metricOccurrence*, **OTF2\_RecorderKind**  
    *recorderKind* )

Writes a [MetricClass](#) definition record into the GlobalDefWriter.

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">MetricClass</a> definition.
<i>numberOfMetrics</i>	Number of metrics within the set.
<i>metricMembers</i>	List of metric members. References a <a href="#">MetricMember</a> definition.
<i>metricOccurrence</i>	Defines occurrence of a metric set.
<i>recorderKind</i>	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.17** **OTF2\_ErrorCode** **OTF2\_GlobalDefWriter\_WriteMetricClassRecorder** (  
    **OTF2\_GlobalDefWriter** \* *writerHandle*, **OTF2\_MetricRef** *metricClass*,  
    **OTF2\_LocationRef** *recorder* )

Writes a [MetricClassRecorder](#) definition record into the GlobalDefWriter.

The metric class recorder definition.

### Parameters

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>metricClass</i>	Parent <a href="#">MetricClass</a> definition to which this one is a supplementary definition. References a <a href="#">MetricClass</a> definition.
<i>recorder</i>	The location which recorded the referenced metric class. References a <a href="#">Location</a> definition.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.18** `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricInstance (`  
    `OTF2_GlobalDefWriter * writerHandle, OTF2_MetricRef self,`  
    `OTF2_MetricRef metricClass, OTF2_LocationRef recorder,`  
    `OTF2_MetricScope metricScope, uint64_t scope )`

Writes a [MetricInstance](#) definition record into the GlobalDefWriter.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type [OTF2\\_METRIC\\_ASYNCHRONOUS](#).

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">MetricClass</a> definition.
<i>metricClass</i>	The instanced <a href="#">MetricClass</a> . This metric class must be of kind <a href="#">OTF2_RECORDER_KIND_ABSTRACT</a> . References a <a href="#">MetricClass</a> definition.
<i>recorder</i>	Recorder of the metric: location ID. References a <a href="#">Location</a> definition.
<i>metricScope</i>	Defines type of scope: location, location group, system tree node, or a generic group of locations.
<i>scope</i>	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.



## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

**E.18.2.19** `OTF2_StatusCode 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 <a href="#">MetricMember</a> definition.
<i>name</i>	Name of the metric. References a <a href="#">String</a> definition.
<i>description</i>	Description of the metric. References a <a href="#">String</a> definition.
<i>metricType</i>	Metric type: PAPI, etc.
<i>metricMode</i>	Metric mode: accumulative, fix, relative, etc.
<i>valueType</i>	Type of the value. Only <a href="#">OTF2_TYPE_INT64</a> , <a href="#">OTF2_TYPE_UINT64</a> , and <a href="#">OTF2_TYPE_DOUBLE</a> are valid types. If this metric member is recorded in an <a href="#">Metric</a> event, then this type and the type in the event must match.
<i>metricBase</i>	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$ , to get the value in its base unit. For example, if the metric values come in as KiBi, then the base should be <a href="#">OTF2_BASE_BINARY</a> and the exponent 10. Then 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 <a href="#">String</a> definition.

Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.18.2.20** **OTF2\_ErrorCode** **OTF2\_GlobalDefWriter\_WriteParameter** (  
**OTF2\_GlobalDefWriter** \* *writerHandle*, **OTF2\_ParameterRef** *self*,  
**OTF2\_StringRef** *name*, **OTF2\_ParameterType** *parameterType* )

Writes a [\*Parameter\*](#) definition record into the GlobalDefWriter.

The parameter definition.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#"><i>Parameter</i></a> definition.
<i>name</i>	Name of the parameter (variable name etc.) References a <a href="#"><i>String</i></a> definition.
<i>parameterType</i>	Type of the parameter, <a href="#"><i>OTF2_ParameterType</i></a> for possible types.

### Since

Version 1.0

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.18.2.21** **OTF2\_ErrorCode** **OTF2\_GlobalDefWriter\_WriteRegion** (  
**OTF2\_GlobalDefWriter** \* *writerHandle*, **OTF2\_RegionRef**  
*self*, **OTF2\_StringRef** *name*, **OTF2\_StringRef** *canonicalName*,  
**OTF2\_StringRef** *description*, **OTF2\_RegionRole** *regionRole*,  
**OTF2\_Paradigm** *paradigm*, **OTF2\_RegionFlag** *regionFlags*,  
**OTF2\_StringRef** *sourceFile*, **uint32\_t** *beginLineNumber*, **uint32\_t**  
*endLineNumber* )

Writes a [\*Region\*](#) definition record into the GlobalDefWriter.

The region definition.

### Parameters

---

## E.18 of2/OTF2\_GlobalDefWriter.h File Reference

---

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">Region</a> definition.
<i>name</i>	Name of the region (demangled name if available). References a <a href="#">String</a> definition.
<i>canonicalName</i>	Alternative name of the region (e.g. mangled name). References a <a href="#">String</a> definition. Since version 1.1.
<i>description</i>	A more detailed description of this region. References a <a href="#">String</a> 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 <a href="#">String</a> definition.
<i>beginLineNumber</i>	Starting line number of this region in the source file.
<i>endLineNumber</i>	Ending line number of this region in the source file.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.18.2.22** **OTF2\_ErrorCode** **OTF2\_GlobalDefWriter\_WriteRmaWin** (  
    **OTF2\_GlobalDefWriter** \* *writerHandle*, **OTF2\_RmaWinRef** *self*,  
    **OTF2\_StringRef** *name*, **OTF2\_CommRef** *comm* )

Writes a [RmaWin](#) definition record into the GlobalDefWriter.

A window defines the communication context for any remote-memory access operation.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">RmaWin</a> definition.
<i>name</i>	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a <a href="#">String</a> definition.
<i>comm</i>	Communicator object used to create the window. References a <a href="#">Comm</a> definition.

**Since**

Version 1.2

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.18.2.23** **OTF2\_ErrorCode** **OTF2\_GlobalDefWriter\_WriteString** (  
    **OTF2\_GlobalDefWriter** \* *writerHandle*, **OTF2\_StringRef** *self*, **const**  
    **char** \* *string* )

Writes a [\*String\*](#) definition record into the GlobalDefWriter.

The string definitions.

**Parameters**

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#"><i>String</i></a> definition.
<i>string</i>	The string, null terminated.

**Since**

Version 1.0

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.18.2.24** **OTF2\_ErrorCode** **OTF2\_GlobalDefWriter\_WriteSystemTreeNode** (  
    **OTF2\_GlobalDefWriter** \* *writerHandle*, **OTF2\_SystemTreeNodeRef**  
    *self*, **OTF2\_StringRef** *name*, **OTF2\_StringRef** *className*,  
    **OTF2\_SystemTreeNodeRef** *parent* )

Writes a [\*SystemTreeNode\*](#) definition record into the GlobalDefWriter.

The system tree node definition.

**Parameters**

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#"><i>SystemTreeNode</i></a> definition.

## E.18 otf2/OTF2\_GlobalDefWriter.h File Reference

---

<i>name</i>	Free form instance name of this node. References a <i>String</i> definition.
<i>className</i>	Free form class name of this node References a <i>String</i> definition.
<i>parent</i>	Parent id of this node. May be <i>OTF2_UNDEFINED_SYSTEM_TREE_NODE</i> to indicate that there is no parent. References a <i>SystemTreeNode</i> definition.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.18.2.25** *OTF2\_ErrorCode* *OTF2\_GlobalDefWriter\_WriteSystemTreeNodeDomain* (  
*OTF2\_GlobalDefWriter* \* *writerHandle*, *OTF2\_SystemTreeNodeRef*  
*systemTreeNode*, *OTF2\_SystemTreeDomain* *systemTreeDomain* )

Writes a *SystemTreeNodeDomain* definition record into the GlobalDefWriter.

The system tree node domain definition.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>systemTreeDomain</i>	The domain in which the referenced <i>SystemTreeNode</i> operates in.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.18.2.26** `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty ( OTF2_GlobalDefWriter * writerHandle, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_StringRef name, OTF2_StringRef value )`

Writes a *SystemTreeNodeProperty* definition record into the GlobalDefWriter.

An arbitrary key/value property for a *SystemTreeNode* definition.

#### Parameters

<i>writerHandle</i>	The writer handle.
<i>systemTreeNode</i>	Parent <i>SystemTreeNode</i> definition to which this one is a supplementary definition. References a <i>SystemTreeNode</i> definition.
<i>name</i>	Name of the property. References a <i>String</i> definition.
<i>value</i>	Property value. References a <i>String</i> definition.

#### Since

Version 1.2

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## E.19 otf2/OTF2\_GlobalEvtReader.h File Reference

This is the global event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_EvtReader.h>
#include <otf2/OTF2_GlobalEvtReaderCallbacks.h>
```

#### Functions

- `OTF2_ErrorCode OTF2_GlobalEvtReader_HasEvent (OTF2_GlobalEvtReader *reader, int *flag)`  
*Has more events.*
- `OTF2_ErrorCode OTF2_GlobalEvtReader_ReadEvent (OTF2_GlobalEvtReader *reader)`  
*Triggers the callback for the next event record.*

## E.19 otf2/OTF2\_GlobalEvtReader.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReader\\_ReadEvents](#) ([OTF2\\_GlobalEvtReader](#) \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)  
*Reads the given number of records from the global event reader.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReader\\_SetCallbacks](#) ([OTF2\\_GlobalEvtReader](#) \*reader, const [OTF2\\_GlobalEvtReaderCallbacks](#) \*callbacks, void \*userData)

*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### E.19.1 Detailed Description

This is the global event reader. Used to read from multiple local event readers, and provide them in a timely ordered sequence.

### E.19.2 Function Documentation

#### E.19.2.1 [OTF2\\_ErrorCode OTF2\\_GlobalEvtReader\\_HasEvent](#) ([OTF2\\_GlobalEvtReader](#) \* reader, int \* flag )

Has more events.

##### Parameters

	<i>reader</i>	Global event reader handle.
out	<i>flag</i>	In case of success, the flag will be set to 1 when there is at least more more event to read. To 0 if not. Otherwise the value is undefined.

##### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

#### E.19.2.2 [OTF2\\_ErrorCode OTF2\\_GlobalEvtReader\\_ReadEvent](#) ([OTF2\\_GlobalEvtReader](#) \* reader )

Triggers the callback for the next event record.

##### Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
---------------	---

---

## APPENDIX E. FILE DOCUMENTATION

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.19.2.3** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReader\_ReadEvents** (  
    **OTF2\_GlobalEvtReader** \* *reader*, **uint64\_t** *recordsToRead*, **uint64\_t** \*  
    *recordsRead* )

Reads the given number of records from the global event reader.

### Parameters

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.
out	<i>recordsRead</i>	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given <i>recordsToRead</i> if there are no more records left in the trace. In this case the programmer can easily check that the reader has finished his job by checking <i>recordsRead</i> < <i>recordsToRead</i> .

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.19.2.4** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReader\_SetCallbacks** ( **OTF2\_GlobalEvtReader** \* *reader*, **const** **OTF2\_GlobalEvtReaderCallbacks** \* *callbacks*, **void** \* *userData* )

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

### Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#"><i>OTF2_GlobalEvtReaderCallbacks_New</i></a> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

This defines the callbacks for the global event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

### Typedefs

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalEvtReaderCallback\\_BufferFlush](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) time, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) stopTime)  
*Callback for the BufferFlush event record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalEvtReaderCallback\\_Enter](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) time, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RegionRef](#) region)  
*Callback for the Enter event record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalEvtReaderCallback\\_Leave](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) time, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_RegionRef](#) region)  
*Callback for the Leave event record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalEvtReaderCallback\\_MeasurementOnOff](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) time, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_MeasurementMode](#) measurementMode)  
*Callback for the MeasurementOnOff event record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalEvtReaderCallback\\_Metric](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) time, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_MetricRef](#) metric, uint8\_t numberOfMetrics, const [OTF2\\_Type](#) \*typeIDs, const [OTF2\\_MetricValue](#) \*metricValues)  
*Callback for the Metric event record.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiCollectiveBegin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList)

*Callback for the MpiCollectiveBegin event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiCollectiveEnd)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CollectiveOp collectiveOp, OTF2\_CommRef communicator, uint32\_t root, uint64\_t sizeSent, uint64\_t sizeReceived)

*Callback for the MpiCollectiveEnd event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiIrecv)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t sender, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID)

*Callback for the MpiIrecv event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiIrecvRequest)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)

*Callback for the MpiIrecvRequest event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiIsend)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID)

*Callback for the MpiIsend event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiIsendComplete)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)

*Callback for the MpiIsendComplete event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiRecv)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t sender, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)

*Callback for the MpiRecv event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiRequestCancelled)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)

*Callback for the MpiRequestCancelled event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiRequestTest)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)

*Callback for the MpiRequestTest event record.*

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_MpiSend)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)  
*Callback for the MpiSend event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpAcquireLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t lockID, uint32\_t acquisitionOrder)  
*Callback for the OmpAcquireLock event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpFork)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t numberOfRequestedThreads)  
*Callback for the OmpFork event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpJoin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList)  
*Callback for the OmpJoin event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpReleaseLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t lockID, uint32\_t acquisitionOrder)  
*Callback for the OmpReleaseLock event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpTaskComplete)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)  
*Callback for the OmpTaskComplete event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpTaskCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)  
*Callback for the OmpTaskCreate event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpTaskSwitch)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)  
*Callback for the OmpTaskSwitch event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ParameterInt)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, int64\_t value)  
*Callback for the ParameterInt event record.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ParameterString)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, OTF2\_StringRef string)  
*Callback for the ParameterString event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ParameterUnsignedInt)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, uint64\_t value)  
*Callback for the ParameterUnsignedInt event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaAcquireLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType)  
*Callback for the RmaAcquireLock event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaAtomic)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, OTF2\_RmaAtomicType type, uint64\_t bytesSent, uint64\_t bytesReceived, uint64\_t matchingId)  
*Callback for the RmaAtomic event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaCollectiveBegin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList)  
*Callback for the RmaCollectiveBegin event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaCollectiveEnd)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CollectiveOp collectiveOp, OTF2\_RmaSyncLevel syncLevel, OTF2\_RmaWinRef win, uint32\_t root, uint64\_t bytesSent, uint64\_t bytesReceived)  
*Callback for the RmaCollectiveEnd event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaGet)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t bytes, uint64\_t matchingId)  
*Callback for the RmaGet event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaGroupSync)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaSyncLevel syncLevel, OTF2\_RmaWinRef win, OTF2\_GroupRef group)  
*Callback for the RmaGroupSync event record.*

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteBlocking)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)

*Callback for the RmaOpCompleteBlocking event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteNonBlocking)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)

*Callback for the RmaOpCompleteNonBlocking event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteRemote)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)

*Callback for the RmaOpCompleteRemote event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaOpTest)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)

*Callback for the RmaOpTest event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaPut)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t bytes, uint64\_t matchingId)

*Callback for the RmaPut event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaReleaseLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId)

*Callback for the RmaReleaseLock event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaRequestLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType)

*Callback for the RmaRequestLock event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaSync)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, OTF2\_RmaSyncType syncType)

*Callback for the RmaSync event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaTryLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType)  
*Callback for the RmaTryLock event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaWaitChange)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)  
*Callback for the RmaWaitChange event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaWinCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)  
*Callback for the RmaWinCreate event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaWinDestroy)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)  
*Callback for the RmaWinDestroy event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadAcquireLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model, uint32\_t lockID, uint32\_t acquisitionOrder)  
*Callback for the ThreadAcquireLock event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadBegin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadContingent, uint64\_t sequenceCount)  
*Callback for the ThreadBegin event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadContingent, uint64\_t sequenceCount)  
*Callback for the ThreadCreate event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadEnd)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadContingent, uint64\_t sequenceCount)  
*Callback for the ThreadEnd event record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadFork)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model, uint32\_t numberOfRequestedThreads)

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

*Callback for the ThreadFork event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model)`

*Callback for the ThreadJoin event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

*Callback for the ThreadReleaseLock event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadTaskComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskComplete event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskCreate event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskSwitch event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadTeamBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam)`

*Callback for the ThreadTeamBegin event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadTeamEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam)`

*Callback for the ThreadTeamEnd event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ThreadWait)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

*Callback for the ThreadWait event record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList)`



---

## APPENDIX E. FILE DOCUMENTATION

---

*Callback for an unknown event record.*

- typedef struct OTF2\_GlobalEvtReaderCallbacks\_struct [OTF2\\_GlobalEvtReaderCallbacks](#)

*Opaque struct which holds all event record callbacks.*

### Functions

- void [OTF2\\_GlobalEvtReaderCallbacks\\_Clear](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks)  
*Clears a struct for the global event callbacks.*
- void [OTF2\\_GlobalEvtReaderCallbacks\\_Delete](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks)  
*Deallocates a struct for the global event callbacks.*
- [OTF2\\_GlobalEvtReaderCallbacks](#) \* [OTF2\\_GlobalEvtReaderCallbacks\\_New](#) (void)  
*Allocates a new struct for the event callbacks.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetBufferFlushCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_BufferFlush](#) bufferFlushCallback)  
*Registers the callback for the BufferFlush event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetEnterCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_Enter](#) enterCallback)  
*Registers the callback for the Enter event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetLeaveCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_Leave](#) leaveCallback)  
*Registers the callback for the Leave event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetMeasurementOnOffCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_MeasurementOnOff](#) measurementOnOffCallback)  
*Registers the callback for the MeasurementOnOff event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetMetricCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_Metric](#) metricCallback)  
*Registers the callback for the Metric event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiCollectiveBeginCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_MpiCollectiveBegin](#) mpiCollectiveBeginCallback)  
*Registers the callback for the MpiCollectiveBegin event.*



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiCollectiveEndCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiCollectiveEnd](#) mpiCollectiveEndCallback)  
*Registers the callback for the MpiCollectiveEnd event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiIrecvCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiIrecv](#) mpiIrecvCallback)  
*Registers the callback for the MpiIrecv event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiIrecvRequestCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiIrecvRequest](#) mpiIrecvRequestCallback)  
*Registers the callback for the MpiIrecvRequest event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiIsendCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiIsend](#) mpiIsendCallback)  
*Registers the callback for the MpiIsend event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiIsendCompleteCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiIsendComplete](#) mpiIsendCompleteCallback)  
*Registers the callback for the MpiIsendComplete event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiRecvCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiRecv](#) mpiRecvCallback)  
*Registers the callback for the MpiRecv event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiRequestCancelledCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiRequestCancelled](#) mpiRequestCancelledCallback)  
*Registers the callback for the MpiRequestCancelled event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiRequestTestCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiRequestTest](#) mpiRequestTestCallback)  
*Registers the callback for the MpiRequestTest event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetMpiSendCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-MpiSend](#) mpiSendCallback)  
*Registers the callback for the MpiSend event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetOmpAcquireLockCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-OmpAcquireLock](#) ompAcquireLockCallback)  
*Registers the callback for the OmpAcquireLock event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetOmpForkCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-OmpFork](#) ompForkCallback)  
*Registers the callback for the OmpFork event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetOmpJoinCallback](#) ([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-OmpJoin](#) ompJoinCallback)  
*Registers the callback for the OmpJoin event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetOmpReleaseLockCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-OmpReleaseLock](#) ompReleaseLockCallback)  
*Registers the callback for the OmpReleaseLock event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetOmpTaskCompleteCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-OmpTaskComplete](#) ompTaskCompleteCallback)  
*Registers the callback for the OmpTaskComplete event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetOmpTaskCreateCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-OmpTaskCreate](#) ompTaskCreateCallback)  
*Registers the callback for the OmpTaskCreate event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetOmpTaskSwitchCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-OmpTaskSwitch](#) ompTaskSwitchCallback)  
*Registers the callback for the OmpTaskSwitch event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetParameterIntCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-ParameterInt](#) parameterIntCallback)  
*Registers the callback for the ParameterInt event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetParameterStringCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-ParameterString](#) parameterStringCallback)  
*Registers the callback for the ParameterString event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetParameterUnsignedIntCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-ParameterUnsignedInt](#) parameterUnsignedIntCallback)  
*Registers the callback for the ParameterUnsignedInt event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaAcquireLockCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaAcquireLock](#) rmaAcquireLockCallback)  
*Registers the callback for the RmaAcquireLock event.*

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaAtomicCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaAtomic](#) rmaAtomicCallback)  
*Registers the callback for the RmaAtomic event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaCollectiveBeginCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaCollectiveBegin](#) rmaCollectiveBeginCallback)  
*Registers the callback for the RmaCollectiveBegin event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaCollectiveEndCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaCollectiveEnd](#) rmaCollectiveEndCallback)  
*Registers the callback for the RmaCollectiveEnd event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaGetCallback](#) ([OTF2\\_-GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaGet](#) rmaGetCallback)  
*Registers the callback for the RmaGet event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaGroupSyncCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaGroupSync](#) rmaGroupSyncCallback)  
*Registers the callback for the RmaGroupSync event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaOpCompleteBlockingCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaOpCompleteBlocking](#) rmaOpCompleteBlockingCallback)  
*Registers the callback for the RmaOpCompleteBlocking event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaOpCompleteNonBlockingCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaOpCompleteNonBlocking](#) rmaOpCompleteNonBlockingCallback)  
*Registers the callback for the RmaOpCompleteNonBlocking event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaOpCompleteRemoteCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaOpCompleteRemote](#) rmaOpCompleteRemoteCallback)  
*Registers the callback for the RmaOpCompleteRemote event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaOpTestCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaOpTest](#) rmaOpTestCallback)  
*Registers the callback for the RmaOpTest event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaPutCallback](#) ([OTF2\\_-GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_-RmaPut](#) rmaPutCallback)  
*Registers the callback for the RmaPut event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaReleaseLockCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-RmaReleaseLock](#) [rmaReleaseLockCallback](#))  
*Registers the callback for the RmaReleaseLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaRequestLockCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-RmaRequestLock](#) [rmaRequestLockCallback](#))  
*Registers the callback for the RmaRequestLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaSyncCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-RmaSync](#) [rmaSyncCallback](#))  
*Registers the callback for the RmaSync event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaTryLockCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-RmaTryLock](#) [rmaTryLockCallback](#))  
*Registers the callback for the RmaTryLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaWaitChangeCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-RmaWaitChange](#) [rmaWaitChangeCallback](#))  
*Registers the callback for the RmaWaitChange event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaWinCreateCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-RmaWinCreate](#) [rmaWinCreateCallback](#))  
*Registers the callback for the RmaWinCreate event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetRmaWinDestroyCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-RmaWinDestroy](#) [rmaWinDestroyCallback](#))  
*Registers the callback for the RmaWinDestroy event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadAcquireLockCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-ThreadAcquireLock](#) [threadAcquireLockCallback](#))  
*Registers the callback for the ThreadAcquireLock event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadBeginCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-ThreadBegin](#) [threadBeginCallback](#))  
*Registers the callback for the ThreadBegin event.*
- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadCreateCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*[globalEvtReaderCallbacks](#), [OTF2\\_GlobalEvtReaderCallback\\_-ThreadCreate](#) [threadCreateCallback](#))  
*Registers the callback for the ThreadCreate event.*

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadEndCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadEnd](#) threadEndCallback)  
*Registers the callback for the ThreadEnd event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadForkCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadFork](#) threadForkCallback)  
*Registers the callback for the ThreadFork event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadJoinCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadJoin](#) threadJoinCallback)  
*Registers the callback for the ThreadJoin event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadReleaseLockCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadReleaseLock](#) threadReleaseLockCallback)  
*Registers the callback for the ThreadReleaseLock event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadTaskCompleteCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadTaskComplete](#) threadTaskCompleteCallback)  
*Registers the callback for the ThreadTaskComplete event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadTaskCreateCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadTaskCreate](#) threadTaskCreateCallback)  
*Registers the callback for the ThreadTaskCreate event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadTaskSwitchCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadTaskSwitch](#) threadTaskSwitchCallback)  
*Registers the callback for the ThreadTaskSwitch event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadTeamBeginCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadTeamBegin](#) threadTeamBeginCallback)  
*Registers the callback for the ThreadTeamBegin event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadTeamEndCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadTeamEnd](#) threadTeamEndCallback)  
*Registers the callback for the ThreadTeamEnd event.*
- [OTF2\\_ErrorCode OTF2\\_GlobalEvtReaderCallbacks\\_SetThreadWaitCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_ThreadWait](#) threadWaitCallback)  
*Registers the callback for the ThreadWait event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalEvtReaderCallbacks\\_SetUnknownCallback](#)  
([OTF2\\_GlobalEvtReaderCallbacks](#) \*globalEvtReaderCallbacks, [OTF2\\_GlobalEvtReaderCallback\\_Unknown](#) unknownCallback)

*Registers the callback for unknown events.*

### E.20.1 Detailed Description

This defines the callbacks for the global event reader.

#### Source Template:

*templates/OTF2\_GlobalEvtReaderCallbacks.tmpl.h*

### E.20.2 Typedef Documentation

**E.20.2.1** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
BufferFlush)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
stopTime)`

Callback for the BufferFlush event record.

This event signals that the internal buffer was flushed at the given time.

#### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>stopTime</i>	The time the buffer flush finished.

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.2.2** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - Enter)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

Callback for the Enter event record.

An enter record indicates that the program enters a code region.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	Needs to be defined in a definition record References a <a href="#">Region</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_REGION</a> is available.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.3** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - Leave)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

Callback for the Leave event record.

A leave record indicates that the program leaves a code region.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	Needs to be defined in a definition record References a <a href="#">Region</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_REGION</a> is available.



### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.4** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
MeasurementOnOff)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, OTF2_MeasurementMode measurementMode)`

Callback for the MeasurementOnOff event record.

This event signals where the measurement system turned measurement on or off.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>measurementMode</i>	Is the measurement turned on ( <a href="#"><i>OTF2_MEASUREMENT_ON</i></a> ) or off ( <a href="#"><i>OTF2_MEASUREMENT_OFF</i></a> )?

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.5** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
Metric)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void  
*userData, OTF2_AttributeList *attributeList, OTF2_MetricRef metric,  
uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue  
*metricValues)`

Callback for the Metric event record.

A metric event is always stored at the location that recorded the metric. A metric event can reference a metric class or metric instance. Therefore, metric classes



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>metric</i>	Could be a metric class or a metric instance. References a <a href="#">MetricClass</a> , or a <a href="#">MetricInstance</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_METRIC</a> is available.
<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding <a href="#">MetricMember</a> definitions.
<i>metricValues</i>	List of metric values.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.6** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
MpiCollectiveBegin)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList)`

Callback for the MpiCollectiveBegin event record.

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.).

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

## Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.7** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
MpiCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList,  
OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator,  
uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)`

Callback for the MpiCollectiveEnd event record.

A MpiCollectiveEnd record marks the end of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.). It keeps the necessary information for this event: type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

## Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a <a href="#"><i>Comm</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_COMM</i></a> is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

## Since

Version 1.0

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

## E.20 oftf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.2.8** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
MpiIrecv)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, uint32_t sender,  
OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t  
requestID)`

Callback for the MpiIrecv event record.

A MpiIrecv record indicates that a MPI message was received (MPI\_IRECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi- cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_- COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.9** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
MpiIrecvRequest)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

Callback for the MpiIrecvRequest event record.

Signals the request of an receive, which can be completed later.

### Parameters

## APPENDIX E. FILE DOCUMENTATION

---

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the requested receive

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.10** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
MpiIsend)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, uint32_t receiver,  
OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength,  
uint64_t requestID)`

Callback for the MpiIsend event record.

A MpiIsend record indicates that a MPI message send process was initiated (MPI\_ISEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi- cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_- COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.11** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
MpiIsendComplete)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, uint64_t requestID)`

Callback for the MpiIsendComplete event record.

Signals the completion of non-blocking send request.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.12** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
MpiRecv)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, uint32_t sender,  
OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

Callback for the MpiRecv event record.

A MpiRecv record indicates that a MPI message was received (MPI\_RECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.13** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
MpiRequestCancelled)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, uint64_t requestID)`

Callback for the `MpiRequestCancelled` event record.

This events appears if the program canceled a request.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

### Since

Version 1.0

## E.20 oftf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.14** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
MpiRequestTest)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

Callback for the MpiRequestTest event record.

This events appears if the program tests if a request has already completed but the test failed.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.15** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
MpiSend)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, uint32_t receiver,  
OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

Callback for the MpiSend event record.

A MpiSend record indicates that a MPI message send process was initiated (MPI - SEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

### Parameters

<i>locationID</i>	The location where this event happened.
-------------------	---

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communicator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.16** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
OmpAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID,  
uint32_t acquisitionOrder)`

Callback for the OmpAcquireLock event record.

An OmpAcquireLock record marks that a thread acquires an OpenMP lock.

This event record is superseded by the [ThreadAcquireLock](#) event record and should not be used when the [ThreadAcquireLock](#) event record is in use.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.17** `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 <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>num- berOfRe- quest- edThreads</i>	Requested size of the team.

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.18** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
OmpJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void  
*userData, OTF2_AttributeList *attributeList)`

Callback for the OmpJoin event record.

---

## APPENDIX E. FILE DOCUMENTATION

---

An OmpJoin record marks that a team of threads is joint and only the master thread continues execution.

This event record is superseded by the [ThreadJoin](#) event record and should not be used when the [ThreadJoin](#) event record is in use.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.19** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
OmpReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID,  
uint32_t acquisitionOrder)`

Callback for the OmpReleaseLock event record.

An OmpReleaseLock record marks that a thread releases an OpenMP lock.

This event record is superseded by the [ThreadReleaseLock](#) event record and should not be used when the [ThreadReleaseLock](#) event record is in use.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

## E.20 oftf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.20** `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 <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the completed task instance.

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.21** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)`

Callback for the OmpTaskCreate event record.

An OmpTaskCreate record marks that an OpenMP Task was/will be created in the current region.

---

## APPENDIX E. FILE DOCUMENTATION

---

This event record is superseded by the [ThreadTaskCreate](#) event record and should not be used when the [ThreadTaskCreate](#) event record is in use.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the newly created task instance.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.22** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
OmpTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)`

Callback for the OmpTaskSwitch event record.

An OmpTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

This event record is superseded by the [ThreadTaskSwitch](#) event record and should not be used when the [ThreadTaskSwitch](#) event record is in use.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the now active task instance.

### Since

Version 1.0

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.23** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef  
parameter, int64_t value)`

Callback for the ParameterInt event record.

A ParameterInt record marks that in the current region, the specified integer parameter has the specified value.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	Parameter ID. References a <a href="#"><i>Parameter</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_PARAMETER</i></a> is available.
<i>value</i>	Value of the recorded parameter.

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.24** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
ParameterString)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList,  
OTF2_ParameterRef parameter, OTF2_StringRef string)`

Callback for the ParameterString event record.

A ParameterString record marks that in the current region, the specified string parameter has the specified value.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>string</i>	Value: Handle of a string definition References a <a href="#">String</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_STRING</a> is available.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.25** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
ParameterUnsignedInt)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, OTF2_ParameterRef parameter, uint64_t value)`

Callback for the ParameterUnsignedInt event record.

A ParameterUnsignedInt record marks that in the current region, the specified unsigned integer parameter has the specified value.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.26** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - RmaAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)`

Callback for the RmaAcquireLock event record.

An RmaAcquireLock record denotes the time a lock was aquired by the process.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> 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\*](#).

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.20.2.27** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
RmaAtomic)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef  
win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent,  
uint64_t bytesReceived, uint64_t matchingId)`

Callback for the RmaAtomic event record.

An RmaAtomic record denotes the time a atomic operation was issued.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the target process.
<i>type</i>	Type of atomic operation.
<i>bytesSent</i>	Bytes sent to target.
<i>bytesReceived</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.28** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
RmaCollectiveBegin)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList)`

Callback for the RmaCollectiveBegin event record.

An RmaCollectiveBegin record denotes the beginnig of a collective RMA operation.



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.29** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - RmaCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)`

Callback for the RmaCollectiveEnd event record.

"An RmaCollectiveEnd record denotes the end of a collective RMA operation.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<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 <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>root</i>	Root process for this operation.
<i>bytesSent</i>	Bytes sent in operation.
<i>bytesReceived</i>	Bytes receives in operation.

**Since**

Version 1.2

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.20.2.30** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
RmaGet)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void  
*userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win,  
uint32_t remote, uint64_t bytes, uint64_t matchingId)`

Callback for the RmaGet event record.

An RmaGet record denotes the time a get operation was issued.

**Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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>bytes</i>	Bytes received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

**Since**

Version 1.2

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.2.31** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - RmaGroupSync)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)`

Callback for the RmaGroupSync event record.

An RmaGroupSync record denotes the synchronization with a subgroup of processes on a window.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<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 <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>group</i>	Group of remote processes involved in synchronization. References a <a href="#">Group</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_GROUP</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.32** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - RmaOpCompleteBlocking)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

Callback for the RmaOpCompleteBlocking event record.

An RmaOpCompleteBlocking record denotes the local completion of a blocking RMA operation.

### Parameters

## APPENDIX E. FILE DOCUMENTATION

---

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.33** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
RmaOpCompleteNonBlocking)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

Callback for the RmaOpCompleteNonBlocking event record.

An RmaOpCompleteNonBlocking record denotes the local completion of a non-blocking RMA operation.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

## E.20 oftf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.34** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
RmaOpCompleteRemote)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

Callback for the RmaOpCompleteRemote event record.

An RmaOpCompleteRemote record denotes the local completion of an RMA operation.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.35** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
RmaOpTest)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef  
win, uint64_t matchingId)`

Callback for the RmaOpTest event record.

An RmaOpTest record denotes that a non-blocking RMA operation has been tested for completion unsuccessfully.

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>matchingId</i>	ID used for matching the appropriate completion record.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.36** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
RmaPut)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void  
*userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win,  
uint32_t remote, uint64_t bytes, uint64_t matchingId)`

Callback for the RmaPut event record.

An RmaPut record denotes the time a put operation was issued.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the target process.
<i>bytes</i>	Bytes sent to target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.37** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
RmaReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)`

Callback for the RmaReleaseLock event record.

An RmaReleaseLock record denotes the time the lock was released.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock released, if multiple locks are defined on a window.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.38** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
RmaRequestLock)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType  
lockType)`

Callback for the RmaRequestLock event record.

## APPENDIX E. FILE DOCUMENTATION

An `RmaRequestLock` record denotes the time a lock was requested and with it the earliest time it could have been granted. It is used to mark (possibly) non-blocking lock request, as defined by the MPI standard.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.39** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ -  
RmaSync)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void  
*userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win,  
uint32_t remote, OTF2_RmaSyncType syncType)`

Callback for the `RmaSync` event record.

An `RmaSync` record denotes the direct synchronization with a possibly remote process.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>remote</i>	Rank of the locked remote process.
<i>syncType</i>	Type of synchronization.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.40** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
RmaTryLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time,  
void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef  
win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)`

Callback for the RmaTryLock event record.

An RmaTryLock record denotes the time of an unsuccessful attempt to acquire the lock.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_RMA_WIN</i></a> is available.
<i>remote</i>	Rank of the locked remote process.
<i>lockId</i>	ID of the lock acquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock acquired.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

## APPENDIX E. FILE DOCUMENTATION

**E.20.2.41** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
RmaWaitChange)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win)`

Callback for the RmaWaitChange event record.

An RmaWaitChange record denotes the change of a window that was waited for.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window used for this operation. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.42** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
RmaWinCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList,  
OTF2_RmaWinRef win)`

Callback for the RmaWinCreate event record.

An RmaWinCreate record denotes the creation of an RMA window.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window created. References a <a href="#">RmaWin</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_RMA_WIN</a> is available.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.43** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - RmaWinDestroy)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win)`

Callback for the RmaWinDestroy event record.

An RmaWinDestroy record denotes the destruction of an RMA window.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>win</i>	ID of the window destroyed. References a <a href="#"><i>RmaWin</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_- MAPPING_RMA_WIN</i></a> is available.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.44** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the ThreadAcquireLock event record.

An ThreadAcquireLock record marks that a thread acquires an lock.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.45** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

Callback for the ThreadBegin event record.

Marks the begin of a thread created by another thread.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>sequenceCount</i>	A threadContingent unique number. The corresponding <a href="#">ThreadCreate</a> event does have the same number.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.46** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

Callback for the ThreadCreate event record.

The location created successfully a new thread.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#"><i>Comm</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_COMM</i></a> is available.
<i>sequenceCount</i>	A <code>threadContingent</code> unique number. The corresponding <a href="#"><i>Thread-Begin</i></a> event does have the same number.

### Since

Version 1.3

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.47** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

Callback for the ThreadEnd event record.

---

## APPENDIX E. FILE DOCUMENTATION

---

Marks the end of a thread.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2-MAPPING_COMM</a> is available.
<i>sequence-Count</i>	A threadContingent unique number. The corresponding <a href="#">ThreadWait</a> event does have the same number. <a href="#">OTF2_UNDEFINED_UINT64</a> in case no corresponding <a href="#">ThreadWait</a> event exists.

### Since

Version 1.3

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.48** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)`

Callback for the ThreadFork event record.

An ThreadFork record marks that an thread forks a thread team.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>numberOfRequestedThreads</i>	Requested size of the team.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.49** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model)`

Callback for the ThreadJoin event record.

An ThreadJoin record marks that a team of threads is joint and only the master thread continues execution.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.50** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the ThreadReleaseLock event record.

An ThreadReleaseLock record marks that a thread releases an lock.

## APPENDIX E. FILE DOCUMENTATION

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>model</i>	The threading paradigm this event took place.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.51** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
ThreadTaskComplete)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t  
generationNumber)`

Callback for the ThreadTaskComplete event record.

An ThreadTaskComplete record indicates that the execution of an OpenMP task has finished.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.52** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

Callback for the ThreadTaskCreate event record.

An ThreadTaskCreate record marks that an task in was/will be created and will be processed by the specified thread team.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalEvtCallbacks</i></a> or <a href="#"><i>OTF2_GlobalEvtReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#"><i>Comm</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_COMM</i></a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.20.2.53** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

Callback for the ThreadTaskSwitch event record.

An ThreadTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

#### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.54** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadTeamBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam)`

Callback for the ThreadTeamBegin event record.

The current location enters the specified thread team.

#### Parameters

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.55** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
ThreadTeamEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList,  
OTF2_CommRef threadTeam)`

Callback for the ThreadTeamEnd event record.

The current location leaves the specified thread team.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.20.2.56** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadWait)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadContingent, uint64_t sequenceCount)`

Callback for the ThreadWait event record.

The location waits for the completion of another thread.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>threadContingent</i>	The thread contingent. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>sequenceCount</i>	A threadContingent unique number. The corresponding <a href="#">Thread-End</a> event does have the same number.

### Since

Version 1.3

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.20.2.57** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown event record.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

### E.20.3 Function Documentation

**E.20.3.1** void `OTF2_GlobalEvtReaderCallbacks_Clear` ( `OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks`  
)

Clears a struct for the global event callbacks.

#### Parameters

<code>globalEvtReaderCallbacks</code>	Handle to a struct previously allocated with <a href="#"><i>OTF2_GlobalEvtReaderCallbacks_New</i></a> .
---------------------------------------	---

**E.20.3.2** void `OTF2_GlobalEvtReaderCallbacks_Delete` ( `OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks`  
)

Deallocates a struct for the global event callbacks.

#### Parameters

<code>globalEvtReaderCallbacks</code>	Handle to a struct previously allocated with <a href="#"><i>OTF2_GlobalEvtReaderCallbacks_New</i></a> .
---------------------------------------	---

**E.20.3.3** `OTF2_GlobalEvtReaderCallbacks*` `OTF2_GlobalEvtReaderCallbacks_New`  
( void )

Allocates a new struct for the event callbacks.

### Returns

A newly allocated struct of type [\*OTF2\\_GlobalEvtReaderCallbacks\*](#).

**E.20.3.4   OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetBufferFlushCallback**  
**( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*,**  
**OTF2\_GlobalEvtReaderCallback\_BufferFlush *bufferFlushCallback* )**

Registers the callback for the BufferFlush event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>bufferFlushCallback</i>	Function which should be called for all BufferFlush events.

**Since**

Version 1.0

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.5   OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetEnterCallback**  
**( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*,**  
**OTF2\_GlobalEvtReaderCallback\_Enter *enterCallback* )**

Registers the callback for the Enter event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter events.

**Since**

Version 1.0

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.6** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_SetLeaveCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_Leave** *leaveCallback* )

Registers the callback for the Leave event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>leaveCallback</i>	Function which should be called for all Leave events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.7** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_SetMeasurementOnOffCallback** ( **OTF2\_GlobalEvtReaderCallbacks**  
\* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_MeasurementOnOff** *measurementOnOffCallback*  
)

Registers the callback for the MeasurementOnOff event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
---------------------------------	---------------------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>measurementOnOff-Callback</i>	Function which should be called for all MeasurementOnOff events.
----------------------------------	--

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.8 OTF2\_StatusCode OTF2\_GlobalEvtReaderCallbacks\_SetMetricCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_Metric *metricCallback* )**

Registers the callback for the Metric event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric events.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument



## E.20 oftf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.3.9 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-**  
**SetMpiCollectiveBeginCallback ( OTF2\_GlobalEvtReaderCallbacks**  
**\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-**  
**MpiCollectiveBegin *mpiCollectiveBeginCallback***  
**)**

Registers the callback for the MpiCollectiveBegin event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin events.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid *defReaderCallbacks* argument

**E.20.3.10 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-**  
**SetMpiCollectiveEndCallback ( OTF2\_GlobalEvtReaderCallbacks**  
**\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-**  
**MpiCollectiveEnd *mpiCollectiveEndCallback* )**

Registers the callback for the MpiCollectiveEnd event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEndCallback</i>	Function which should be called for all MpiCollectiveEnd events.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.20.3.11** `OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvCallback ( OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_MpiIrecv mpiIrecvCallback )`

Registers the callback for the `MpiIrecv` event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all <code>MpiIrecv</code> events.

**Since**

Version 1.0

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.20.3.12** `OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetMpiIrecvRequestCallback ( OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_MpiIrecvRequest mpiIrecvRequestCallback )`

Registers the callback for the `MpiIrecvRequest` event.

**Parameters**

---

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all <code>MpiIrecvRequest</code> events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.13 OTF2\_StatusCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiIrecvCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_MpiIrecv** *mpiIrecvCallback* )

Registers the callback for the `MpiIrecv` event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all <code>MpiIrecv</code> events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.14** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-SetMpiIsendCompleteCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_-MpiIsendComplete** *mpiIsendCompleteCallback* )

Registers the callback for the MpiIsendComplete event.

#### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIsendCompleteCallback</i>	Function which should be called for all MpiIsendComplete events.

#### Since

Version 1.0

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.20.3.15** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_SetMpiRecvCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_MpiRecv** *mpiRecvCallback* )

Registers the callback for the MpiRecv event.

#### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecvCallback</i>	Function which should be called for all MpiRecv events.

#### Since

Version 1.0

## E.20 oftf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.16** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_**  
**SetMpiRequestCancelledCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_**  
**MpiRequestCancelled** *mpiRequestCancelledCallback*  
)

Registers the callback for the `MpiRequestCancelled` event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRequestCancelledCallback</i>	Function which should be called for all <code>MpiRequestCancelled</code> events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.17** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_**  
**SetMpiRequestTestCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_**  
**MpiRequestTest** *mpiRequestTestCallback* )

Registers the callback for the `MpiRequestTest` event.

### Parameters

## APPENDIX E. FILE DOCUMENTATION

---

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRequestTestCallback</i>	Function which should be called for all <code>MpiRequestTest</code> events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.18** `OTF2_StatusCode OTF2_GlobalEvtReaderCallbacks_SetMpiSendCallback ( OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_MpiSend mpiSendCallback )`

Registers the callback for the `MpiSend` event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSendCallback</i>	Function which should be called for all <code>MpiSend</code> events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.3.19** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_**  
**SetOmpAcquireLockCallback** ( **OTF2\_GlobalEvtReaderCallbacks**  
**\*** *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_**  
**OmpAcquireLock** *ompAcquireLockCallback* )

Registers the callback for the OmpAcquireLock event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompAcquireLockCallback</i>	Function which should be called for all OmpAcquireLock events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.20.3.20** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_**  
**SetOmpForkCallback** ( **OTF2\_GlobalEvtReaderCallbacks** **\*** *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_OmpFork** *ompForkCallback* )

Registers the callback for the OmpFork event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompForkCallback</i>	Function which should be called for all OmpFork events.

### Since

Version 1.0

## APPENDIX E. FILE DOCUMENTATION

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.21** **OTF2\_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.
<i>ompJoinCallback</i>	Function which should be called for all OmpJoin events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.22** **OTF2\_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>ompReleaseLockCallback</i>	Function which should be called for all OmpReleaseLock events.



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.23** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_**  
**SetOmpTaskCompleteCallback** ( **OTF2\_GlobalEvtReaderCallbacks**  
**\*** *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_**  
**OmpTaskComplete** *ompTaskCompleteCallback*  
)

Registers the callback for the OmpTaskComplete event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCompleteCallback</i>	Function which should be called for all OmpTaskComplete events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.24** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_**  
**SetOmpTaskCreateCallback** ( **OTF2\_GlobalEvtReaderCallbacks** **\***  
*globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_**  
**OmpTaskCreate** *ompTaskCreateCallback* )

Registers the callback for the OmpTaskCreate event.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCreateCallback</i>	Function which should be called for all OmpTaskCreate events.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.25** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-SetOmpTaskSwitchCallback** ( **OTF2\_GlobalEvtReaderCallbacks \*** *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_-OmpTaskSwitch** *ompTaskSwitchCallback* )

Registers the callback for the OmpTaskSwitch event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskSwitchCallback</i>	Function which should be called for all OmpTaskSwitch events.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.3.26** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_SetParameterIntCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_ParameterInt** *parameterIntCallback*  
)

Registers the callback for the ParameterInt event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterIntCallback</i>	Function which should be called for all ParameterInt events.

### Since

Version 1.0

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.27** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_SetParameterStringCallback** ( **OTF2\_GlobalEvtReaderCallbacks**  
\* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_ParameterString** *parameterStringCallback* )

Registers the callback for the ParameterString event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterStringCallback</i>	Function which should be called for all ParameterString events.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.28** **OTF2\_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 <code>ParameterUnsignedInt</code> events.

### Since

Version 1.0

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.29** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-SetRmaAcquireLockCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* ***globalEvtReaderCallbacks***, **OTF2\_GlobalEvtReaderCallback\_** **RmaAcquireLock** ***rmaAcquireLockCallback*** )

Registers the callback for the `RmaAcquireLock` event.

### Parameters

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaAcquireLockCallback</i>	Function which should be called for all RmaAcquireLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.30** **OTF2\_**`ErrorCode` **OTF2\_GlobalEvtReaderCallbacks\_SetRmaAtomicCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_RmaAtomic** *rmaAtomicCallback* )

Registers the callback for the RmaAtomic event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaAtomicCallback</i>	Function which should be called for all RmaAtomic events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.31** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-**  
**SetRmaCollectiveBeginCallback (** **OTF2\_GlobalEvtReaderCallbacks**  
**\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-**  
**RmaCollectiveBegin *rmaCollectiveBeginCallback***  
**)**

Registers the callback for the RmaCollectiveBegin event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaCollectiveBeginCallback</i>	Function which should be called for all RmaCollectiveBegin events.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.32** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-**  
**SetRmaCollectiveEndCallback (** **OTF2\_GlobalEvtReaderCallbacks**  
**\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-**  
**RmaCollectiveEnd *rmaCollectiveEndCallback* )**

Registers the callback for the RmaCollectiveEnd event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaCollectiveEndCallback</i>	Function which should be called for all RmaCollectiveEnd events.

E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

Since

Version 1.2

Returns

*OTF2\_SUCCESS* if successful  
*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.20.3.33** `OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaGetCallback ( OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_RmaGet rmaGetCallback )`

Registers the callback for the RmaGet event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaGetCallback</i>	Function which should be called for all RmaGet events.

Since

Version 1.2

Returns

*OTF2\_SUCCESS* if successful  
*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.20.3.34** `OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaGroupSyncCallback ( OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_RmaGroupSync rmaGroupSyncCallback )`

Registers the callback for the RmaGroupSync event.

Parameters

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaGroupSyncCallback</i>	Function which should be called for all RmaGroupSync events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.35 OTF2\_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.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument



E.20 oftf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.3.36 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.
<i>rmaOpCompleteNonBlockingCallback</i>	Function which should be called for all RmaOpCompleteNonBlocking events.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful  
*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.20.3.37 OTF2\_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>rmaOpCompleteRemoteCallback</i>	Function which should be called for all RmaOpCompleteRemote events.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.20.3.38** `OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaOpTestCallback ( OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_RmaOpTest rmaOpTestCallback )`

Registers the callback for the RmaOpTest event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpTestCallback</i>	Function which should be called for all RmaOpTest events.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.20.3.39** `OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_SetRmaPutCallback ( OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_RmaPut rmaPutCallback )`

Registers the callback for the RmaPut event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>rmaPut-Callback</i>	Function which should be called for all RmaPut events.
------------------------	--

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.40 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-SetRmaReleaseLockCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-RmaReleaseLock *rmaReleaseLockCallback* )**

Registers the callback for the RmaReleaseLock event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaReleaseLockCallback</i>	Function which should be called for all RmaReleaseLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.20.3.41** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_**  
**SetRmaRequestLockCallback** ( **OTF2\_GlobalEvtReaderCallbacks**  
**\*** *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_**  
**RmaRequestLock** *rmaRequestLockCallback* )

Registers the callback for the RmaRequestLock event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaRequestLockCallback</i>	Function which should be called for all RmaRequestLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks*  
argument

**E.20.3.42** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_**  
**SetRmaSyncCallback** ( **OTF2\_GlobalEvtReaderCallbacks** **\*** *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_RmaSync** *rmaSyncCallback* )

Registers the callback for the RmaSync event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaSyncCallback</i>	Function which should be called for all RmaSync events.

### Since

Version 1.2

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.43** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_SetRmaTryLockCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_RmaTryLock** *rmaTryLockCallback* )

Registers the callback for the RmaTryLock event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaTryLockCallback</i>	Function which should be called for all RmaTryLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.44** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_SetRmaWaitChangeCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_RmaWaitChange** *rmaWaitChangeCallback* )

Registers the callback for the RmaWaitChange event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
---------------------------------	---------------------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>rmaWait-Change-Callback</i>	Function which should be called for all RmaWaitChange events.
--------------------------------	---

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.45 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-SetRmaWinCreateCallback ( OTF2\_GlobalEvtReaderCallbacks \* globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_-RmaWinCreate *rmaWinCreateCallback* )**

Registers the callback for the RmaWinCreate event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWinCreateCallback</i>	Function which should be called for all RmaWinCreate events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.3.46** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-SetRmaWinDestroyCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_-RmaWinDestroy** *rmaWinDestroyCallback* )

Registers the callback for the RmaWinDestroy event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWinDestroyCallback</i>	Function which should be called for all RmaWinDestroy events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.20.3.47** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-SetThreadAcquireLockCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_-ThreadAcquireLock** *threadAcquireLockCallback* )

Registers the callback for the ThreadAcquireLock event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadAcquireLockCallback</i>	Function which should be called for all ThreadAcquireLock events.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.48** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_SetThreadBeginCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_ThreadBegin** *threadBeginCallback* )

Registers the callback for the ThreadBegin event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadBeginCallback</i>	Function which should be called for all ThreadBegin events.

**Since**

Version 1.3

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.49** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_SetThreadCreateCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_ThreadCreate** *threadCreateCallback*  
)

Registers the callback for the ThreadCreate event.

**Parameters**

---



## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadCreateCallback</i>	Function which should be called for all ThreadCreate events.

### Since

Version 1.3

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.50** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_SetThreadEndCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_ThreadEnd** *threadEndCallback* )

Registers the callback for the ThreadEnd event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadEndCallback</i>	Function which should be called for all ThreadEnd events.

### Since

Version 1.3

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## APPENDIX E. FILE DOCUMENTATION

---

**E.20.3.51** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_SetThreadForkCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_ThreadFork** *threadForkCallback* )

Registers the callback for the ThreadFork event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadForkCallback</i>	Function which should be called for all ThreadFork events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.20.3.52** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_SetThreadJoinCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_ThreadJoin** *threadJoinCallback* )

Registers the callback for the ThreadJoin event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadJoinCallback</i>	Function which should be called for all ThreadJoin events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

E.20.3.53 **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-SetThreadReleaseLockCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_-ThreadReleaseLock** *threadReleaseLockCallback* )

Registers the callback for the ThreadReleaseLock event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadReleaseLockCallback</i>	Function which should be called for all ThreadReleaseLock events.

Since

Version 1.2

Returns

*OTF2\_SUCCESS* if successful  
*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

E.20.3.54 **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-SetThreadTaskCompleteCallback** ( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*, **OTF2\_GlobalEvtReaderCallback\_-ThreadTaskComplete** *threadTaskCompleteCallback* )

Registers the callback for the ThreadTaskComplete event.

Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
---------------------------------	---------------------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>thread-TaskCompleteCallback</i>	Function which should be called for all ThreadTaskComplete events.
------------------------------------	--

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.55** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-SetThreadTaskCreateCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-ThreadTaskCreate *threadTaskCreateCallback* )**

Registers the callback for the ThreadTaskCreate event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTaskCreateCallback</i>	Function which should be called for all ThreadTaskCreate events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## E.20 otf2/OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

**E.20.3.56** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-**  
**SetThreadTaskSwitchCallback** ( **OTF2\_GlobalEvtReaderCallbacks**  
\* **globalEvtReaderCallbacks**, **OTF2\_GlobalEvtReaderCallback\_-**  
**ThreadTaskSwitch** **threadTaskSwitchCallback** )

Registers the callback for the ThreadTaskSwitch event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTaskSwitchCallback</i>	Function which should be called for all ThreadTaskSwitch events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.57** **OTF2\_ErrorCode** **OTF2\_GlobalEvtReaderCallbacks\_-**  
**SetThreadTeamBeginCallback** ( **OTF2\_GlobalEvtReaderCallbacks**  
\* **globalEvtReaderCallbacks**, **OTF2\_GlobalEvtReaderCallback\_-**  
**ThreadTeamBegin** **threadTeamBeginCallback** )

Registers the callback for the ThreadTeamBegin event.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTeamBeginCallback</i>	Function which should be called for all ThreadTeamBegin events.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.58 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-SetThreadTeamEndCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-ThreadTeamEnd *threadTeamEndCallback* )**

Registers the callback for the ThreadTeamEnd event.

**Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTeamEndCallback</i>	Function which should be called for all ThreadTeamEnd events.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.20.3.59 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadWaitCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ThreadWait *threadWaitCallback* )**

Registers the callback for the ThreadWait event.

**Parameters**

## E.21 otf2/OTF2\_GlobalSnapReader.h File Reference

---

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadWait-Callback</i>	Function which should be called for all ThreadWait events.

### Since

Version 1.3

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.20.3.60** **OTF2\_StatusCode** **OTF2\_GlobalEvtReaderCallbacks\_SetUnknownCallback**  
( **OTF2\_GlobalEvtReaderCallbacks** \* *globalEvtReaderCallbacks*,  
**OTF2\_GlobalEvtReaderCallback\_Unknown** *unknownCallback* )

Registers the callback for unknown events.

### Parameters

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	Function which should be called for all unknown events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## E.21 otf2/OTF2\_GlobalSnapReader.h File Reference

This is the global snapshot event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
```

---

## APPENDIX E. FILE DOCUMENTATION

---

```
#include <otf2/OTF2_SnapReader.h>
#include <otf2/OTF2_GlobalSnapReaderCallbacks.h>
```

### Functions

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReader\\_ReadSnapshots](#) ([OTF2\\_GlobalSnapReader](#) \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)  
*Reads the given number of records from the global snap event reader.*
- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReader\\_SetCallbacks](#) ([OTF2\\_GlobalSnapReader](#) \*reader, const [OTF2\\_GlobalSnapReaderCallbacks](#) \*callbacks, void \*userData)

*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### E.21.1 Detailed Description

This is the global snapshot event reader.

#### Since

Version 1.2

Used to read from multiple local snap event readers, and provide them in a timely ordered sequence.

### E.21.2 Function Documentation

**E.21.2.1** [OTF2\\_ErrorCode OTF2\\_GlobalSnapReader\\_ReadSnapshots](#) (  
[OTF2\\_GlobalSnapReader](#) \* reader, uint64\_t recordsToRead, uint64\_t \*  
recordsRead )

Reads the given number of records from the global snap event reader.

#### Parameters

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.



## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

out	<i>recordsRead</i>	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given recordsToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finished his job by checking <code>recordsRead &lt; recordsToRead</code> .
-----	--------------------	---

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.21.2.2 OTF2\_ErrorCode OTF2\_GlobalSnapReader\_SetCallbacks**  
( OTF2\_GlobalSnapReader \* *reader*, const OTF2\_GlobalSnapReaderCallbacks \* *callbacks*, void \* *userData* )

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

### Parameters

<i>reader</i>	Reader object which reads the snap events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#"><i>OTF2_GlobalSnapReaderCallbacks_New</i></a> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

This defines the callbacks for the global snap reader.

---

## APPENDIX E. FILE DOCUMENTATION

---

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

### Typedefs

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalSnapReaderCallback\\_Enter](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_RegionRef](#) region)  
*Callback for the Enter snap record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalSnapReaderCallback\\_MeasurementOnOff](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_MeasurementMode](#) measurementMode)  
*Callback for the MeasurementOnOff snap record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalSnapReaderCallback\\_Metric](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_MetricRef](#) metric, uint8\_t numberOfMetrics, const [OTF2\\_Type](#) \*typeIDs, const [OTF2\\_MetricValue](#) \*metricValues)  
*Callback for the Metric snap record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalSnapReaderCallback\\_MpiCollectiveBegin](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime)  
*Callback for the MpiCollectiveBegin snap record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalSnapReaderCallback\\_MpiCollectiveEnd](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_CollectiveOp](#) collectiveOp, [OTF2\\_CommRef](#) communicator, uint32\_t root, uint64\_t sizeSent, uint64\_t sizeReceived)  
*Callback for the MpiCollectiveEnd snap record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalSnapReaderCallback\\_MpiIrecv](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, uint32\_t sender, [OTF2\\_CommRef](#) communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID)  
*Callback for the MpiIrecv snap record.*

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_MpiIrecvRequest)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint64\_t requestID)  
*Callback for the MpiIrecvRequest snap record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_MpiIsend)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID)  
*Callback for the MpiIsend snap record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_MpiIsendComplete)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint64\_t requestID)  
*Callback for the MpiIsendComplete snap record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_MpiRecv)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint32\_t sender, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)  
*Callback for the MpiRecv snap record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_MpiSend)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)  
*Callback for the MpiSend snap record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_OmpAcquireLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint32\_t lockID, uint32\_t acquisitionOrder)  
*Callback for the OmpAcquireLock snap record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_OmpFork)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint32\_t numberOfRequestedThreads)  
*Callback for the OmpFork snap record.*
- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_OmpTaskCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint64\_t taskID)

---

## APPENDIX E. FILE DOCUMENTATION

---

*Callback for the OmpTaskCreate snap record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_OmpTaskSwitch)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, uint64\_t taskID)

*Callback for the OmpTaskSwitch snap record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_ParameterInt)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, OTF2\_ParameterRef parameter, int64\_t value)

*Callback for the ParameterInt snap record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_ParameterString)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, OTF2\_ParameterRef parameter, OTF2\_StringRef string)

*Callback for the ParameterString snap record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_ParameterUnsignedInt)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, OTF2\_ParameterRef parameter, uint64\_t value)

*Callback for the ParameterUnsignedInt snap record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_SnapshotEnd)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t contReadPos)

*Callback for the SnapshotEnd snap record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_SnapshotStart)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t numberOfRecords)

*Callback for the SnapshotStart snap record.*

- typedef OTF2\_CallbackCode(\* OTF2\_GlobalSnapReaderCallback\_Unknown)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList)

*Callback for an unknown snap record.*

- typedef struct OTF2\_GlobalSnapReaderCallbacks\_struct OTF2\_GlobalSnapReaderCallbacks

*Opaque struct which holds all snap record callbacks.*

### Functions

- void OTF2\_GlobalSnapReaderCallbacks\_Clear (OTF2\_GlobalSnapReaderCallbacks \*globalSnapReaderCallbacks)

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

*Clears a struct for the global snap callbacks.*

- void [OTF2\\_GlobalSnapReaderCallbacks\\_Delete](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks)

*Deallocates a struct for the global snap callbacks.*

- [OTF2\\_GlobalSnapReaderCallbacks](#) \* [OTF2\\_GlobalSnapReaderCallbacks\\_New](#) (void)

*Allocates a new struct for the snap callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalSnapReaderCallbacks\\_SetEnterCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_Enter](#) enterCallback)

*Registers the callback for the Enter snap.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalSnapReaderCallbacks\\_SetMeasurementOnOffCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MeasurementOnOff](#) measurementOnOffCallback)

*Registers the callback for the MeasurementOnOff snap.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalSnapReaderCallbacks\\_SetMetricCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_Metric](#) metricCallback)

*Registers the callback for the Metric snap.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiCollectiveBeginCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiCollectiveBegin](#) mpiCollectiveBeginCallback)

*Registers the callback for the MpiCollectiveBegin snap.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiCollectiveEndCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiCollectiveEnd](#) mpiCollectiveEndCallback)

*Registers the callback for the MpiCollectiveEnd snap.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiIrecvCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiIrecv](#) mpiIrecvCallback)

*Registers the callback for the MpiIrecv snap.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiIrecvRequestCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiIrecvRequest](#) mpiIrecvRequestCallback)

*Registers the callback for the MpiIrecvRequest snap.*

- [OTF2\\_ErrorCode](#) [OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiIsendCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiIsend](#) mpiIsendCallback)

---

## APPENDIX E. FILE DOCUMENTATION

---

*Registers the callback for the MpiIsend snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiIsendCompleteCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiIsendComplete](#) mpiIsendCompleteCallback)

*Registers the callback for the MpiIsendComplete snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiRecvCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiRecv](#) mpiRecvCallback)

*Registers the callback for the MpiRecv snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiSendCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiSend](#) mpiSendCallback)

*Registers the callback for the MpiSend snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetOmpAcquireLockCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_OmpAcquireLock](#) ompAcquireLockCallback)

*Registers the callback for the OmpAcquireLock snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetOmpForkCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_OmpFork](#) ompForkCallback)

*Registers the callback for the OmpFork snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetOmpTaskCreateCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_OmpTaskCreate](#) ompTaskCreateCallback)

*Registers the callback for the OmpTaskCreate snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetOmpTaskSwitchCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_OmpTaskSwitch](#) ompTaskSwitchCallback)

*Registers the callback for the OmpTaskSwitch snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetParameterIntCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_ParameterInt](#) parameterIntCallback)

*Registers the callback for the ParameterInt snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetParameterStringCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_ParameterString](#) parameterStringCallback)

*Registers the callback for the ParameterString snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetParameterUnsignedIntCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_ParameterUnsignedInt](#) parameterUnsignedIntCallback)

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

*Registers the callback for the ParameterUnsignedInt snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetSnapshotEndCallback](#)  
([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_SnapshotEnd](#) snapshotEndCallback)

*Registers the callback for the SnapshotEnd snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetSnapshotStartCallback](#)  
([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_SnapshotStart](#) snapshotStartCallback)

*Registers the callback for the SnapshotStart snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetUnknownCallback](#)  
([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_Unknown](#) unknownCallback)

*Registers the callback for unknown snaps.*

### E.22.1 Detailed Description

This defines the callbacks for the global snap reader.

#### Source Template:

*templates/OTF2\_GlobalSnapReaderCallbacks.tmpl.h*

### E.22.2 Typedef Documentation

**E.22.2.1** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_Enter)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_RegionRef region)`

Callback for the Enter snap record.

This record exists for each [Enter](#) event where the corresponding [Leave](#) event did not occur before the snapshot.

#### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>region</i>	Needs to be defined in a definition record References a <a href="#">Region</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_REGION</a> is available.
---------------	--

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.2** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
MeasurementOnOff)(OTF2_LocationRef locationID,  
OTF2_TimeStamp snapTime, void *userData, OTF2_  
AttributeList *attributeList, OTF2_TimeStamp origEventTime,  
OTF2_MeasurementMode measurementMode)`

Callback for the MeasurementOnOff snap record.

The last occurrence of an [MeasurementOnOff](#) event of this location, if any.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent- Time</i>	The original time this event happended.
<i>measure- mentMode</i>	Is the measurement turned on ( <a href="#">OTF2_MEASUREMENT_ON</a> ) or off ( <a href="#">OTF2_MEASUREMENT_OFF</a> )?

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).



## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

**E.22.2.3** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_Metric)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)`

Callback for the Metric snap record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

As an exception for metric classes where the metric mode detontes an [OTF2\\_METRIC\\_VALUE\\_RELATIVE](#) mode the value indicates the accumulation of all previous metric values recorded.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>metric</i>	Could be a metric class or a metric instance. References a <a href="#">MetricClass</a> , or a <a href="#">MetricInstance</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_METRIC</a> is available.
<i>numberOf-Metrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding <a href="#">MetricMember</a> definitions.
<i>metricVal-ues</i>	List of metric values.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.4** `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_  
MpiCollectiveBegin)(OTF2_LocationRef locationID,  
OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList  
*attributeList, OTF2_TimeStamp origEventTime)`

Callback for the MpiCollectiveBegin snap record.

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

#### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.5** `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_  
MpiCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp,  
OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t  
sizeReceived)`

Callback for the MpiCollectiveEnd snap record.

Indicates that this location completed a collective operation locally but not all of the participating locations completed the operation yet. The corresponding [Mpi-CollectiveBeginSnap](#) record is still in the snapshot though.

#### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.6** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
MpiIrecv)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength, uint64_t requestID)`

Callback for the MpiIrecv snap record.

This record exists for each [MpiIrecv](#) event where the matching send message event did not occur on the remote location before the snapshot. This could either be an [MpiSend](#) or an [MpiIsendComplete](#) event. Or an [MpiIrecvRequest](#) occurred before this event but the corresponding [MpiIrecv](#) event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing [MpiIrecvRequest](#) is not yet known.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.7** `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback -  
MpiIrecvRequest)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t requestID)`

Callback for the `MpiIrecvRequest` snap record.

This record exists for each [MpiIrecvRequest](#) event where an corresponding [MpiIrecv](#) or [MpiRequestCancelled](#) event did not occur on this location before the snapshot. Or the corresponding [MpiIrecv](#) did occurred (the [MpiIrecvSnap](#) record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. This could either be an [MpiRecv](#) or an [MpiIrecv](#) event.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the requested receive

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.8** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback -  
MpiIsend)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength, uint64_t requestID)`

Callback for the MpiIsend snap record.

This record exists for each [MpiIsend](#) event where an corresponding [MpiIsendComplete](#) or [MpiRequestCancelled](#) event did not occur on this location before the snapshot. Or the corresponding [MpiIsendComplete](#) did occurred (the [MpiIsendCompleteSnap](#) record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. (This could either be an [MpiRecv](#) or an [MpiIrecv](#) event.)

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.2

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.22.2.9** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
MpiIsendComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t requestID)`

Callback for the MpiIsendComplete snap record.

This record exists for each [\*MpiIsend\*](#) event where the corresponding [\*MpiIsendComplete\*](#) event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an [\*MpiRecv\*](#) or an [\*MpiIrecv\*](#) event.) .

## Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalSnapCallbacks</i></a> or <a href="#"><i>OTF2_GlobalSnapReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the related request

## Since

Version 1.2

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.22.2.10** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
MpiRecv)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength)`

Callback for the MpiRecv snap record.

This record exists for each [\*MpiRecv\*](#) event where the matching send message event did not occur on the remote location before the snapshot. This could either be

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

an [MpiSend](#) or an [MpiIsendComplete](#) event. Or an [MpiIrecvRequest](#) occurred before this event but the corresponding [MpiIrecv](#) event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing [MpiIrecvRequest](#) is not yet known.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.11** `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_  
MpiSend)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength)`

Callback for the MpiSend snap record.

This record exists for each [MpiSend](#) event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an [MpiRecv](#) or an [MpiIrecv](#) event. Note that it may so, that a previous [MpiIsend](#) with the same envelope than this one is neither completed not canceled yet, thus the matching receive may already occurred, but the matching couldn't be done yet.

### Parameters

## APPENDIX E. FILE DOCUMENTATION

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<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 communicator.
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

Since

Version 1.2

Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.12** `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback -  
OmpAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the OmpAcquireLock snap record.

This record exists for each [OmpAcquireLock](#) event where the corresponding [OmpReleaseLock](#) did not occurred before this snapshot yet.

Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<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-sitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible).
<b>628</b>	Corresponding acquire-release events have same number.



## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.22.2.13** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_ -  
OmpFork)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint32_t numberOfRequestedThreads)`

Callback for the OmpFork snap record.

This record exists for each [\*OmpFork\*](#) event where the corresponding [\*OmpJoin\*](#) did not occurred before this snapshot.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalSnapCallbacks</i></a> or <a href="#"><i>OTF2_GlobalSnapReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent- Time</i>	The original time this event happended.
<i>num- berOfRe- quest- edThreads</i>	Requested size of the team.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.22.2.14** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t taskID)`

Callback for the OmpTaskCreate snap record.

This record exists for each *OmpTaskCreate* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the newly created task instance.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.22.2.15** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
OmpTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t taskID)`

Callback for the OmpTaskSwitch snap record.

This record exists for each *OmpTaskSwitch* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the now active task instance.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.16** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter,  
int64_t value)`

Callback for the ParameterInt snap record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent-Time</i>	The original time this event happended.
<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

### Since

Version 1.2

## Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.17** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
ParameterString)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter,  
OTF2_StringRef string)`

Callback for the ParameterString snap record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

## Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<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 <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>string</i>	Value: Handle of a string definition References a <a href="#">String</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_STRING</a> is available.

## Since

Version 1.2

## Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

**E.22.2.18** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
ParameterUnsignedInt)(OTF2_LocationRef locationID,  
OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList  
*attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef  
parameter, uint64_t value)`

Callback for the ParameterUnsignedInt snap record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<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 <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.19** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_  
SnapshotEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t  
contReadPos)`

Callback for the SnapshotEnd snap record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use [OTF2\\_EvtReader\\_Seek](#) with `contReadPos` as the position.

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>contRead-Pos</i>	Position to continue reading in the event trace.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.22.2.20** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_ - SnapshotStart)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t numberOfRecords)`

Callback for the SnapshotStart snap record.

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one [SnapshotStart](#) record and closes with one [SnapshotEnd](#) record. All snapshot records inbetween are ordered by the `origEventTime`, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>num-berOfRecord</i>	Number of snapshot event records in this snapshot. Excluding the <a href="#">SnapshotEnd</a> record.

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.22.2.21** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown snap record.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>snapTime</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterGlobalSnapCallbacks</i></a> or <a href="#"><i>OTF2_GlobalSnapReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this snap.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.22.2.22** `typedef struct OTF2_GlobalSnapReaderCallbacks_struct OTF2_GlobalSnapReaderCallbacks`

Opaque struct which holds all snap record callbacks.

### Since

Version 1.2

**E.22.3 Function Documentation**

**E.22.3.1** void OTF2\_GlobalSnapReaderCallbacks\_Clear ( OTF2\_-  
GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*  
)

Clears a struct for the global snap callbacks.

**Parameters**

<i>global-SnapReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_-GlobalSnapReaderCallbacks_New</a> .
-----------------------------------	--

**Since**

Version 1.2

**E.22.3.2** void OTF2\_GlobalSnapReaderCallbacks\_Delete ( OTF2\_-  
GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*  
)

Deallocates a struct for the global snap callbacks.

**Parameters**

<i>global-SnapReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_-GlobalSnapReaderCallbacks_New</a> .
-----------------------------------	--

**Since**

Version 1.2

**E.22.3.3** OTF2\_GlobalSnapReaderCallbacks\* OTF2\_GlobalSnapReaderCallbacks\_-  
New ( void )

Allocates a new struct for the snap callbacks.

**Since**

Version 1.2



## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Returns

A newly allocated struct of type *OTF2\_GlobalSnapReaderCallbacks*.

**E.22.3.4** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_SetEnterCallback** (  
    **OTF2\_GlobalSnapReaderCallbacks** \* *globalSnapReaderCallbacks*,  
    **OTF2\_GlobalSnapReaderCallback\_Enter** *enterCallback* )

Registers the callback for the Enter snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter snaps.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

**E.22.3.5** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_-SetMeasurementOnOffCallback** (  
    **OTF2\_GlobalSnapReaderCallbacks** \* *globalSnapReaderCallbacks*,  
    **OTF2\_GlobalSnapReaderCallback\_-MeasurementOnOff** *measurementOnOffCallback*  
    )

Registers the callback for the MeasurementOnOff snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>measurementOnOffCallback</i>	Function which should be called for all MeasurementOnOff snaps.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.22.3.6** `OTF2_StatusCode OTF2_GlobalSnapReaderCallbacks_SetMetricCallback ( OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_Metric metricCallback )`

Registers the callback for the Metric snap.

**Parameters**

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric snaps.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.22.3.7** `OTF2_StatusCode OTF2_GlobalSnapReaderCallbacks_SetMpiCollectiveBeginCallback ( OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin mpiCollectiveBeginCallback )`

Registers the callback for the MpiCollectiveBegin snap.

**Parameters**

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

**E.22.3.8 OTF2\_StatusCode OTF2\_GlobalSnapReaderCallbacks\_-SetMpiCollectiveEndCallback ( OTF2\_GlobalSnapReaderCallbacks \* globalSnapReaderCallbacks, OTF2\_GlobalSnapReaderCallback\_ - MpiCollectiveEnd mpiCollectiveEndCallback )**

Registers the callback for the MpiCollectiveEnd snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEndCallback</i>	Function which should be called for all MpiCollectiveEnd snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

**E.22.3.9** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_SetMpiIrecvCallback**  
**( OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*,**  
**OTF2\_GlobalSnapReaderCallback\_MpiIrecv *mpiIrecvCallback* )**

Registers the callback for the MpiIrecv snap.

**Parameters**

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all MpiIrecv snaps.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.22.3.10** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_-SetMpiIrecvRequestCallback**  
**( OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*,**  
**OTF2\_GlobalSnapReaderCallback\_-MpiIrecvRequest *mpiIrecvRequestCallback* )**

Registers the callback for the MpiIrecvRequest snap.

**Parameters**

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all MpiIrecvRequest snaps.

**Since**

Version 1.2

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.22.3.11** **OTF2\_StatusCode** **OTF2\_GlobalSnapReaderCallbacks\_SetMpiSendCallback**  
( **OTF2\_GlobalSnapReaderCallbacks** \* *globalSnapReaderCallbacks*,  
**OTF2\_GlobalSnapReaderCallback\_MpiSend** *mpisendCallback* )

Registers the callback for the MpiSend snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpisend-Callback</i>	Function which should be called for all MpiSend snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.22.3.12** **OTF2\_StatusCode** **OTF2\_GlobalSnapReaderCallbacks\_-SetMpiSendCompleteCallback** ( **OTF2\_GlobalSnapReaderCallbacks** \* *globalSnapReaderCallbacks*, **OTF2\_GlobalSnapReaderCallback\_-MpiSendComplete** *mpisendCompleteCallback* )

Registers the callback for the MpiSendComplete snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
-----------------------------------	---------------------------

## APPENDIX E. FILE DOCUMENTATION

---

<i>mpiIsend-Complete-Callback</i>	Function which should be called for all MpiIsendComplete snaps.
-----------------------------------	---

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.22.3.13** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_SetMpiRecvCallback**  
( **OTF2\_GlobalSnapReaderCallbacks** \* *globalSnapReaderCallbacks*,  
**OTF2\_GlobalSnapReaderCallback\_MpiRecv** *mpiRecvCallback* )

Registers the callback for the MpiRecv snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecv-Callback</i>	Function which should be called for all MpiRecv snaps.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.22.3.14** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_SetMpiSendCallback**  
( **OTF2\_GlobalSnapReaderCallbacks** \* *globalSnapReaderCallbacks*,  
**OTF2\_GlobalSnapReaderCallback\_MpiSend** *mpiSendCallback* )

Registers the callback for the MpiSend snap.

## E.22 oftf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSend-Callback</i>	Function which should be called for all MpiSend snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

**E.22.3.15 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-SetOmpAcquireLockCallback ( OTF2\_GlobalSnapReaderCallbacks \* globalSnapReaderCallbacks, OTF2\_GlobalSnapReaderCallback\_ OmpAcquireLock ompAcquireLockCallback )**

Registers the callback for the OmpAcquireLock snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>ompAcquireLock-Callback</i>	Function which should be called for all OmpAcquireLock snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.22.3.16** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_SetOmpForkCallback**  
( **OTF2\_GlobalSnapReaderCallbacks** \* *globalSnapReaderCallbacks*,  
**OTF2\_GlobalSnapReaderCallback\_OmpFork** *ompForkCallback* )

Registers the callback for the OmpFork snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>ompFork-Callback</i>	Function which should be called for all OmpFork snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.22.3.17** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_-SetOmpTaskCreateCallback** ( **OTF2\_GlobalSnapReaderCallbacks** \* *globalSnapReaderCallbacks*, **OTF2\_GlobalSnapReaderCallback\_-OmpTaskCreate** *ompTaskCreateCallback* )

Registers the callback for the OmpTaskCreate snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskCreate-Callback</i>	Function which should be called for all OmpTaskCreate snaps.

### Since

Version 1.2



## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.22.3.18** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_-SetOmpTaskSwitchCallback** ( **OTF2\_GlobalSnapReaderCallbacks** \* **globalSnapReaderCallbacks**, **OTF2\_GlobalSnapReaderCallback\_-OmpTaskSwitch** **ompTaskSwitchCallback** )

Registers the callback for the OmpTaskSwitch snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskSwitch-Callback</i>	Function which should be called for all OmpTaskSwitch snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.22.3.19** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_-SetParameterIntCallback** ( **OTF2\_GlobalSnapReaderCallbacks** \* **globalSnapReaderCallbacks**, **OTF2\_GlobalSnapReaderCallback\_-ParameterInt** **parameterIntCallback** )

Registers the callback for the ParameterInt snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
-----------------------------------	---------------------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>parameter-IntCallback</i>	Function which should be called for all ParameterInt snaps.
------------------------------	---

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.22.3.20** ***OTF2\_ErrorCode*** ***OTF2\_GlobalSnapReaderCallbacks\_-SetParameterStringCallback*** ( ***OTF2\_GlobalSnapReaderCallbacks \**** ***globalSnapReaderCallbacks***, ***OTF2\_GlobalSnapReaderCallback\_-ParameterString parameterStringCallback*** )

Registers the callback for the ParameterString snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>parameter-StringCallback</i>	Function which should be called for all ParameterString snaps.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## E.22 otf2/OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

**E.22.3.21** **OTF2\_***ErrorCode* **OTF2\_GlobalSnapReaderCallbacks\_**  
**SetParameterUnsignedIntCallback** ( **OTF2\_GlobalSnapReaderCallbacks**  
**\*** *globalSnapReaderCallbacks*, **OTF2\_GlobalSnapReaderCallback\_**  
**ParameterUnsignedInt** *parameterUnsignedIntCallback*  
**)**

Registers the callback for the ParameterUnsignedInt snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt snaps.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid defReaderCallbacks argument

**E.22.3.22** **OTF2\_***ErrorCode* **OTF2\_GlobalSnapReaderCallbacks\_**  
**SetSnapshotEndCallback** ( **OTF2\_GlobalSnapReaderCallbacks** **\***  
*globalSnapReaderCallbacks*, **OTF2\_GlobalSnapReaderCallback\_**  
**SnapshotEnd** *snapshotEndCallback* )

Registers the callback for the SnapshotEnd snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>snapshotEndCallback</i>	Function which should be called for all SnapshotEnd snaps.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.22.3.23** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_**  
**SetSnapshotStartCallback (** **OTF2\_GlobalSnapReaderCallbacks**  
**\* *globalSnapReaderCallbacks*, OTF2\_GlobalSnapReaderCallback\_**  
**SnapshotStart *snapshotStartCallback* )**

Registers the callback for the SnapshotStart snap.

**Parameters**

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>snapshot-StartCallback</i>	Function which should be called for all SnapshotStart snaps.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.22.3.24** **OTF2\_ErrorCode** **OTF2\_GlobalSnapReaderCallbacks\_SetUnknownCallback**  
**(** **OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*,**  
**OTF2\_GlobalSnapReaderCallback\_Unknown *unknownCallback* )**

Registers the callback for unknown snaps.

**Parameters**

### E.23 otf2/OTF2\_IdMap.h File Reference

---

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	Function which should be called for all unknown snaps.

#### Since

Version 1.2

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful  
[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid `defReaderCallbacks` argument

### E.23 otf2/OTF2\_IdMap.h File Reference

Identifier mapping data structure, based on Scalasca's `epk_idmap.h`.

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <otf2/OTF2_ErrorCodes.h>
```

#### Typedefs

- typedef struct OTF2\_IdMap\_struct [\*OTF2\\_IdMap\*](#)
- typedef void(\* [\*OTF2\\_IdMap\\_TraverseCallback\*](#))(uint64\_t localId, uint64\_t globalId, void \*userData)  
*Function prototype for use in OTF2\_IdMap\_Traverse.*
- typedef uint8\_t [\*OTF2\\_IdMapMode\*](#)

#### Enumerations

- enum [\*OTF2\\_IdMapMode\\_enum\*](#) {  
    [\*OTF2\\_ID\\_MAP\\_DENSE\*](#),  
    [\*OTF2\\_ID\\_MAP\\_SPARSE\*](#) }

## Functions

- [OTF2\\_ErrorCode OTF2\\_IdMap\\_AddIdPair](#) ([OTF2\\_IdMap](#) \*instance, uint64\_t localId, uint64\_t globalId)
- [OTF2\\_ErrorCode OTF2\\_IdMap\\_Clear](#) ([OTF2\\_IdMap](#) \*instance)
- [OTF2\\_IdMap](#) \* [OTF2\\_IdMap\\_Create](#) ([OTF2\\_IdMapMode](#) mode, uint64\_t capacity)
- [OTF2\\_IdMap](#) \* [OTF2\\_IdMap\\_CreateFromUint32Array](#) (uint64\_t length, const uint32\_t \*mappings, bool optimizeSize)
- [OTF2\\_IdMap](#) \* [OTF2\\_IdMap\\_CreateFromUint64Array](#) (uint64\_t length, const uint64\_t \*mappings, bool optimizeSize)
- void [OTF2\\_IdMap\\_Free](#) ([OTF2\\_IdMap](#) \*instance)
- [OTF2\\_ErrorCode OTF2\\_IdMap\\_GetGlobalId](#) (const [OTF2\\_IdMap](#) \*instance, uint64\_t localId, uint64\_t \*globalId)
- [OTF2\\_ErrorCode OTF2\\_IdMap\\_GetGlobalIdSave](#) (const [OTF2\\_IdMap](#) \*instance, uint64\_t localId, uint64\_t \*globalId)
- [OTF2\\_ErrorCode OTF2\\_IdMap\\_GetMode](#) (const [OTF2\\_IdMap](#) \*instance, [OTF2\\_IdMapMode](#) \*mode)
- [OTF2\\_ErrorCode OTF2\\_IdMap\\_GetSize](#) (const [OTF2\\_IdMap](#) \*instance, uint64\_t \*size)
- [OTF2\\_ErrorCode OTF2\\_IdMap\\_Traverse](#) (const [OTF2\\_IdMap](#) \*instance, [OTF2\\_IdMap\\_TraverseCallback](#) callback, void \*userData)

### E.23.1 Detailed Description

Identifier mapping data structure, based on Scalasca's `epk_idmap.h`. This file provides type definitions and function prototypes for an identifier mapping data structure which is used to store mapping tables for converting local into global identifiers.

This mapping data structure can operate in two different modes (see [OTF2\\_IdMapMode](#)): A dense mapping can be used if the local identifiers are consecutively enumerated from 0 to N-1. In this case, only the global identifier are stored in the table at the corresponding entry, leading to compact storage and fast look-up. By contrast, if the local identifiers can consist of arbitrary numbers, a sparse mapping is necessary. Here, (localId, globalId) tuples are stored, which requires a more complicated look-up procedure.

### E.23.2 Typedef Documentation

#### E.23.2.1 typedef struct OTF2\_IdMap\_struct OTF2\_IdMap

Opaque data structure representing an ID mapping table.

## E.23 otf2/OTF2\_IdMap.h File Reference

---

### E.23.2.2 typedef uint8\_t OTF2\_IdMapMode

Wrapper around enum OTF2\_IdMapMode\_enum, so that it is guaranteed that it is a uint8\_t

### E.23.3 Enumeration Type Documentation

#### E.23.3.1 enum OTF2\_IdMapMode\_enum

Enumeration type defining the two different modes of an identifier mapping table.

**Enumerator:**

*OTF2\_ID\_MAP\_DENSE* Dense mapping table  
*OTF2\_ID\_MAP\_SPARSE* Sparse mapping table

### E.23.4 Function Documentation

#### E.23.4.1 OTF2\_ErrorCode OTF2\_IdMap\_AddIdPair ( OTF2\_IdMap \* *instance*, uint64\_t *localId*, uint64\_t *globalId* )

Adds the given mapping from *localId* to *globalId* to the mapping table *instance*. If the current capacity does not suffice, the data structure is automatically resized.

**Note**

If the mapping table operates in dense mapping mode, the parameter *localId* has to correspond to the next entry in the mapping table.

**Parameters**

<i>instance</i>	Object to add the mapping to.
<i>localId</i>	Local identifier.
<i>globalId</i>	Global identifier.

**Returns**

OTF2\_SUCCESS, or error code.

#### E.23.4.2 OTF2\_ErrorCode OTF2\_IdMap\_Clear ( OTF2\_IdMap \* *instance* )

Removes all entries in the given mapping table *instance*. It can be used, e.g., to reuse an mapping table object for new input data.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>instance</i>	Object to remove entries from.
-----------------	--------------------------------

### Returns

OTF2\_SUCCESS, or error code.

#### E.23.4.3 OTF2\_IdMap\* OTF2\_IdMap.Create ( OTF2\_IdMapMode *mode*, uint64\_t *capacity* )

Creates and returns a new instance of OTF2\_IdMap with the given *mode* and initial *capacity*. If the memory allocation request cannot be fulfilled, NULL is returned.

### Parameters

<i>mode</i>	Mapping mode.
<i>capacity</i>	Initial capacity.

### Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.

#### E.23.4.4 OTF2\_IdMap\* OTF2\_IdMap.CreateFromUint32Array ( uint64\_t *length*, const uint32\_t \* *mappings*, bool *optimizeSize* )

Creates and returns a new instance of OTF2\_IdMap from the array given by *mappings*.

Same as *OTF2\_IdMap\_CreateFromUint64Array*, excpet from a uint32\_t array.

### Parameters

<i>length</i>	Number of elements in the <i>mappings</i> array.
<i>mappings</i>	Array with a dense mapping.
<i>optimize-Size</i>	Creates a SPARSE mapping, if the number of non- identities is less than half the array length.

### Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.



## E.23 otf2/OTF2\_IdMap.h File Reference

---

### E.23.4.5 OTF2\_IdMap\* OTF2\_IdMap.CreateFromUint64Array ( uint64\_t *length*, const uint64\_t \* *mappings*, bool *optimizeSize* )

Creates and returns a new instance of OTF2\_IdMap from the array given by *mappings*.

This creates always a DENSE mapping if *optimizeSize* is false. If it is true, it creates a SPARSE mapping, if the number of non-identity entries in the *mappings* array (ie. `mapping[ i ] != i`) is less than half the *length*.

Returns NULL when *optimizeSize* is true and the number of non-identity entries equals zero, ie. the given map is the identity map.

#### Parameters

<i>length</i>	Number of elements in the <i>mappings</i> array.
<i>mappings</i>	Array with a dense mapping.
<i>optimize-Size</i>	Creates a SPARSE mapping, if the number of non- identities is less than half the array length.

#### Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.

### E.23.4.6 void OTF2\_IdMap.Free ( OTF2\_IdMap \* *instance* )

Destroys the given *instance* of OTF2\_IdMap and releases the allocated memory.

#### Parameters

<i>instance</i>	Object to be freed
-----------------	--------------------

### E.23.4.7 OTF2\_ErrorCode OTF2\_IdMap.GetGlobalId ( const OTF2\_IdMap \* *instance*, uint64\_t *localId*, uint64\_t \* *globalId* )

Maps the given *localId* to the global id and store it in the starge provide by *globalId*.

If the given *localId* is not in the mapping, sets *globalId* to the *localId*.

#### Parameters

	<i>instance</i>	Object to add the mapping to.
	<i>localId</i>	Local identifier.
out	<i>globalId</i>	Global identifier.

**Returns**

OTF2\_SUCCESS, or error code.

**E.23.4.8** **OTF2\_ErrorCode** **OTF2\_IdMap\_GetGlobalIdSave** ( **const** **OTF2\_IdMap** \* *instance*, **uint64\_t** *localId*, **uint64\_t** \* *globalId* )

Maps the given *localId* to the global id and store it in the storage provided by *globalId*.

If the given *localId* is not in the mapping, returns [\*OTF2\\_ERROR\\_INDEX\\_OUT\\_OF\\_BOUNDS\*](#).

**Parameters**

	<i>instance</i>	Object to add the mapping to.
	<i>localId</i>	Local identifier.
out	<i>globalId</i>	Global identifier.

**Returns**

OTF2\_SUCCESS, or error code.

**E.23.4.9** **OTF2\_ErrorCode** **OTF2\_IdMap\_GetMode** ( **const** **OTF2\_IdMap** \* *instance*, **OTF2\_IdMapMode** \* *mode* )

Returns the identifier mapping mode (dense/sparse) used for the given mapping table *instance*.

**Parameters**

	<i>instance</i>	Queried object.
out	<i>mode</i>	Identifier mapping mode.

**Returns**

OTF2\_SUCCESS, or error code.

**E.23.4.10** **OTF2\_ErrorCode** **OTF2\_IdMap\_GetSize** ( **const** **OTF2\_IdMap** \* *instance*, **uint64\_t** \* *size* )

Returns the actual number of entries stored in the given **OTF2\_IdMap** *instance*.

## E.24 otf2/OTF2\_Marker.h File Reference

---

### Parameters

	<i>instance</i>	Queried object.
out	<i>size</i>	Number of entries.

### Returns

OTF2\_SUCCESS, or error code.

#### E.23.4.11 **OTF2\_ErrorCode** OTF2\_IdMap\_Traverse ( const OTF2\_IdMap \* *instance*, OTF2\_IdMap\_TraverseCallback *callback*, void \* *userData* )

Calls for each mapping pair the callback *callback*.

### Parameters

<i>instance</i>	Object to add the mapping to.
<i>callback</i>	Callback function which is called for eaach mapping pair.
<i>userData</i>	Data which is passed to the <i>callback</i> function.

### Returns

OTF2\_SUCCESS, or error code.

## E.24 otf2/OTF2\_Marker.h File Reference

This file provides types and enums for markers.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
```

### Defines

- #define **OTF2\_UNDEFINED\_MARKER** ( ( OTF2\_MarkerRef ) OTF2\_UNDEFINED\_UINT32 )

*The invalid value for a reference to a [DefMarker](#) definition.*

### Typedefs

- typedef uint32\_t **OTF2\_MarkerRef**

*Type used to indicate a reference to a [DefMarker](#) definition.*

- typedef uint8\_t [OTF2\\_MarkerScope](#)  
*Wrapper for enum [OTF2\\_MarkerScope\\_enum](#).*
- typedef uint8\_t [OTF2\\_MarkerSeverity](#)  
*Wrapper for enum [OTF2\\_MarkerSeverity\\_enum](#).*

### Enumerations

- enum [OTF2\\_MarkerScope\\_enum](#) {  
    [OTF2\\_MARKER\\_SCOPE\\_GLOBAL](#),  
    [OTF2\\_MARKER\\_SCOPE\\_LOCATION](#),  
    [OTF2\\_MARKER\\_SCOPE\\_LOCATION\\_GROUP](#),  
    [OTF2\\_MARKER\\_SCOPE\\_SYSTEM\\_TREE\\_NODE](#),  
    [OTF2\\_MARKER\\_SCOPE\\_GROUP](#),  
    [OTF2\\_MARKER\\_SCOPE\\_COMM](#) }  
• enum [OTF2\\_MarkerSeverity\\_enum](#) {  
    [OTF2\\_SEVERITY\\_NONE](#),  
    [OTF2\\_SEVERITY\\_LOW](#),  
    [OTF2\\_SEVERITY\\_MEDIUM](#),  
    [OTF2\\_SEVERITY\\_HIGH](#) }

#### E.24.1 Detailed Description

This file provides types and enums for markers.

#### E.24.2 Enumeration Type Documentation

##### E.24.2.1 enum [OTF2\\_MarkerScope\\_enum](#)

A user marker does have a scope of it validity.

##### Enumerator:

**[OTF2\\_MARKER\\_SCOPE\\_GLOBAL](#)** The user marker has a global scope (could also be NONE).

**[OTF2\\_MARKER\\_SCOPE\\_LOCATION](#)** The user marker has a scope of a location.

**[OTF2\\_MARKER\\_SCOPE\\_LOCATION\\_GROUP](#)** The user marker has a scope of a location group.

## E.25 otf2/OTF2\_MarkerReader.h File Reference

---

**OTF2\_MARKER\_SCOPE\_SYSTEM\_TREE\_NODE** The user marker has a scope of a system tree.

**OTF2\_MARKER\_SCOPE\_GROUP** The user marker has a scope of a group.

**OTF2\_MARKER\_SCOPE\_COMM** The user marker has a scope of a communicator.

### E.24.2.2 enum OTF2\_MarkerSeverity\_enum

A list of possible severities of user markers.

**Enumerator:**

**OTF2\_SEVERITY\_NONE** The marker does not have a severity.

**OTF2\_SEVERITY\_LOW** The marker has a low severity.

**OTF2\_SEVERITY\_MEDIUM** The marker has a medium severity.

**OTF2\_SEVERITY\_HIGH** The marker has a high severity.

## E.25 otf2/OTF2\_MarkerReader.h File Reference

This file provides all routines that read marker records.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Marker.h>
#include <otf2/OTF2_MarkerReaderCallbacks.h>
```

### Functions

- [OTF2\\_ErrorCode OTF2\\_MarkerReader\\_ReadMarkers](#) ([OTF2\\_MarkerReader](#) \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)

*After callback registration, the markers could be read with the following function. The user of this function tells the system how many markers it is able to handle (recordsToRead) and the function returns how many markers where in the stream (recordsRead). It should usually be the case that both values are the same. If this is not the case, then there where less records than requested in the stream.*

- [OTF2\\_ErrorCode OTF2\\_MarkerReader\\_SetCallbacks](#) ([OTF2\\_MarkerReader](#) \*reader, const [OTF2\\_MarkerReaderCallbacks](#) \*callbacks, void \*userData)

---

## APPENDIX E. FILE DOCUMENTATION

---

*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### E.25.1 Detailed Description

This file provides all routines that read marker records.

### E.25.2 Function Documentation

**E.25.2.1** `OTF2_ErrorCode OTF2_MarkerReader_ReadMarkers (`  
`OTF2_MarkerReader * reader, uint64_t recordsToRead, uint64_t *`  
`recordsRead )`

After callback registration, the markers could be read with the following function. The user of this function tells the system how many markers it is able to handle (recordsToRead) and the function returns how many markers where in the stream (recordsRead). It should usually be the case that both values are the same. If this is not the case, then there where less records than requested in the stream.

#### Parameters

<i>reader</i>	Reader Object.
<i>recordsToRead</i>	How many records have to be read next.
<i>recordsRead</i>	How many records where read?

#### Since

Version 1.2

#### Returns

OTF2\_ErrorCode with !=OTF2\_SUCCESS if there was an error.

**E.25.2.2** `OTF2_ErrorCode OTF2_MarkerReader_SetCallbacks (`  
`OTF2_MarkerReader * reader, const OTF2_MarkerReaderCallbacks`  
`* callbacks, void * userData )`

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function.

## E.26 otf2/OTF2\_MarkerReaderCallbacks.h File Reference

---

Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

### Parameters

<i>reader</i>	This given reader object will be setted up with new callback functions.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#">OTF2_MarkerReaderCallbacks_New</a> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.26 otf2/OTF2\_MarkerReaderCallbacks.h File Reference

This defines the callbacks for the marker reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_IdMap.h>
#include <otf2/OTF2_Marker.h>
```

### Typedefs

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_MarkerReaderCallback\\_DefMarker](#))(void \*userData, [OTF2\\_MarkerRef](#) self, const char \*markerGroup, const char \*markerCategory, [OTF2\\_MarkerSeverity](#) severity)  
*Function pointer definition for the callback which is triggered by a [DefMarker](#) definition record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_MarkerReaderCallback\\_Marker](#))(void \*userData, [OTF2\\_TimeStamp](#) timestamp, [OTF2\\_TimeStamp](#) duration, [OTF2\\_MarkerRef](#) marker, [OTF2\\_MarkerScope](#) scope, uint64\_t scopeRef, const char \*text)  
*Function pointer definition for the callback which is triggered by a [Marker](#) record.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_MarkerReaderCallback\\_Unknown](#))(void \*userData)

*Function pointer definition for the callback which is triggered for an unknown marker.*

- typedef struct [OTF2\\_MarkerReaderCallbacks\\_struct](#) [OTF2\\_MarkerReaderCallbacks](#)

*Opaque struct which holds all definition record callbacks.*

### Functions

- void [OTF2\\_MarkerReaderCallbacks\\_Clear](#) ([OTF2\\_MarkerReaderCallbacks](#) \*markerReaderCallbacks)

*Clears a struct for the marker callbacks.*

- void [OTF2\\_MarkerReaderCallbacks\\_Delete](#) ([OTF2\\_MarkerReaderCallbacks](#) \*markerReaderCallbacks)

*Deallocates a struct for the marker callbacks.*

- [OTF2\\_MarkerReaderCallbacks](#) \* [OTF2\\_MarkerReaderCallbacks\\_New](#) (void)

*Allocates a new struct for the marker callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_MarkerReaderCallbacks\\_SetDefMarkerCallback](#) ([OTF2\\_MarkerReaderCallbacks](#) \*markerReaderCallbacks, [OTF2\\_MarkerReaderCallback\\_DefMarker](#) defMarkerCallback)

*Registers the callback for the [DefMarker](#) definition.*

- [OTF2\\_ErrorCode](#) [OTF2\\_MarkerReaderCallbacks\\_SetMarkerCallback](#) ([OTF2\\_MarkerReaderCallbacks](#) \*markerReaderCallbacks, [OTF2\\_MarkerReaderCallback\\_Marker](#) markerCallback)

*Registers the callback for the [Marker](#) record.*

- [OTF2\\_ErrorCode](#) [OTF2\\_MarkerReaderCallbacks\\_SetUnknownCallback](#) ([OTF2\\_MarkerReaderCallbacks](#) \*markerReaderCallbacks, [OTF2\\_MarkerReaderCallback\\_Unknown](#) unknownCallback)

*Registers the callback for an unknown marker.*

### E.26.1 Detailed Description

This defines the callbacks for the marker reader.



## E.26 otf2/OTF2\_MarkerReaderCallbacks.h File Reference

---

### E.26.2 Typedef Documentation

**E.26.2.1** `typedef OTF2_CallbackCode( * OTF2_MarkerReaderCallback_  
DefMarker)(void *userData, OTF2_MarkerRef self, const char  
*markerGroup, const char *markerCategory, OTF2_MarkerSeverity severity)`

Function pointer definition for the callback which is triggered by a [DefMarker](#) definition record.

#### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterMarkerCallbacks</a> or <a href="#">OTF2_MarkerReader_SetCallbacks</a> .
<i>self</i>	Reference to this marker definition.
<i>marker-Group</i>	Group name, e.g., "MUST", ...
<i>markerCategory</i>	Marker category, e.g., "Argument type error", ... The tuple (marker-Group, markerCategory) must be unique over all marker definitions.
<i>severity</i>	The severity for this marker category.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.26.2.2** `typedef OTF2_CallbackCode( * OTF2_MarkerReaderCallback_  
Marker)(void *userData, OTF2_TimeStamp timestamp,  
OTF2_TimeStamp duration, OTF2_MarkerRef marker,  
OTF2_MarkerScope scope, uint64_t scopeRef, const char *text)`

Function pointer definition for the callback which is triggered by a [Marker](#) record.

#### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterMarkerCallbacks</a> or <a href="#">OTF2_MarkerReader_SetCallbacks</a> .
<i>timestamp</i>	Timestamp of the marker.
<i>duration</i>	Duration the marker applies.
<i>marker</i>	Reference to the marker definition.
<i>scope</i>	The type of scope of this marker instance.
<i>scopeRef</i>	The reference to an element of the scope of this marker. Depends on <i>scope</i> .
<i>text</i>	A textual description for this marker.

**Since**

Version 1.2

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.26.2.3 typedef OTF2\_CallbackCode( \* OTF2\_MarkerReaderCallback\_Unknown)(void \*userData)**

Function pointer definition for the callback which is triggered for an unknown marker.

**Parameters**

<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterMarkerCallbacks</i> or <i>OTF2_MarkerReader_SetCallbacks</i> .
-----------------	---

**Since**

Version 1.2

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.26.3 Function Documentation****E.26.3.1 void OTF2\_MarkerReaderCallbacks\_Clear ( OTF2\_MarkerReaderCallbacks \* markerReaderCallbacks )**

Clears a struct for the marker callbacks.

**Since**

Version 1.2

**Parameters**

<i>marker-Reader-Callbacks</i>	Handle to a struct previously allocated with <i>OTF2_MarkerReaderCallbacks_New</i> .
--------------------------------	--

## E.26 otf2/OTF2\_MarkerReaderCallbacks.h File Reference

---

**E.26.3.2** void OTF2\_MarkerReaderCallbacks\_Delete ( OTF2\_MarkerReaderCallbacks  
\* *markerReaderCallbacks* )

Deallocates a struct for the marker callbacks.

### Since

Version 1.2

### Parameters

<i>marker-Reader-Callbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_MarkerReaderCallbacks_New</a> .
--------------------------------	---

**E.26.3.3** OTF2\_MarkerReaderCallbacks\* OTF2\_MarkerReaderCallbacks\_New ( void  
)

Allocates a new struct for the marker callbacks.

### Since

Version 1.2

### Returns

A newly allocated struct of type [OTF2\\_MarkerReaderCallbacks](#).

**E.26.3.4** OTF2\_ErrorCode OTF2\_MarkerReaderCallbacks\_SetDefMarkerCallback  
( OTF2\_MarkerReaderCallbacks \* *markerReaderCallbacks*,  
OTF2\_MarkerReaderCallback\_DefMarker *defMarkerCallback* )

Registers the callback for the [DefMarker](#) definition.

### Parameters

<i>marker-Reader-Callbacks</i>	Struct for all callbacks.
<i>defMarker-Callback</i>	Function which should be called for all <a href="#">DefMarker</a> definitions.

## APPENDIX E. FILE DOCUMENTATION

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.26.3.5** **OTF2\_StatusCode** **OTF2\_MarkerReaderCallbacks\_SetMarkerCallback**  
( **OTF2\_MarkerReaderCallbacks** \* *markerReaderCallbacks*,  
**OTF2\_MarkerReaderCallback\_Marker** *markerCallback* )

Registers the callback for the *Marker* record.

### Parameters

<i>marker-Reader-Callbacks</i>	Struct for all callbacks.
<i>marker-Callback</i>	Function which should be called for all <i>Marker</i> records.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.26.3.6** **OTF2\_StatusCode** **OTF2\_MarkerReaderCallbacks\_SetUnknownCallback**  
( **OTF2\_MarkerReaderCallbacks** \* *markerReaderCallbacks*,  
**OTF2\_MarkerReaderCallback\_Unknown** *unknownCallback* )

Registers the callback for an unknown marker.

### Parameters

<i>marker-Reader-Callbacks</i>	Struct for all callbacks.

## E.27 otf2/OTF2\_MarkerWriter.h File Reference

---

<i>unknown-Callback</i>	Function which should be called for all unknown definitions.
-------------------------	--

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*](#) for an invalid `defReaderCallbacks` argument

## E.27 otf2/OTF2\_MarkerWriter.h File Reference

This file provides all routines that write marker records.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_Marker.h>
```

### Typedefs

- typedef struct `OTF2_MarkerWriter_struct` [\*OTF2\\_MarkerWriter\*](#)  
*Handle definition for the external marker writer.*

### Functions

- [\*OTF2\\_ErrorCode\*](#) [\*OTF2\\_MarkerWriter\\_WriteDefMarker\*](#) ([\*OTF2\\_MarkerWriter\*](#) \*writerHandle, [\*OTF2\\_MarkerRef\*](#) self, const char \*markerGroup, const char \*markerCategory, [\*OTF2\\_MarkerSeverity\*](#) severity)  
*Write a marker definition.*
- [\*OTF2\\_ErrorCode\*](#) [\*OTF2\\_MarkerWriter\\_WriteMarker\*](#) ([\*OTF2\\_MarkerWriter\*](#) \*writerHandle, [\*OTF2\\_TimeStamp\*](#) timestamp, [\*OTF2\\_TimeStamp\*](#) duration, [\*OTF2\\_MarkerRef\*](#) marker, [\*OTF2\\_MarkerScope\*](#) scope, uint64\_t scopeRef, const char \*text)  
*Write a marker record.*

**E.27.1 Detailed Description**

This file provides all routines that write marker records.

**E.27.2 Function Documentation**

**E.27.2.1** **OTF2\_ErrorCode** **OTF2\_MarkerWriter\_WriteDefMarker** (  
    **OTF2\_MarkerWriter** \* *writerHandle*, **OTF2\_MarkerRef** *self*, **const**  
    **char** \* *markerGroup*, **const char** \* *markerCategory*, **OTF2\_MarkerSeverity**  
    *severity* )

Write a marker definition.

**Parameters**

<i>writerHandle</i>	Marker writer handle.
<i>self</i>	Reference to this marker definition.
<i>markerGroup</i>	Group name, e.g., "MUST", ...
<i>markerCategory</i>	Marker category, e.g., "Argument type error", ... The tuple (markerGroup, markerCategory) must be unique over all marker definitions.
<i>severity</i>	The severity for this marker category.

**Since**

Version 1.2

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.27.2.2** **OTF2\_ErrorCode** **OTF2\_MarkerWriter\_WriteMarker** (  
    **OTF2\_MarkerWriter** \* *writerHandle*, **OTF2\_TimeStamp**  
    *timestamp*, **OTF2\_TimeStamp** *duration*, **OTF2\_MarkerRef** *marker*,  
    **OTF2\_MarkerScope** *scope*, **uint64\_t** *scopeRef*, **const char** \* *text* )

Write a marker record.

**Parameters**

<i>writerHandle</i>	Marker writer handle.
<i>timestamp</i>	Time of the marker.

## E.28 otf2/OTF2\_MPI\_Collectives.h File Reference

---

<i>duration</i>	A possible duration of this marker. May be 0.
<i>marker</i>	Reference to a marker definition.
<i>scope</i>	The type of scope of this marker instance: <a href="#">OTF2_MARKER_SCOPE_GLOBAL</a> , <a href="#">OTF2_MARKER_SCOPE_LOCATION</a> , <a href="#">OTF2_MARKER_SCOPE_LOCATION_GROUP</a> , <a href="#">OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE</a> , <a href="#">OTF2_MARKER_SCOPE_GROUP</a> , or <a href="#">OTF2_MARKER_SCOPE_COMM</a> .
<i>scopeRef</i>	The reference to an element of the scope of this marker. Depends on <i>scope</i> .
<i>text</i>	A textual description for this marker.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.28 otf2/OTF2\_MPI\_Collectives.h File Reference

MPI collectives for OTF2.

```
#include <otf2/otf2.h>
```

```
#include <mpi.h>
```

### Data Structures

- struct [OTF2\\_CollectiveContext](#)  
*Collective context which wraps an MPI communicator.*
- struct [OTF2\\_MPI\\_UserData](#)  
*User data structure, which will be used by the MPI collectives.*

### Defines

- #define [OTF2\\_MPI\\_DOUBLE](#) MPI\_DOUBLE  
*Define this macro to an suitable MPI datatype to be used for double before including the header. MPI\_DOUBLE is used as proper default value.*
- #define [OTF2\\_MPI\\_FLOAT](#) MPI\_FLOAT  
*Define this macro to an suitable MPI datatype to be used for float before including the header. MPI\_FLOAT is used as proper default value.*

- `#define OTF2_MPI_INT16_T MPI_SHORT`  
*Define this macro to an suitable MPI datatype to be used for `int32_t` before including the header. `MPI_SHORT` or `MPI_INT32_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.*
- `#define OTF2_MPI_INT32_T MPI_INT`  
*Define this macro to an suitable MPI datatype to be used for `int32_t` before including the header. `MPI_INT` or `MPI_INT32_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.*
- `#define OTF2_MPI_INT64_T MPI_INT64_T`  
*Define this macro to an suitable MPI datatype to be used for `int64_t` before including the header. `MPI_INT64_T` is used as proper default value in case of an MPI 3.0 conforming implementation.*
- `#define OTF2_MPI_INT8_T MPI_CHAR`  
*Define this macro to an suitable MPI datatype to be used for `int8_t` before including the header. `MPI_CHAR` or `MPI_INT8_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.*
- `#define OTF2_MPI_UINT16_T MPI_UNSIGNED_SHORT`  
*Define this macro to an suitable MPI datatype to be used for `uint16_t` before including the header. `MPI_UNSIGNED_SHORT` or `MPI_UINT16_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.*
- `#define OTF2_MPI_UINT32_T MPI_UNSIGNED`  
*Define this macro to an suitable MPI datatype to be used for `uint32_t` before including the header. `MPI_UNSIGNED` or `MPI_UINT32_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.*
- `#define OTF2_MPI_UINT64_T MPI_UINT64_T`  
*Define this macro to an suitable MPI datatype to be used for `uint64_t` before including the header. `MPI_UINT64_T` is used as proper default value in case of an MPI 3.0 conforming implementation.*
- `#define OTF2_MPI_UINT8_T MPI_UNSIGNED_CHAR`  
*Define this macro to an suitable MPI datatype to be used for `uint8_t` before including the header. `MPI_UNSIGNED_CHAR` or `MPI_UINT8_T` are used as proper default value. The latter in case of an MPI 3.0 conforming implementation.*
- `#define OTF2_MPI_USE_PMPI`  
*If you want that the collectives call the PMPI interface, define this macro before including the header.*

### Functions

- static `OTF2_ErrorCode OTF2_MPI_Archive_SetCollectiveCallbacks (OTF2_Archive *archive, MPI_Comm globalComm, MPI_Comm localComm)`  
*Register an MPI collective context to an OTF2 archive.*



## E.28 otf2/OTF2\_MPI\_Collectives.h File Reference

---

- static [OTF2\\_ErrorCode](#) [OTF2\\_MPI\\_Archive\\_SetCollectiveCallbacksSplit](#) ([OTF2\\_Archive](#) \*archive, MPI\_Comm globalComm, uint32\_t numberOfFiles)  
*Register an MPI collective context to an OTF2 archive.*
- static [OTF2\\_ErrorCode](#) [OTF2\\_MPI\\_Reader\\_SetCollectiveCallbacks](#) ([OTF2\\_Reader](#) \*reader, MPI\_Comm globalComm)  
*Register an MPI collective context to an OTF2 reader.*

### E.28.1 Detailed Description

MPI collectives for OTF2. See [Usage in reading mode - MPI example](#) and [Usage in writing mode - MPI example](#) for instruction how to use.

### E.28.2 Function Documentation

**E.28.2.1** static [OTF2\\_ErrorCode](#) [OTF2\\_MPI\\_Archive\\_SetCollectiveCallbacks](#) (  
[OTF2\\_Archive](#) \* archive, MPI\_Comm globalComm, MPI\_Comm localComm )  
[static]

Register an MPI collective context to an OTF2 archive.

#### Parameters

<i>archive</i>	The archive handle.
<i>global-Comm</i>	The global communicator to use. Will be duplicated.
<i>localComm</i>	The local communicator to use. Maybe MPI_COMM_NULL, otherwise all localComm must be disjoint and join to globalComm. Will be duplicated.

#### Returns

Success or error code.

**E.28.2.2** static [OTF2\\_ErrorCode](#) [OTF2\\_MPI\\_Archive\\_SetCollectiveCallbacksSplit](#) (  
[OTF2\\_Archive](#) \* archive, MPI\_Comm globalComm, uint32\_t numberOfFiles )  
[static]

Register an MPI collective context to an OTF2 archive.

#### Parameters

<i>archive</i>	The archive handle.
----------------	---------------------

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>global-Comm</i>	The global communicator to use. Will be duplicated.
<i>numberOfFiles</i>	Splits the <code>globalComm</code> into <code>numberOfFiles</code> disjoint sub-communicators and evenly distribute the ranks among them.

### Returns

Success or error code.

**E.28.2.3** `static OTF2_ErrorCode OTF2_MPI_Reader_SetCollectiveCallbacks (`  
`OTF2_Reader * reader, MPI_Comm globalComm ) [static]`

Register an MPI collective context to an OTF2 reader.

### Parameters

<i>reader</i>	The reader handle.
<i>global-Comm</i>	The global communicator to use. Will be duplicated.

### Returns

Success or error code.

## E.29 otf2/OTF2\_Reader.h File Reference

Reading interface for OTF2 archives.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Archive.h>
```

### Typedefs

- `typedef struct OTF2_Reader_struct OTF2_Reader`  
*Keeps all necessary information for the reader.*

### Functions

- `OTF2_ErrorCode OTF2_Reader_Close (OTF2_Reader *reader)`

## E.29 otf2/OTF2\_Reader.h File Reference

---

*Close a reader handle.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseDefFiles](#) (OTF2\_Reader \*reader)

*Closes the local definitions file container.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseDefReader](#) (OTF2\_Reader \*reader, OTF2\_DefReader \*defReader)

*Close a local definition reader.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseEvtFiles](#) (OTF2\_Reader \*reader)

*Closes the events file container.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseEvtReader](#) (OTF2\_Reader \*reader, OTF2\_EvtReader \*evtReader)

*Close a local event reader.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseGlobalDefReader](#) (OTF2\_Reader \*reader, OTF2\_GlobalDefReader \*globalDefReader)

*Closes the global definition reader.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseGlobalEvtReader](#) (OTF2\_Reader \*reader, OTF2\_GlobalEvtReader \*globalEvtReader)

*Closes the global event reader.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseGlobalSnapReader](#) (OTF2\_Reader \*reader, OTF2\_GlobalSnapReader \*globalSnapReader)

*Closes the global snapshot reader.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseMarkerReader](#) (OTF2\_Reader \*reader, OTF2\_MarkerReader \*markerReader)

*Closes the marker reader.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseMarkerWriter](#) (OTF2\_Reader \*reader, OTF2\_MarkerWriter \*markerWriter)

*Closes the marker writer.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseSnapFiles](#) (OTF2\_Reader \*reader)

*Closes the snapshots file container.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseSnapReader](#) (OTF2\_Reader \*reader, OTF2\_SnapReader \*snapReader)

*Close a local snapshot reader.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_CloseThumbReader](#) (OTF2\_Reader \*reader, OTF2\_ThumbReader \*thumbReader)

*Close an opened thumbnail reader.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetBoolProperty](#) (OTF2\_Reader \*reader, const char \*name, bool \*value)

*Get the value of the named trace file property as boolean.*

- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetChunkSize](#) (OTF2\_Reader \*reader, uint64\_t \*chunkSizeEvents, uint64\_t \*chunkSizeDefinitions)

*Get event and definition chunk sizes.*

- [OTF2\\_StatusCode](#) [OTF2\\_Reader\\_GetCompression](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_Compression](#) \*compression)

*Get compression mode.*

- [OTF2\\_StatusCode](#) [OTF2\\_Reader\\_GetCreator](#) ([OTF2\\_Reader](#) \*reader, char \*\*creator)

*Get creator name.*

- [OTF2\\_DefReader](#) \* [OTF2\\_Reader\\_GetDefReader](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_LocationRef](#) location)

*Get a local definition reader.*

- [OTF2\\_StatusCode](#) [OTF2\\_Reader\\_GetDescription](#) ([OTF2\\_Reader](#) \*reader, char \*\*description)

*Get description.*

- [OTF2\\_EvtReader](#) \* [OTF2\\_Reader\\_GetEvtReader](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_LocationRef](#) location)

*Get a local event reader.*

- [OTF2\\_StatusCode](#) [OTF2\\_Reader\\_GetFileSubstrate](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_FileSubstrate](#) \*substrate)

*Get file substrate information.*

- [OTF2\\_GlobalDefReader](#) \* [OTF2\\_Reader\\_GetGlobalDefReader](#) ([OTF2\\_Reader](#) \*reader)

*Get a global definition reader.*

- [OTF2\\_GlobalEvtReader](#) \* [OTF2\\_Reader\\_GetGlobalEvtReader](#) ([OTF2\\_Reader](#) \*reader)

*Get a global event reader.*

- [OTF2\\_GlobalSnapReader](#) \* [OTF2\\_Reader\\_GetGlobalSnapReader](#) ([OTF2\\_Reader](#) \*reader)

*Get a global snap reader.*

- [OTF2\\_StatusCode](#) [OTF2\\_Reader\\_GetMachineName](#) ([OTF2\\_Reader](#) \*reader, char \*\*machineName)

*Get machine name.*

- [OTF2\\_MarkerReader](#) \* [OTF2\\_Reader\\_GetMarkerReader](#) ([OTF2\\_Reader](#) \*reader)

*Get a marker reader.*

- [OTF2\\_MarkerWriter](#) \* [OTF2\\_Reader\\_GetMarkerWriter](#) ([OTF2\\_Reader](#) \*reader)

*Get a marker writer.*

- [OTF2\\_StatusCode](#) [OTF2\\_Reader\\_GetNumberOfGlobalDefinitions](#) ([OTF2\\_Reader](#) \*reader, [uint64\\_t](#) \*numberOfDefinitions)

*Get number of global definitions.*

## E.29 otf2/OTF2\_Reader.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetNumberOfLocations](#) (OTF2\_Reader \*reader, uint64\_t \*numberOfLocations)  
*Get number of locations.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetNumberOfSnapshots](#) (OTF2\_Reader \*reader, uint32\_t \*number)  
*Get number of snapshots.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetNumberOfThumbnails](#) (OTF2\_Reader \*reader, uint32\_t \*number)  
*Get number of thumbs.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetProperty](#) (OTF2\_Reader \*reader, const char \*name, char \*\*value)  
*Get the value of the named trace file property.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetPropertyNames](#) (OTF2\_Reader \*reader, uint32\_t \*numberOfProperties, char \*\*\*names)  
*Get the names of all trace file properties.*
- [OTF2\\_SnapReader \\* OTF2\\_Reader\\_GetSnapReader](#) (OTF2\_Reader \*reader, OTF2\_LocationRef location)  
*Get a local snapshot reader.*
- [OTF2\\_ThumbReader \\* OTF2\\_Reader\\_GetThumbReader](#) (OTF2\_Reader \*reader, uint32\_t number)  
*Get a thumb reader.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetTraceId](#) (OTF2\_Reader \*reader, uint64\_t \*id)  
*Get the identifier of the trace file.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_GetVersion](#) (OTF2\_Reader \*reader, uint8\_t \*major, uint8\_t \*minor, uint8\_t \*bugfix)  
*Get OTF2 version.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_HasGlobalEvent](#) (OTF2\_Reader \*reader, OTF2\_GlobalEvtReader \*evtReader, int \*flag)  
*Has the global event reader at least one more event to deliver.*
- [OTF2\\_Reader \\* OTF2\\_Reader\\_Open](#) (const char \*anchorFilePath)  
*Create a new reader handle.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_OpenDefFiles](#) (OTF2\_Reader \*reader)  
*Open the local definitions file container.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_OpenEvtFiles](#) (OTF2\_Reader \*reader)  
*Open the events file container.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_OpenSnapFiles](#) (OTF2\_Reader \*reader)  
*Open the snapshots file container.*
- [OTF2\\_ErrorCode OTF2\\_Reader\\_ReadAllGlobalDefinitions](#) (OTF2\_Reader \*reader, OTF2\_GlobalDefReader \*defReader, uint64\_t \*definitionsRead)

---

## APPENDIX E. FILE DOCUMENTATION

---

*Read all definitions via a global definition reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadAllGlobalEvents](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalEvtReader](#) \*evtReader, uint64\_t \*eventsRead)

*Read all events via a global event reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadAllGlobalSnapshots](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalSnapReader](#) \*snapReader, uint64\_t \*recordsRead)

*Read all records via a global snapshot reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadAllLocalDefinitions](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_DefReader](#) \*defReader, uint64\_t \*definitionsRead)

*Read all definitions via a local definition reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadAllLocalEvents](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_EvtReader](#) \*evtReader, uint64\_t \*eventsRead)

*Read all events via a local event reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadAllLocalSnapshots](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_SnapReader](#) \*snapReader, uint64\_t \*recordsRead)

*Read all records via a local snapshot reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadAllMarkers](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_MarkerReader](#) \*markerReader, uint64\_t \*markersRead)

*Read all markers via a marker reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadGlobalDefinitions](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalDefReader](#) \*defReader, uint64\_t definitionsToRead, uint64\_t \*definitionsRead)

*Read a given number of definitions via a global definition reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadGlobalEvent](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalEvtReader](#) \*evtReader)

*Read an event via a global event reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadGlobalEvents](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalEvtReader](#) \*evtReader, uint64\_t eventsToRead, uint64\_t \*eventsRead)

*Read a given number of events via a global event reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadGlobalSnapshots](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalSnapReader](#) \*snapReader, uint64\_t recordsToRead, uint64\_t \*recordsRead)

*Read a given number of records via a global snapshot reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadLocalDefinitions](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_DefReader](#) \*defReader, uint64\_t definitionsToRead, uint64\_t \*definitionsRead)

*Read a given number of definitions via a local definition reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadLocalEvents](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_EvtReader](#) \*evtReader, uint64\_t eventsToRead, uint64\_t \*eventsRead)

## E.29 otf2/OTF2\_Reader.h File Reference

---

*Read a given number of events via a local event reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadLocalEventsBackward](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_EvtReader](#) \*evtReader, uint64\_t eventsToRead, uint64\_t \*eventsRead)

*Read a given number of events via a local event reader backwards.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadLocalSnapshots](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_SnapReader](#) \*snapReader, uint64\_t recordsToRead, uint64\_t \*recordsRead)

*Read a given number of records via a local snapshot reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_ReadMarkers](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_MarkerReader](#) \*markerReader, uint64\_t markersToRead, uint64\_t \*markersRead)

*Read a given number of markers via a marker reader.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_RegisterDefCallbacks](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_DefReader](#) \*defReader, const [OTF2\\_DefReaderCallbacks](#) \*callbacks, void \*userData)

*Register local definition reader callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_RegisterEvtCallbacks](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_EvtReader](#) \*evtReader, const [OTF2\\_EvtReaderCallbacks](#) \*callbacks, void \*userData)

*Register event reader callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_RegisterGlobalDefCallbacks](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalDefReader](#) \*defReader, const [OTF2\\_GlobalDefReaderCallbacks](#) \*callbacks, void \*userData)

*Register global definition reader callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_RegisterGlobalEvtCallbacks](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalEvtReader](#) \*evtReader, const [OTF2\\_GlobalEvtReaderCallbacks](#) \*callbacks, void \*userData)

*Register global event reader callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_RegisterGlobalSnapCallbacks](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_GlobalSnapReader](#) \*evtReader, const [OTF2\\_GlobalSnapReaderCallbacks](#) \*callbacks, void \*userData)

*Register global event reader callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_RegisterMarkerCallbacks](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_MarkerReader](#) \*markerReader, const [OTF2\\_MarkerReaderCallbacks](#) \*callbacks, void \*userData)

*Register marker reader callbacks.*

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_RegisterSnapCallbacks](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_SnapReader](#) \*snapReader, const [OTF2\\_SnapReaderCallbacks](#) \*callbacks, void \*userData)

*Register snapshot event reader callbacks.*

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_SelectLocation](#) ([OTF2\\_Reader](#) \*reader, [OTF2\\_LocationRef](#) location)  
*Select a location to be read.*
- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_SetCollectiveCallbacks](#) ([OTF2\\_Reader](#) \*reader, const [OTF2\\_CollectiveCallbacks](#) \*collectiveCallbacks, void \*collectiveData, [OTF2\\_CollectiveContext](#) \*globalCommContext, [OTF2\\_CollectiveContext](#) \*localCommContext)  
*Set the collective callbacks for the reader.*
- [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_SetSerialCollectiveCallbacks](#) ([OTF2\\_Reader](#) \*reader)  
*Convenient function to set the collective callbacks to an serial implementation.*

### E.29.1 Detailed Description

Reading interface for OTF2 archives.

### E.29.2 Function Documentation

#### E.29.2.1 [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_Close](#) ( [OTF2\\_Reader](#) \* reader )

Close a reader handle.

Closes a reader handle and releases all associated handles. Does nothing if NULL is provided.

#### Parameters

<i>reader</i>	Reader handle.
---------------	----------------

#### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

#### E.29.2.2 [OTF2\\_ErrorCode](#) [OTF2\\_Reader\\_CloseDefFiles](#) ( [OTF2\\_Reader](#) \* reader )

Closes the local definitions file container.

This function is an collective operation.

All previously used local definition readers must be closed before this call.

#### Parameters

<i>reader</i>	Reader handle.
---------------	----------------



## E.29 otf2/OTF2\_Reader.h File Reference

---

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.29.2.3 **OTF2\_ErrorCode** **OTF2\_Reader\_CloseDefReader** ( **OTF2\_Reader** \* *reader*, **OTF2\_DefReader** \* *defReader* )

Close a local definition reader.

### Parameters

<i>reader</i>	Valid reader handle.
<i>defReader</i>	Definition reader to be closed.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.29.2.4 **OTF2\_ErrorCode** **OTF2\_Reader\_CloseEvtFiles** ( **OTF2\_Reader** \* *reader* )

Closes the events file container.

All previously used event readers must be closed before this call.

This function is an collective operation.

### Parameters

<i>reader</i>	Reader handle.
---------------	----------------

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.5   OTF2\_ErrorCode OTF2\_Reader\_CloseEvtReader ( OTF2\_Reader \* *reader*, OTF2\_EvtReader \* *evtReader* )**

Close a local event reader.

**Parameters**

<i>reader</i>	Valid reader handle.
<i>evtReader</i>	Event reader to be closed.

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.6   OTF2\_ErrorCode OTF2\_Reader\_CloseGlobalDefReader ( OTF2\_Reader \* *reader*, OTF2\_GlobalDefReader \* *globalDefReader* )**

Closes the global definition reader.

**Parameters**

<i>reader</i>	Valid reader handle.
<i>globalDef-Reader</i>	The global definition reader.

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.7   OTF2\_ErrorCode OTF2\_Reader\_CloseGlobalEvtReader ( OTF2\_Reader \* *reader*, OTF2\_GlobalEvtReader \* *globalEvtReader* )**

Closes the global event reader.

This closes also all local event readers.

**Parameters**

<i>reader</i>	Valid reader handle.
<i>globalEvtReader</i>	The global event reader.

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.8** **OTF2\_ErrorCode** **OTF2\_Reader\_CloseGlobalSnapReader** ( **OTF2\_Reader \***  
***reader***, **OTF2\_GlobalSnapReader \*** ***globalSnapReader*** )

Closes the global snapshot reader.

### Parameters

<i>reader</i>	Valid reader handle.
<i>global-SnapReader</i>	The global snapshot reader.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### Since

Version 1.2

**E.29.2.9** **OTF2\_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.

**E.29.2.10   OTF2\_StatusCode OTF2\_Reader\_CloseMarkerWriter ( OTF2\_Reader \*  
reader, OTF2\_MarkerWriter \* markerWriter )**

Closes the marker writer.

**Parameters**

<i>reader</i>	Valid reader handle.
<i>marker-Writer</i>	The marker writer.

**Since**

Version 1.2

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.11   OTF2\_StatusCode OTF2\_Reader\_CloseSnapFiles ( OTF2\_Reader \* reader  
)**

Closes the snapshots file container.

This function is an collective operation.

All previously used snapshot readers must be closed before this call.

**Parameters**

<i>reader</i>	Reader handle.
---------------	----------------

**Since**

Version 1.3

**Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.12   OTF2\_StatusCode OTF2\_Reader\_CloseSnapReader ( OTF2\_Reader \*  
reader, OTF2\_SnapReader \* snapReader )**

Close a local snapshot reader.

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Parameters

<i>reader</i>	Valid reader handle.
<i>snapReader</i>	snapshot reader to be closed.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### Since

Version 1.2

#### E.29.2.13 **OTF2\_ErrorCode** OTF2\_Reader\_CloseThumbReader ( OTF2\_Reader \* *reader*, OTF2\_ThumbReader \* *thumbReader* )

Close an opened thumbnail reader.

### Parameters

<i>reader</i>	Reader handle.
<i>thumbReader</i>	Thumbn reader handle to be closed.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

#### E.29.2.14 **OTF2\_ErrorCode** OTF2\_Reader\_GetBoolProperty ( OTF2\_Reader \* *reader*, const char \* *name*, bool \* *value* )

Get the value of the named trace file property as boolean.

### Parameters

	<i>reader</i>	Reader handle.
	<i>name</i>	Name of the property.
out	<i>value</i>	Returned boolean value of the property.

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_PROPERTY\_NOT\_FOUND* if the named property was not found

*OTF2\_ERROR\_PROPERTY\_VALUE\_INVALID* if the value could not be interpreted as an boolean value

**E.29.2.15** `OTF2_StatusCode OTF2_Reader_GetChunkSize ( OTF2_Reader * reader, uint64_t * chunkSizeEvents, uint64_t * chunkSizeDefinitions )`

Get event and definition chunk sizes.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>chunk-SizeEvents</i>	Returned size of event chunks
out	<i>chunk-SizeDefinitions</i>	Returned size of definition chunks.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.29.2.16** `OTF2_StatusCode OTF2_Reader_GetCompression ( OTF2_Reader * reader, OTF2_Compression * compression )`

Get copression mode.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>compression</i>	Returned compression mode.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## E.29 otf2/OTF2\_Reader.h File Reference

---

**E.29.2.17** `OTF2_ErrorCode OTF2_Reader_GetCreator ( OTF2_Reader * reader,  
char ** creator )`

Get creator name.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>creator</i>	Returned creator. Allocated with <i>malloc</i> .

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.18** `OTF2_DefReader* OTF2_Reader_GetDefReader ( OTF2_Reader * reader,  
OTF2_LocationRef location )`

Get a local definition reader.

### Parameters

	<i>reader</i>	Valid reader handle.
	<i>location</i>	Location ID for the requested local reader.

### Returns

Returns a handle to the local definition reader if successful, NULL otherwise.

**E.29.2.19** `OTF2_ErrorCode OTF2_Reader_GetDescription ( OTF2_Reader * reader,  
char ** description )`

Get description.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>description</i>	Returned description. Allocated with <i>malloc</i> .

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

### E.29.2.20 **OTF2\_EvtReader\*** **OTF2\_Reader\_GetEvtReader** ( **OTF2\_Reader \*** *reader*, **OTF2\_LocationRef** *location* )

Get a local event reader.

#### Parameters

<i>reader</i>	Valid reader handle.
<i>location</i>	Location ID for the requested local reader.

#### Returns

Returns a handle to the local event reader if successful, NULL otherwise.

### E.29.2.21 **OTF2\_ErrorCode** **OTF2\_Reader\_GetFileSubstrate** ( **OTF2\_Reader \*** *reader*, **OTF2\_FileSubstrate \*** *substrate* )

Get file substrate information.

#### Parameters

	<i>reader</i>	Reader handle.
out	<i>substrate</i>	Returned file substrate.

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### E.29.2.22 **OTF2\_GlobalDefReader\*** **OTF2\_Reader\_GetGlobalDefReader** ( **OTF2\_Reader \*** *reader* )

Get a global definition reader.

#### Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

#### Returns

Returns a handle to the global definition reader if successful, NULL otherwise.



## E.29 otf2/OTF2\_Reader.h File Reference

---

### E.29.2.23 OTF2\_GlobalEvtReader\* OTF2\_Reader\_GetGlobalEvtReader ( OTF2\_Reader \* *reader* )

Get a global event reader.

#### Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

#### Returns

Returns a handle to the global event reader if successful, NULL otherwise.

### E.29.2.24 OTF2\_GlobalSnapReader\* OTF2\_Reader\_GetGlobalSnapReader ( OTF2\_Reader \* *reader* )

Get a global snap reader.

#### Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

#### Returns

Returns a handle to the global snap reader if successful, NULL otherwise.

#### Since

Version 1.2

### E.29.2.25 OTF2\_ErrorCode OTF2\_Reader\_GetMachineName ( OTF2\_Reader \* *reader*, char \*\* *machineName* )

Get machine name.

#### Parameters

	<i>reader</i>	Reader handle.
out	<i>machine-Name</i>	Returned machine name. Allocated with <i>malloc</i> .

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

### E.29.2.26 **OTF2\_MarkerReader\*** **OTF2\_Reader\_GetMarkerReader** ( **OTF2\_Reader** \* *reader* )

Get a marker reader.

#### Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

#### Since

Version 1.2

#### Returns

Returns a handle to the marker reader if successful, NULL otherwise.

### E.29.2.27 **OTF2\_MarkerWriter\*** **OTF2\_Reader\_GetMarkerWriter** ( **OTF2\_Reader** \* *reader* )

Get a marker writer.

#### Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

#### Since

Version 1.2

#### Returns

Returns a handle to the marker writer if successful, NULL otherwise.

### E.29.2.28 **OTF2\_ErrorCode** **OTF2\_Reader\_GetNumberOfGlobalDefinitions** ( **OTF2\_Reader** \* *reader*, **uint64\_t** \* *numberOfDefinitions* )

Get number of global definitions.

#### Parameters

	<i>reader</i>	Reader handle.
out	<i>numberOfDefinitions</i>	Returned number of global definitions.

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.29** **OTF2\_ErrorCode** OTF2\_Reader\_GetNumberOfLocations ( OTF2\_Reader  
\* *reader*, uint64\_t \* *numberOfLocations* )

Get number of locations.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>num- berOfLoca- tions</i>	Returned number of locations.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.30** **OTF2\_ErrorCode** OTF2\_Reader\_GetNumberOfSnapshots ( OTF2\_Reader  
\* *reader*, uint32\_t \* *number* )

Get number of snapshots.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>number</i>	Returned number of snapshots.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### Since

Version 1.2

**E.29.2.31** **OTF2\_ErrorCode** OTF2\_Reader\_GetNumberOfThumbnails ( OTF2\_Reader  
\* *reader*, uint32\_t \* *number* )

Get number of thumbs.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

	<i>reader</i>	Reader handle.
out	<i>number</i>	Returned number of thumbs.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### Since

Version 1.2

**E.29.2.32** **OTF2\_StatusCode** **OTF2\_Reader\_GetProperty** ( **OTF2\_Reader** \* *reader*,  
const char \* *name*, char \*\* *value* )

Get the value of the named trace file property.

### Parameters

	<i>reader</i>	Reader handle.
	<i>name</i>	Name of the property.
out	<i>value</i>	Returned value of the property. Allocated with <i>malloc</i> .

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_PROPERTY\\_NOT\\_FOUND\*](#) if the named property was not found

**E.29.2.33** **OTF2\_StatusCode** **OTF2\_Reader\_GetPropertyNames** ( **OTF2\_Reader** \*  
*reader*, uint32\_t \* *numberOfProperties*, char \*\*\* *names* )

Get the names of all trace file properties.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>numberOfProperties</i>	Returned number of trace file properties.
out	<i>names</i>	Returned list of property names. Allocated with <i>malloc</i> . To release memory, just pass * <i>names</i> to <i>free</i> .

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.34** `OTF2_SnapReader* OTF2_Reader_GetSnapReader ( OTF2_Reader *  
reader, OTF2_LocationRef location )`

Get a local snapshot reader.

### Parameters

<i>reader</i>	Valid reader handle.
<i>location</i>	Location ID for the requested local reader.

### Returns

Returns a handle to the local event reader if successful, NULL otherwise.

### Since

Version 1.2

**E.29.2.35** `OTF2_ThumbReader* OTF2_Reader_GetThumbReader ( OTF2_Reader  
* reader, uint32_t number )`

Get a thumb reader.

### Parameters

<i>reader</i>	Reader handle.
<i>number</i>	Thumbnail number.

### Since

Version 1.2

### Returns

Returns a global definition writer handle if successful, NULL if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.29.2.36** **OTF2\_ErrorCode** **OTF2\_Reader\_GetTraceId** ( **OTF2\_Reader** \* *reader*,  
uint64\_t \* *id* )

Get the identifier of the trace file.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>id</i>	Trace identifier.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.37** **OTF2\_ErrorCode** **OTF2\_Reader\_GetVersion** ( **OTF2\_Reader** \* *reader*,  
uint8\_t \* *major*, uint8\_t \* *minor*, uint8\_t \* *bugfix* )

Get OTF2 version.

### Parameters

	<i>reader</i>	Valid reader handle.
out	<i>major</i>	Major version.
out	<i>minor</i>	Minor version.
out	<i>bugfix</i>	Bugfix revision.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.38** **OTF2\_ErrorCode** **OTF2\_Reader\_HasGlobalEvent** ( **OTF2\_Reader** \*  
*reader*, **OTF2\_GlobalEvtReader** \* *evtReader*, int \* *flag* )

Has the global event reader at least one more event to deliver.

### Parameters

	<i>reader</i>	Global event reader handle.
	<i>evtReader</i>	Global event reader handle.
out	<i>flag</i>	In case of success, the flag will be set to 1 when there is at least more more event to read. To 0 if not. Otherwise the value is undefined.

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### E.29.2.39 OTF2\_Reader\* OTF2\_Reader\_Open ( const char \* *anchorFilePath* )

Create a new reader handle.

Creates a new reader handle, opens an according archive handle, and calls a routine to register all necessary function pointers.

### Parameters

<i>anchor-FilePath</i>	Path to the anchor file e.g. 'trace.otf2'. This can be a relative as well as an absolute path.
------------------------	--

### Returns

Returns a handle to the reader if successful, NULL otherwise.

### E.29.2.40 OTF2\_ErrorCode OTF2\_Reader\_OpenDefFiles ( OTF2\_Reader \* *reader* )

Open the local definitions file container.

This function is an collective operation.

### Parameters

<i>reader</i>	Reader handle.
---------------	----------------

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### E.29.2.41 OTF2\_ErrorCode OTF2\_Reader\_OpenEvtFiles ( OTF2\_Reader \* *reader* )

Open the events file container.

This function is an collective operation.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>reader</i>	Reader handle.
---------------	----------------

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.42** **OTF2\_ErrorCode** **OTF2\_Reader\_OpenSnapFiles** ( **OTF2\_Reader** \* *reader*  
)

Open the snapshots file container.

This function is an collective operation.

### Parameters

<i>reader</i>	Reader handle.
---------------	----------------

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.43** **OTF2\_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.



## E.29 otf2/OTF2\_Reader.h File Reference

---

**E.29.2.44** `OTF2_Reader_ReadAllGlobalEvents ( OTF2_Reader *  
reader, OTF2_GlobalEvtReader * evtReader, uint64_t * eventsRead )`

Read all events via a global event reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Global event reader handle.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.45** `OTF2_Reader_ReadAllGlobalSnapshots ( OTF2_Reader  
* reader, OTF2_GlobalSnapReader * snapReader, uint64_t * recordsRead  
)`

Read all records via a global snapshot reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Global snapshot reader handle.
out	<i>recordsRead</i>	Return pointer to the number of records

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### Since

Version 1.2

**E.29.2.46** `OTF2_Reader_ReadAllLocalDefinitions ( OTF2_Reader  
* reader, OTF2_DefReader * defReader, uint64_t * definitionsRead )`

Read all definitions via a local definition reader.

### Parameters

	<i>reader</i>	Reader handle.
--	---------------	----------------

## APPENDIX E. FILE DOCUMENTATION

---

	<i>defReader</i>	Local definition reader handle.
out	<i>definition- sRead</i>	Return pointer to the number of definitions actually read.

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK* if an user supplied call-back returned *OTF2\_CALLBACK\_INTERRUPT*

*OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE* if an duplicate mapping table definition was read

*otherwise* the error code

**E.29.2.47** **OTF2\_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.

**E.29.2.48** **OTF2\_ErrorCode** **OTF2\_Reader\_ReadAllLocalSnapshots** ( **OTF2\_Reader** \* **reader**, **OTF2\_SnapReader** \* **snapReader**, **uint64\_t** \* **recordsRead** )

Read all records via a local snapshot reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Local snapshot reader handle.
out	<i>record- sRead</i>	Return pointer to the number of records

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### Since

Version 1.2

**E.29.2.49** **OTF2\_ErrorCode** **OTF2\_Reader\_ReadAllMarkers** ( **OTF2\_Reader** \* *reader*,  
**OTF2\_MarkerReader** \* *markerReader*, **uint64\_t** \* *markersRead* )

Read all markers via a marker reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>marker-Reader</i>	Marker reader handle.
out	<i>marker-sRead</i>	Return pointer to the number of markers actually read.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.50** **OTF2\_ErrorCode** **OTF2\_Reader\_ReadGlobalDefinitions** ( **OTF2\_Reader** \*  
*reader*, **OTF2\_GlobalDefReader** \* *defReader*, **uint64\_t** *definitionsToRead*,  
**uint64\_t** \* *definitionsRead* )

Read a given number of definitions via a global definition reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>defReader</i>	Global definition reader handle.
	<i>definition-sToRead</i>	Number definitions to be read.
out	<i>definition-sRead</i>	Return pointer to the number of definitions actually read.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.29.2.51** `OTF2_StatusCode OTF2_Reader_ReadGlobalEvent ( OTF2_Reader *  
reader, OTF2_GlobalEvtReader * evtReader )`

Read an event via a global event reader.

### Parameters

<i>reader</i>	Reader handle.
<i>evtReader</i>	Global event reader handle.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.29.2.52** `OTF2_StatusCode OTF2_Reader_ReadGlobalEvents ( OTF2_Reader *  
reader, OTF2_GlobalEvtReader * evtReader, uint64_t eventsToRead,  
uint64_t * eventsRead )`

Read a given number of events via a global event reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Global event reader handle.
	<i>eventsToRead</i>	Number events to be read.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.29.2.53** `OTF2_StatusCode OTF2_Reader_ReadGlobalSnapshots ( OTF2_Reader *  
reader, OTF2_GlobalSnapReader * snapReader, uint64_t recordsToRead,  
uint64_t * recordsRead )`

Read a given number of records via a global snapshot reader.

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Parameters

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Global snapshot reader handle.
	<i>recordsToRead</i>	Number records to be read.
out	<i>recordsRead</i>	Return pointer to the number of records actually read.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### Since

Version 1.2

**E.29.2.54** **OTF2\_ErrorCode** OTF2\_Reader\_ReadLocalDefinitions ( OTF2\_Reader \*  
*reader*, OTF2\_DefReader \* *defReader*, uint64\_t *definitionsToRead*, uint64\_t  
\* *definitionsRead* )

Read a given number of definitions via a local definition reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>defReader</i>	Local definition reader handle.
	<i>definitionsToRead</i>	Number definitions to be read.
out	<i>definitionsRead</i>	Return pointer to the number of definitions actually read.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful

[\*OTF2\\_ERROR\\_INTERRUPTED\\_BY\\_CALLBACK\*](#) if an user supplied call-back returned OTF2\_CALLBACK\_INTERRUPT

[\*OTF2\\_ERROR\\_DUPLICATE\\_MAPPING\\_TABLE\*](#) if an duplicate mapping table definition was read

*otherwise* the error code

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.29.2.55** **OTF2\_StatusCode** **OTF2\_Reader\_ReadLocalEvents** ( **OTF2\_Reader** \* *reader*, **OTF2\_EvtReader** \* *evtReader*, **uint64\_t** *eventsToRead*, **uint64\_t** \* *eventsRead* )

Read a given number of events via a local event reader.

### Parameters

<i>reader</i>	Reader handle.
<i>evtReader</i>	Local event reader handle.
<i>eventsToRead</i>	Number events to be read.
<i>eventsRead</i>	Return pointer to the number of events actually read.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.56** **OTF2\_StatusCode** **OTF2\_Reader\_ReadLocalEventsBackward** ( **OTF2\_Reader** \* *reader*, **OTF2\_EvtReader** \* *evtReader*, **uint64\_t** *eventsToRead*, **uint64\_t** \* *eventsRead* )

Read a given number of events via a local event reader backwards.

### Parameters

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Local event reader handle.
	<i>eventsToRead</i>	Number events to be read.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.57** **OTF2\_StatusCode** **OTF2\_Reader\_ReadLocalSnapshots** ( **OTF2\_Reader** \* *reader*, **OTF2\_SnapReader** \* *snapReader*, **uint64\_t** *recordsToRead*, **uint64\_t** \* *recordsRead* )

Read a given number of records via a local snapshot reader.

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Parameters

<i>reader</i>	Reader handle.
<i>snapReader</i>	Local snapshot reader handle.
<i>recordsToRead</i>	Number records to be read.
<i>recordsRead</i>	Return pointer to the number of records actually read.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

### Since

Version 1.2

**E.29.2.58** **OTF2\_ErrorCode** OTF2\_Reader\_ReadMarkers ( OTF2\_Reader \* *reader*, OTF2\_MarkerReader \* *markerReader*, uint64\_t *markersToRead*, uint64\_t \* *markersRead* )

Read a given number of markers via a marker reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>markerReader</i>	Marker reader handle.
	<i>markersToRead</i>	Number markers to be read.
out	<i>markersRead</i>	Return pointer to the number of markers actually read.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.29.2.59** **OTF2\_ErrorCode** **OTF2\_Reader\_RegisterDefCallbacks** (  
    **OTF2\_Reader** \* *reader*, **OTF2\_DefReader** \* *defReader*, **const**  
    **OTF2\_DefReaderCallbacks** \* *callbacks*, **void** \* *userData* )

Register local definition reader callbacks.

### Parameters

<i>reader</i>	OTF2_Reader handle.
<i>defReader</i>	Local definition reader handle.
<i>callbacks</i>	Callbacks for the local definition readers.
<i>userData</i>	Addition user data.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.60** **OTF2\_ErrorCode** **OTF2\_Reader\_RegisterEvtCallbacks** (  
    **OTF2\_Reader** \* *reader*, **OTF2\_EvtReader** \* *evtReader*, **const**  
    **OTF2\_EvtReaderCallbacks** \* *callbacks*, **void** \* *userData* )

Register event reader callbacks.

### Parameters

<i>reader</i>	OTF2_Reader handle.
<i>evtReader</i>	Local event reader handle.
<i>callbacks</i>	Callbacks for the event readers.
<i>userData</i>	Addition user data.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.61** **OTF2\_ErrorCode** **OTF2\_Reader\_RegisterGlobalDefCallbacks** (  
    **OTF2\_Reader** \* *reader*, **OTF2\_GlobalDefReader** \* *defReader*, **const**  
    **OTF2\_GlobalDefReaderCallbacks** \* *callbacks*, **void** \* *userData* )

Register global definition reader callbacks.

### Parameters

<i>reader</i>	OTF2_Reader handle.
---------------	---------------------



## E.29 otf2/OTF2\_Reader.h File Reference

---

<i>defReader</i>	Global definition reader handle.
<i>callbacks</i>	Callbacks for the global definition readers.
<i>userData</i>	Addition user data.

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.62** **OTF2\_ErrorCode** **OTF2\_Reader\_RegisterGlobalEvtCallbacks** (  
OTF2\_Reader \* *reader*, OTF2\_GlobalEvtReader \* *evtReader*, const  
OTF2\_GlobalEvtReaderCallbacks \* *callbacks*, void \* *userData* )

Register global event reader callbacks.

### Parameters

<i>reader</i>	OTF2_Reader handle.
<i>evtReader</i>	Global event reader handle.
<i>callbacks</i>	Callbacks for the global event reader.
<i>userData</i>	Addition user data.

### Returns

Returns [\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.63** **OTF2\_ErrorCode** **OTF2\_Reader\_RegisterGlobalSnapCallbacks** (  
OTF2\_Reader \* *reader*, OTF2\_GlobalSnapReader \* *evtReader*, const  
OTF2\_GlobalSnapReaderCallbacks \* *callbacks*, void \* *userData* )

Register global event reader callbacks.

### Parameters

<i>reader</i>	OTF2_Reader handle.
<i>evtReader</i>	Global event reader handle.
<i>callbacks</i>	Callbacks for the global event reader.
<i>userData</i>	Addition user data.

### Since

Version 1.2

**Returns**

Returns *OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.29.2.64** **OTF2\_StatusCode** **OTF2\_Reader\_RegisterMarkerCallbacks** (  
    **OTF2\_Reader** \* *reader*, **OTF2\_MarkerReader** \* *markerReader*, **const**  
    **OTF2\_MarkerReaderCallbacks** \* *callbacks*, **void** \* *userData* )

Register marker reader callbacks.

**Parameters**

<i>reader</i>	OTF2_Reader handle.
<i>marker-Reader</i>	Marker reader handle.
<i>callbacks</i>	Callbacks for the marker reader.
<i>userData</i>	Addition user data.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.29.2.65** **OTF2\_StatusCode** **OTF2\_Reader\_RegisterSnapCallbacks** (  
    **OTF2\_Reader** \* *reader*, **OTF2\_SnapReader** \* *snapReader*, **const**  
    **OTF2\_SnapReaderCallbacks** \* *callbacks*, **void** \* *userData* )

Register snapshot event reader callbacks.

**Parameters**

<i>reader</i>	OTF2_Reader handle.
<i>snapReader</i>	Local snap reader handle.
<i>callbacks</i>	Callbacks for the event readers.
<i>userData</i>	Addition user data.

**Since**

Version 1.2

## E.29 otf2/OTF2\_Reader.h File Reference

---

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.66** **OTF2\_ErrorCode** **OTF2\_Reader\_SelectLocation** ( **OTF2\_Reader** \* *reader*,  
**OTF2\_LocationRef** *location* )

Select a location to be read.

### Parameters

<i>reader</i>	Reader handle.
<i>location</i>	Location ID.

### Since

Version 1.3

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.29.2.67** **OTF2\_ErrorCode** **OTF2\_Reader\_SetCollectiveCallbacks** ( **OTF2\_Reader**  
\* *reader*, **const** **OTF2\_CollectiveCallbacks** \* *collectiveCallbacks*, **void**  
\* *collectiveData*, **OTF2\_CollectiveContext** \* *globalCommContext*,  
**OTF2\_CollectiveContext** \* *localCommContext* )

Set the collective callbacks for the reader.

The reader has as the default the serial collectives set.

This function is an collective operation.

### Parameters

<i>reader</i>	Reader handle.
<i>collective-Callbacks</i>	Struct holding the collective callback functions.
<i>collective-Data</i>	Data passed to the collective callbacks in the <code>userData</code> argument.
<i>global-CommContext</i>	Global communication context.
<i>local-CommContext</i>	Local communication context. Unsued in reading mode. A local communication context may be created via the callbacks which fits the one used when the given trace was written.

**Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.29.2.68 OTF2\_ErrorCode OTF2\_Reader\_SetSerialCollectiveCallbacks ( OTF2\_Reader \* reader )**

Convenient function to set the collective callbacks to an serial implementation.

**Parameters**

<i>reader</i>	Reader handle.
---------------	----------------

**Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.30 otf2/OTF2\_SnapReader.h File Reference**

This is the local snap reader, which reads snapshot events from one location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_SnapReaderCallbacks.h>
```

**Functions**

- [OTF2\\_ErrorCode OTF2\\_SnapReader\\_GetLocationID](#) (const [OTF2\\_SnapReader](#) \*reader, [OTF2\\_LocationRef](#) \*location)  
*Return the location ID of the reading related location.*
- [OTF2\\_ErrorCode OTF2\\_SnapReader\\_ReadSnapshots](#) ([OTF2\\_SnapReader](#) \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)  
*After callback registration, the local events could be read with the following function. Readn reads recordsToRead records. The reader indicates that it reached the end of the trace by just reading less records than requested.*
- [OTF2\\_ErrorCode OTF2\\_SnapReader\\_Seek](#) ([OTF2\\_SnapReader](#) \*reader, uint64\_t req\_time, bool \*found)

## E.30 otf2/OTF2\_SnapReader.h File Reference

---

*Seek jumps to start of latest snapshot that was made before a given time 'req\_time'.*

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapReader\\_SetCallbacks](#) ([OTF2\\_SnapReader](#) \*reader, const [OTF2\\_SnapReaderCallbacks](#) \*callbacks, void \*userData)

*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### E.30.1 Detailed Description

This is the local snap reader, which reads snapshot events from one location.

### E.30.2 Function Documentation

#### E.30.2.1 [OTF2\\_ErrorCode](#) [OTF2\\_SnapReader\\_GetLocationID](#) ( const [OTF2\\_SnapReader](#) \* reader, [OTF2\\_LocationRef](#) \* location )

Return the location ID of the reading related location.

#### Parameters

	<i>reader</i>	Reader object which reads the snapshot events from its buffer.
out	<i>location</i>	ID of the location.

#### Since

Version 1.2

#### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

#### E.30.2.2 [OTF2\\_ErrorCode](#) [OTF2\\_SnapReader\\_ReadSnapshots](#) ( [OTF2\\_SnapReader](#) \* reader, [uint64\\_t](#) recordsToRead, [uint64\\_t](#) \* recordsRead )

After callback registration, the local events could be read with the following function. Readn reads *recordsToRead* records. The reader indicates that it reached the end of the trace by just reading less records than requested.

#### Parameters

---

---

## APPENDIX E. FILE DOCUMENTATION

---

	<i>reader</i>	Reader object which reads the events from its buffer.
	<i>recordsToRead</i>	How many records can be read next.
out	<i>recordsRead</i>	Return how many records were really read.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.30.2.3** **OTF2\_ErrorCode OTF2\_SnapReader\_Seek ( OTF2\_SnapReader \* *reader*,  
uint64\_t *req\_time*, bool \* *found* )**

Seek jumps to start of latest snapshot that was made before a given time '*req\_time*'.

### Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
<i>req_time</i>	Requested time (see above)
<i>found</i>	returns if a matching snapshot was found

### Since

Version 1.2

### Returns

OTF2\_ErrorCode with !=OTF2\_SUCCESS if there was an error.

**E.30.2.4** **OTF2\_ErrorCode OTF2\_SnapReader\_SetCallbacks ( OTF2\_SnapReader \*  
*reader*, const OTF2\_SnapReaderCallbacks \* *callbacks*, void \* *userData* )**

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

These callbacks are ignored, if the events are read by an global event reader.

## E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

### Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#">OTF2_SnapReaderCallbacks_New</a> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

This defines the callbacks for the snap reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

### Typedefs

- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_Enter](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_RegionRef](#) region)  
*Callback for the Enter snap event record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_MeasurementOnOff](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_MeasurementMode](#) measurementMode)  
*Callback for the MeasurementOnOff snap event record.*
- typedef [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_Metric](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_MetricRef](#) metric, uint8\_t numberOfMetrics, const [OTF2\\_Type](#) \*typeIDs, const [OTF2\\_MetricValue](#) \*metricValues)

*Callback for the Metric snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime)`

*Callback for the MpiCollectiveBegin snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)`

*Callback for the MpiCollectiveEnd snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiIrecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`

*Callback for the MpiIrecv snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)`

*Callback for the MpiIrecvRequest snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiIsend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`

*Callback for the MpiIsend snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)`

*Callback for the MpiIsendComplete snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiRecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

*Callback for the MpiRecv snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiSend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t re-`



### E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

ceiver, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength)

*Callback for the `MpiSend` snap event record.*

- [typedef](#) [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_OmpAcquireLock](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [uint32\\_t](#) lockID, [uint32\\_t](#) acquisitionOrder)

*Callback for the `OmpAcquireLock` snap event record.*

- [typedef](#) [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_OmpFork](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [uint32\\_t](#) numberOfRequestedThreads)

*Callback for the `OmpFork` snap event record.*

- [typedef](#) [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_OmpTaskCreate](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [uint64\\_t](#) taskID)

*Callback for the `OmpTaskCreate` snap event record.*

- [typedef](#) [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_OmpTaskSwitch](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [uint64\\_t](#) taskID)

*Callback for the `OmpTaskSwitch` snap event record.*

- [typedef](#) [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_ParameterInt](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_ParameterRef](#) parameter, [int64\\_t](#) value)

*Callback for the `ParameterInt` snap event record.*

- [typedef](#) [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_ParameterString](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_ParameterRef](#) parameter, [OTF2\\_StringRef](#) string)

*Callback for the `ParameterString` snap event record.*

- [typedef](#) [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_ParameterUnsignedInt](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_ParameterRef](#) parameter, [uint64\\_t](#) value)

*Callback for the `ParameterUnsignedInt` snap event record.*

- [typedef](#) [OTF2\\_CallbackCode](#)(\* [OTF2\\_SnapReaderCallback\\_SnapshotEnd](#))([OTF2\\_LocationRef](#) location, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [uint64\\_t](#) contReadPos)

---

## APPENDIX E. FILE DOCUMENTATION

---

*Callback for the SnapshotEnd snap event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_SnapReaderCallback\_SnapshotStart)(OTF2\_LocationRef location, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t numberOfRecords)

*Callback for the SnapshotStart snap event record.*

- typedef OTF2\_CallbackCode(\* OTF2\_SnapReaderCallback\_Unknown)(OTF2\_LocationRef location, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList)

*Callback for an unknown snap event record.*

- typedef struct OTF2\_SnapReaderCallbacks\_struct OTF2\_SnapReaderCallbacks

*Opaque struct which holds all snap event record callbacks.*

### Functions

- void OTF2\_SnapReaderCallbacks\_Clear (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks)

*Clears a struct for the snap event callbacks.*

- void OTF2\_SnapReaderCallbacks\_Delete (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks)

*Deallocates a struct for the snap event callbacks.*

- OTF2\_SnapReaderCallbacks \* OTF2\_SnapReaderCallbacks\_New (void)

*Allocates a new struct for the snap event callbacks.*

- OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetEnterCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_Enter enterCallback)

*Registers the callback for the Enter snap event.*

- OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMeasurementOnOffCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_MeasurementOnOff measurementOnOffCallback)

*Registers the callback for the MeasurementOnOff snap event.*

- OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMetricCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_Metric metricCallback)

*Registers the callback for the Metric snap event.*

- OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiCollectiveBeginCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_MpiCollectiveBegin mpiCollectiveBeginCallback)

*Registers the callback for the MpiCollectiveBegin snap event.*

### E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetMpiCollectiveEndCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-MpiCollectiveEnd](#) mpiCollectiveEndCallback)  
*Registers the callback for the MpiCollectiveEnd snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetMpiIrecvCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-MpiIrecv](#) mpiIrecvCallback)  
*Registers the callback for the MpiIrecv snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetMpiIrecvRequestCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-MpiIrecvRequest](#) mpiIrecvRequestCallback)  
*Registers the callback for the MpiIrecvRequest snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetMpiIsendCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-MpiIsend](#) mpiIsendCallback)  
*Registers the callback for the MpiIsend snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetMpiIsendCompleteCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-MpiIsendComplete](#) mpiIsendCompleteCallback)  
*Registers the callback for the MpiIsendComplete snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetMpiRecvCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-MpiRecv](#) mpiRecvCallback)  
*Registers the callback for the MpiRecv snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetMpiSendCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-MpiSend](#) mpiSendCallback)  
*Registers the callback for the MpiSend snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetOmpAcquireLockCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-OmpAcquireLock](#) ompAcquireLockCallback)  
*Registers the callback for the OmpAcquireLock snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetOmpForkCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-OmpFork](#) ompForkCallback)  
*Registers the callback for the OmpFork snap event.*
- [OTF2\\_ErrorCode OTF2\\_SnapReaderCallbacks\\_SetOmpTaskCreateCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-OmpTaskCreate](#) ompTaskCreateCallback)  
*Registers the callback for the OmpTaskCreate snap event.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapReaderCallbacks\\_SetOmpTaskSwitchCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-OmpTaskSwitch](#) ompTaskSwitchCallback)

*Registers the callback for the OmpTaskSwitch snap event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapReaderCallbacks\\_SetParameterIntCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-ParameterInt](#) parameterIntCallback)

*Registers the callback for the ParameterInt snap event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapReaderCallbacks\\_SetParameterStringCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-ParameterString](#) parameterStringCallback)

*Registers the callback for the ParameterString snap event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapReaderCallbacks\\_SetParameterUnsignedIntCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-ParameterUnsignedInt](#) parameterUnsignedIntCallback)

*Registers the callback for the ParameterUnsignedInt snap event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapReaderCallbacks\\_SetSnapshotEndCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-SnapshotEnd](#) snapshotEndCallback)

*Registers the callback for the SnapshotEnd snap event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapReaderCallbacks\\_SetSnapshotStartCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-SnapshotStart](#) snapshotStartCallback)

*Registers the callback for the SnapshotStart snap event.*

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapReaderCallbacks\\_SetUnknownCallback](#) ([OTF2\\_SnapReaderCallbacks](#) \*snapReaderCallbacks, [OTF2\\_SnapReaderCallback\\_-Unknown](#) unknownCallback)

*Registers the callback for the Unknown snap event.*

### E.31.1 Detailed Description

This defines the callbacks for the snap reader.

#### Source Template:

*templates/OTF2\_SnapReaderCallbacks.tmpl.h*

## E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

### E.31.2 Typedef Documentation

**E.31.2.1** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -  
Enter)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void  
*userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, OTF2_RegionRef region)`

Callback for the Enter snap event record.

This record exists for each *Enter* event where the corresponding *Leave* event did not occur before the snapshot.

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_-SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent- Time</i>	The original time this event happened.
<i>region</i>	Needs to be defined in a definition record References a <i>Region</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_REGION</i> is available.

#### Since

Version 1.2

#### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.31.2.2** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -  
MeasurementOnOff)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_MeasurementMode  
measurementMode)`

Callback for the MeasurementOnOff snap event record.

The last occurrence of an *MeasurementOnOff* event of this location, if any.

#### Parameters

---

## APPENDIX E. FILE DOCUMENTATION

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>measure-mentMode</i>	Is the measurement turned on ( <a href="#">OTF2_MEASUREMENT_ON</a> ) or off ( <a href="#">OTF2_MEASUREMENT_OFF</a> )?

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.3** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_Metric)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)`

Callback for the Metric snap event record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

As an exception for metric classes where the metric mode detontes an [OTF2\\_METRIC\\_VALUE\\_RELATIVE](#) mode the value indicates the accumulation of all previous metric values recorded.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<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 <a href="#">MetricClass</a> , or a <a href="#">MetricInstance</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_METRIC</a> is available.

### E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding <a href="#">MetricMember</a> definitions.
<i>metricValues</i>	List of metric values.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.4** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_  
MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime)`

Callback for the MpiCollectiveBegin snap event record.

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEventTime</i>	The original time this event happened.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.5** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback -  
MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp,  
OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t  
sizeReceived)`

Callback for the MpiCollectiveEnd snap event record.

Indicates that this location completed a collective operation locally but not all of the participating locations completed the operation yet. The corresponding *Mpi-CollectiveBeginSnap* record is still in the snapshot though.

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>collec-tiveOp</i>	Determines which collective operation it is.
<i>communi-cator</i>	Communicator References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeRe-ceived</i>	Size of the received message.

#### Since

Version 1.2

#### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.



## E.31 oftf2/OTF2\_SnapReaderCallbacks.h File Reference

---

**E.31.2.6** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_  
MpiIrecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength, uint64_t requestID)`

Callback for the MpiIrecv snap event record.

This record exists for each [MpiIrecv](#) event where the matching send message event did not occur on the remote location before the snapshot. This could either be an [MpiSend](#) or an [MpiSendComplete](#) event. Or an [MpiIrecvRequest](#) occurred before this event but the corresponding [MpiIrecv](#) event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing [MpiIrecvRequest](#) is not yet known.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.7** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_  
MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t requestID)`

Callback for the MpiIrecvRequest snap event record.

This record exists for each *MpiIrecvRequest* event where an corresponding *MpiIrecv* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIrecv* did occurred (the *MpiIrecvSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event.

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the requested receive

#### Since

Version 1.2

#### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.31.2.8** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_  
MpiIsend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength, uint64_t requestID)`

Callback for the MpiIsend snap event record.

This record exists for each *MpiIsend* event where an corresponding *MpiIsendComplete* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIsendComplete* did occurred (the *MpiIsendCompleteSnap* record exists in the snapshot) but the matching receive message event

### E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

did not occur on the remote location before the snapshot. (This could either be an [MpiRecv](#) or an [MpiIrecv](#) event.)

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<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 <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.9** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_  
MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t requestID)`

Callback for the `MpiIsendComplete` snap event record.

This record exists for each [MpiIsend](#) event where the corresponding [MpiIsendComplete](#) event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an [MpiRecv](#) or an [MpiIrecv](#) event.) .

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.

## APPENDIX E. FILE DOCUMENTATION

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the related request

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.10** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
MpiRecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength)`

Callback for the MpiRecv snap event record.

This record exists for each [MpiRecv](#) event where the matching send message event did not occur on the remote location before the snapshot. This could either be an [MpiSend](#) or an [MpiSendComplete](#) event. Or an [MpiIrecvRequest](#) occurred before this event but the corresponding [MpiIrecv](#) event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing [MpiIrecvRequest](#) is not yet known.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

## E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.11** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -  
MpiSend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength)`

Callback for the MpiSend snap event record.

This record exists for each [MpiSend](#) event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an [MpiRecv](#) or an [MpiIrecv](#) event. Note that it may so, that a previous [MpiIsend](#) with the same envelope than this one is neither completed not canceled yet, thus the matching receive may already occurred, but the matching couldn't be done yet.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.31.2.12** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_  
OmpAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the OmpAcquireLock snap event record.

This record exists for each *OmpAcquireLock* event where the corresponding *OmpReleaseLock* did not occurred before this snapshot yet.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEventTime</i>	The original time this event happended.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**E.31.2.13** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_  
OmpFork)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t numberOfRequestedThreads)`

Callback for the OmpFork snap event record.

This record exists for each *OmpFork* event where the corresponding *OmpJoin* did not occurred before this snapshot.

### Parameters

<i>location</i>	The location where this snap event happened.
-----------------	--

### E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>num-berOfRe-quest-edThreads</i>	Requested size of the team.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.14** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -  
OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t taskID)`

Callback for the OmpTaskCreate snap event record.

This record exists for each [OmpTaskCreate](#) event where the corresponding [OmpTaskComplete](#) event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the newly created task instance.

#### Since

Version 1.2

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.31.2.15** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -  
OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t taskID)`

Callback for the OmpTaskSwitch snap event record.

This record exists for each [\*OmpTaskSwitch\*](#) event where the corresponding [\*OmpTaskComplete\*](#) event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

## Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterSnapCallbacks</i></a> or <a href="#"><i>OTF2_SnapReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEventTime</i>	The original time this event happended.
<i>taskID</i>	Identifier of the now active task instance.

## Since

Version 1.2

## Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.31.2.16** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -  
ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter,  
int64_t value)`

Callback for the ParameterInt snap event record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greates timestamp less or equal the timestamp of this record.



### E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<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 <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.17** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -  
ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter,  
OTF2_StringRef string)`

Callback for the ParameterString snap event record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

#### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<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 <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.

## APPENDIX E. FILE DOCUMENTATION

---

<i>string</i>	Value: Handle of a string definition References a <a href="#">String</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_STRING</a> is available.
---------------	--

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.18** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -  
ParameterUnsignedInt)(OTF2_LocationRef location,  
OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList  
*attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef  
parameter, uint64_t value)`

Callback for the ParameterUnsignedInt snap event record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<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 <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

**E.31.2.19** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ - SnapshotEnd)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t contReadPos)`

Callback for the SnapshotEnd snap event record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use [OTF2\\_EvtReader\\_Seek](#) with `contReadPos` as the position.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>contRead-Pos</i>	Position to continue reading in the event trace.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**E.31.2.20** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ - SnapshotStart)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t numberOfRecords)`

Callback for the SnapshotStart snap event record.

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one [SnapshotStart](#) record and closes with one [SnapshotEnd](#) record. All snapshot records inbetween are ordered by the `origEventTime`, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterSnapCallbacks</i></a> or <a href="#"><i>OTF2_SnapReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.
<i>num-berOfRecord</i>	Number of snapshot event records in this snapshot. Excluding the <a href="#"><i>Snap-shotEnd</i></a> record.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.31.2.21** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_Unknown)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown snap event record.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#"><i>OTF2_Reader_RegisterSnapCallbacks</i></a> or <a href="#"><i>OTF2_SnapReader_SetCallbacks</i></a> .
<i>attributeList</i>	Additional attributes for this event.

### Since

Version 1.2

### Returns

[\*OTF2\\_CALLBACK\\_SUCCESS\*](#) or [\*OTF2\\_CALLBACK\\_INTERRUPT\*](#).

**E.31.2.22** `typedef struct OTF2_SnapReaderCallbacks_struct OTF2_SnapReaderCallbacks`

Opaque struct which holds all snap event record callbacks.

## E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### E.31.3 Function Documentation

#### E.31.3.1 void OTF2\_SnapReaderCallbacks\_Clear ( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks* )

Clears a struct for the snap event callbacks.

### Parameters

<i>snapReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_SnapReaderCallbacks_New</a> .
----------------------------	---

### Since

Version 1.2

#### E.31.3.2 void OTF2\_SnapReaderCallbacks\_Delete ( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks* )

Deallocates a struct for the snap event callbacks.

### Parameters

<i>snapReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_SnapReaderCallbacks_New</a> .
----------------------------	---

### Since

Version 1.2

#### E.31.3.3 OTF2\_SnapReaderCallbacks\* OTF2\_SnapReaderCallbacks\_New ( void )

Allocates a new struct for the snap event callbacks.

### Since

Version 1.2

### Returns

A newly allocated struct of type [OTF2\\_SnapReaderCallbacks](#).

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.31.3.4 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetEnterCallback**  
( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_Enter *enterCallback* )

Registers the callback for the Enter snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.31.3.5 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMeasurementOnOffCallback**  
( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_MeasurementOnOff  
*measurementOnOffCallback* )

Registers the callback for the MeasurementOnOff snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>measurementOnOffCallback</i>	Function which should be called for all MeasurementOnOff events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

## E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.31.3.6** **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetMetricCallback**  
( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
**OTF2\_SnapReaderCallback\_Metric** *metricCallback* )

Registers the callback for the Metric snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.31.3.7** **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetMpiCollectiveBeginCallback**  
( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
**OTF2\_SnapReaderCallback\_MpiCollectiveBegin**  
*mpiCollectiveBeginCallback* )

Registers the callback for the MpiCollectiveBegin snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin events.

**Since**

Version 1.2

**Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.8** **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetMpiCollectiveEndCallback**  
( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
**OTF2\_SnapReaderCallback\_MpiCollectiveEnd**  
*mpiCollectiveEndCallback* )

Registers the callback for the MpiCollectiveEnd snap event.

**Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEndCallback</i>	Function which should be called for all MpiCollectiveEnd events.

**Since**

Version 1.2

**Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.9** **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetMpiIrecvCallback**  
( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
**OTF2\_SnapReaderCallback\_MpiIrecv** *mpilrecvCallback* )

Registers the callback for the MpiIrecv snap event.

**Parameters**

---



### E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all MpiIrecv events.

#### Since

Version 1.2

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.10 OTF2\_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

---

## APPENDIX E. FILE DOCUMENTATION

---

**E.31.3.11** **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetMpiIsendCallback**  
( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
**OTF2\_SnapReaderCallback\_MpiIsend** *mpiIsendCallback* )

Registers the callback for the MpiIsend snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIsendCallback</i>	Function which should be called for all MpiIsend events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.31.3.12** **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_-SetMpiIsendCompleteCallback** ( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*, **OTF2\_SnapReaderCallback\_-MpiIsendComplete** *mpiIsendCompleteCallback* )

Registers the callback for the MpiIsendComplete snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIsendCompleteCallback</i>	Function which should be called for all MpiIsendComplete events.

### Since

Version 1.2

## E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.13** **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetMpiRecvCallback**  
( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
**OTF2\_SnapReaderCallback\_MpiRecv** *mpiRecvCallback* )

Registers the callback for the MpiRecv snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecvCallback</i>	Function which should be called for all MpiRecv events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.14** **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetMpiSendCallback**  
( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
**OTF2\_SnapReaderCallback\_MpiSend** *mpiSendCallback* )

Registers the callback for the MpiSend snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSendCallback</i>	Function which should be called for all MpiSend events.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.31.3.15** `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpAcquireLockCallback ( OTF2_SnapReaderCallbacks * snapReaderCallbacks, OTF2_SnapReaderCallback_OmpAcquireLock ompAcquireLockCallback )`

Registers the callback for the OmpAcquireLock snap event.

**Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompAcquireLockCallback</i>	Function which should be called for all OmpAcquireLock events.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**E.31.3.16** `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetOmpForkCallback ( OTF2_SnapReaderCallbacks * snapReaderCallbacks, OTF2_SnapReaderCallback_OmpFork ompForkCallback )`

Registers the callback for the OmpFork snap event.

**Parameters**

---

### E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference

---

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompForkCallback</i>	Function which should be called for all OmpFork events.

#### Since

Version 1.2

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.17** **OTF2\_StatusCode** **OTF2\_SnapReaderCallbacks\_SetOmpTaskCreateCallback**  
( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_OmpTaskCreate *ompTaskCreateCallback*  
)

Registers the callback for the OmpTaskCreate snap event.

#### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskCreateCallback</i>	Function which should be called for all OmpTaskCreate events.

#### Since

Version 1.2

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.18**   **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetOmpTaskSwitchCallback**  
                   ( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
                   **OTF2\_SnapReaderCallback\_OmpTaskSwitch** *ompTaskSwitchCallback*  
                   )

Registers the callback for the OmpTaskSwitch snap event.

**Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>ompTaskSwitchCallback</i>	Function which should be called for all OmpTaskSwitch events.

**Since**

Version 1.2

**Returns**

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**E.31.3.19**   **OTF2\_ErrorCode** **OTF2\_SnapReaderCallbacks\_SetParameterIntCallback**  
                   ( **OTF2\_SnapReaderCallbacks** \* *snapReaderCallbacks*,  
                   **OTF2\_SnapReaderCallback\_ParameterInt** *parameterIntCallback* )

Registers the callback for the ParameterInt snap event.

**Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterIntCallback</i>	Function which should be called for all ParameterInt events.

**Since**

Version 1.2

**Returns**

**OTF2\_SUCCESS** if successful

**E.31 otf2/OTF2\_SnapReaderCallbacks.h File Reference**

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.31.3.20** `OTF2_StatusCode OTF2_SnapReaderCallbacks_SetParameterStringCallback ( OTF2_SnapReaderCallbacks * snapReaderCallbacks, OTF2_SnapReaderCallback_ParameterString parameterStringCallback )`

Registers the callback for the ParameterString snap event.

**Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterStringCallback</i>	Function which should be called for all ParameterString events.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful  
***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**E.31.3.21** `OTF2_StatusCode OTF2_SnapReaderCallbacks_SetParameterUnsignedIntCallback ( OTF2_SnapReaderCallbacks * snapReaderCallbacks, OTF2_SnapReaderCallback_ParameterUnsignedInt parameterUnsignedIntCallback )`

Registers the callback for the ParameterUnsignedInt snap event.

**Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt events.

---

## APPENDIX E. FILE DOCUMENTATION

---

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.22** **OTF2\_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

**E.31.3.23** **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.
----------------------------	---------------------------



## E.32 otf2/OTF2\_SnapWriter.h File Reference

---

<i>snapshot-StartCallback</i>	Function which should be called for all SnapshotStart events.
-------------------------------	---

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**E.31.3.24 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks.SetUnknownCallback**  
( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_Unknown *unknownCallback* )

Registers the callback for the Unknown snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>unknownCallback</i>	Function which should be called for all unknown snap events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## E.32 otf2/OTF2\_SnapWriter.h File Reference

This lowest user-visible layer provides write routines to write snapshot records for a single location.

```
#include <stdint.h>
```

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_AttributeList.h>
```

### Typedefs

- typedef struct OTF2\_SnapWriter\_struct [OTF2\\_SnapWriter](#)  
*Keeps all necessary information about the snap writer. See OTF2\_SnapWriter\_-struct for detailed information.*

### Functions

- [OTF2\\_ErrorCode OTF2\\_SnapWriter\\_Enter](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_RegionRef](#) region)  
*Records an Enter snapshot record.*
- [OTF2\\_ErrorCode OTF2\\_SnapWriter\\_GetLocationID](#) (const [OTF2\\_SnapWriter](#) \*writer, [OTF2\\_LocationRef](#) \*locationID)  
*Function to get the location ID of a snap writer object.*
- [OTF2\\_ErrorCode OTF2\\_SnapWriter\\_MeasurementOnOff](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_MeasurementMode](#) measurementMode)  
*Records an MeasurementOnOff snapshot record.*
- [OTF2\\_ErrorCode OTF2\\_SnapWriter\\_Metric](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_MetricRef](#) metric, [uint8\\_t](#) numberOfMetrics, const [OTF2\\_Type](#) \*typeIDs, const [OTF2\\_MetricValue](#) \*metricValues)  
*Records an Metric snapshot record.*
- [OTF2\\_ErrorCode OTF2\\_SnapWriter\\_MpiCollectiveBegin](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime)  
*Records an MpiCollectiveBegin snapshot record.*
- [OTF2\\_ErrorCode OTF2\\_SnapWriter\\_MpiCollectiveEnd](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_CollectiveOp](#) collectiveOp, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) root, [uint64\\_t](#) sizeSent, [uint64\\_t](#) sizeReceived)  
*Records an MpiCollectiveEnd snapshot record.*

## E.32 otf2/OTF2\_SnapWriter.h File Reference

---

- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_MpiIrecv](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint32\\_t](#) sender, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength, [uint64\\_t](#) requestID)  
*Records an MpiIrecv snapshot record.*
- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_MpiIrecvRequest](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint64\\_t](#) requestID)  
*Records an MpiIrecvRequest snapshot record.*
- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_MpiIsend](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint32\\_t](#) receiver, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength, [uint64\\_t](#) requestID)  
*Records an MpiIsend snapshot record.*
- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_MpiIsendComplete](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint64\\_t](#) requestID)  
*Records an MpiIsendComplete snapshot record.*
- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_MpiRecv](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint32\\_t](#) sender, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength)  
*Records an MpiRecv snapshot record.*
- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_MpiSend](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint32\\_t](#) receiver, [OTF2\\_CommRef](#) communicator, [uint32\\_t](#) msgTag, [uint64\\_t](#) msgLength)  
*Records an MpiSend snapshot record.*
- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_OmpAcquireLock](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint32\\_t](#) lockID, [uint32\\_t](#) acquisitionOrder)  
*Records an OmpAcquireLock snapshot record.*
- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_OmpFork](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint32\\_t](#) numberOfRequestedThreads)  
*Records an OmpFork snapshot record.*
- [OTF2\\_ErrorCode](#) [OTF2\\_SnapWriter\\_OmpTaskCreate](#) ([OTF2\\_SnapWriter](#) \*writer, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) snapTime, [OTF2\\_TimeStamp](#) origEventTime, [uint64\\_t](#) taskID)  
*Records an OmpTaskCreate snapshot record.*

---

## APPENDIX E. FILE DOCUMENTATION

---

- `OTF2_ErrorCode OTF2_SnapWriter_OmpTaskSwitch (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint64_t taskID)`

*Records an OmpTaskSwitch snapshot record.*

- `OTF2_ErrorCode OTF2_SnapWriter_ParameterInt (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, int64_t value)`

*Records an ParameterInt snapshot record.*

- `OTF2_ErrorCode OTF2_SnapWriter_ParameterString (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, OTF2_StringRef string)`

*Records an ParameterString snapshot record.*

- `OTF2_ErrorCode OTF2_SnapWriter_ParameterUnsignedInt (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, uint64_t value)`

*Records an ParameterUnsignedInt snapshot record.*

- `OTF2_ErrorCode OTF2_SnapWriter_SnapshotEnd (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, uint64_t contReadPos)`

*Records an SnapshotEnd snapshot record.*

- `OTF2_ErrorCode OTF2_SnapWriter_SnapshotStart (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, uint64_t numberOfRecords)`

*Records an SnapshotStart snapshot record.*

### E.32.1 Detailed Description

This lowest user-visible layer provides write routines to write snapshot records for a single location.

#### Source Template:

*templates/OTF2\_SnapWriter.tmpl.h*

### E.32.2 Typedef Documentation

#### E.32.2.1 typedef struct OTF2\_SnapWriter\_struct OTF2\_SnapWriter

Keeps all necessary information about the snap writer. See `OTF2_SnapWriter_struct` for detailed information.

## E.32 otf2/OTF2\_SnapWriter.h File Reference

---

### Since

Version 1.2

### E.32.3 Function Documentation

#### E.32.3.1 OTF2\_ErrorCode OTF2\_SnapWriter\_Enter ( OTF2\_SnapWriter \* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *snapTime*, OTF2\_TimeStamp *origEventTime*, OTF2\_RegionRef *region* )

Records an Enter snapshot record.

This record exists for each [Enter](#) event where the corresponding [Leave](#) event did not occur before the snapshot.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happended.
<i>region</i>	Needs to be defined in a definition record References a <a href="#">Region</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_REGION</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

#### E.32.3.2 OTF2\_ErrorCode OTF2\_SnapWriter\_GetLocationID ( const OTF2\_SnapWriter \* *writer*, OTF2\_LocationRef \* *locationID* )

Function to get the location ID of a snap writer object.

### Parameters

<i>writer</i>	Snap writer object of interest
<i>locationID</i>	Pointer to a variable where the ID is returned in

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.3** `OTF2_StatusCode OTF2_SnapWriter_MeasurementOnOff (`  
`OTF2_SnapWriter * writer, OTF2_AttributeList * attributeList,`  
`OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime,`  
`OTF2_MeasurementMode measurementMode )`

Records an MeasurementOnOff snapshot record.

The last occurrence of an [MeasurementOnOff](#) event of this location, if any.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happened.
<i>measurementMode</i>	Is the measurement turned on ( <a href="#">OTF2_MEASUREMENT_ON</a> ) or off ( <a href="#">OTF2_MEASUREMENT_OFF</a> )?

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.4** `OTF2_StatusCode OTF2_SnapWriter_Metric ( OTF2_SnapWriter * writer,`  
`OTF2_AttributeList * attributeList, OTF2_TimeStamp snapTime,`  
`OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t`  
`numberOfMetrics, const OTF2_Type * typeIds, const OTF2_MetricValue *`  
`metricValues )`

Records an Metric snapshot record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

## E.32 oftf2/OTF2\_SnapWriter.h File Reference

---

As an exception for metric classes where the metric mode detontes an [OTF2\\_METRIC\\_VALUE\\_RELATIVE](#) mode the value indicates the accumulation of all previous metric values recorded.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>metric</i>	Could be a metric class or a metric instance. References a <a href="#">MetricClass</a> , or a <a href="#">MetricInstance</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_METRIC</a> is available.
<i>numberOf-Metrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types. These types must match that of the corresponding <a href="#">MetricMember</a> definitions.
<i>metricVal-ues</i>	List of metric values.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.5 OTF2\_ErrorCode OTF2\_SnapWriter\_MpiCollectiveBegin (**  
**OTF2\_SnapWriter \* *writer*, OTF2\_AttributeList \* *attributeList*,**  
**OTF2\_TimeStamp *snapTime*, OTF2\_TimeStamp *origEventTime* )**

Records an MpiCollectiveBegin snapshot record.

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.6** `OTF2_ErrorCode OTF2_SnapWriter_MpiCollectiveEnd (`  
`OTF2_SnapWriter * writer, OTF2_AttributeList * attributeList,`  
`OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime,`  
`OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator,`  
`uint32_t root, uint64_t sizeSent, uint64_t sizeReceived )`

Records an MpiCollectiveEnd snapshot record.

Indicates that this location completed a collective operation locally but not all of the participating locations completed the operation yet. The corresponding [Mpi-CollectiveBeginSnap](#) record is still in the snapshot though.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>collec-tiveOp</i>	Determines which collective operation it is.
<i>communi-cator</i>	Communicator References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.
<i>root</i>	MPI rank of root in communicator.
<i>sizeSent</i>	Size of the sent message.
<i>sizeRe-ceived</i>	Size of the received message.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.



## E.32 otf2/OTF2\_SnapWriter.h File Reference

---

**E.32.3.7** `OTF2_ErrorCode OTF2_SnapWriter_Mpilrecv ( OTF2_SnapWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp snapTime,  
OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef  
communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID )`

Records an MpiIrecv snapshot record.

This record exists for each [MpiIrecv](#) event where the matching send message event did not occur on the remote location before the snapshot. This could either be an [MpiSend](#) or an [MpiSendComplete](#) event. Or an [MpiIrecvRequest](#) occurred before this event but the corresponding [MpiIrecv](#) event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing [MpiIrecvRequest](#) is not yet known.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>sender</i>	MPI rank of sender in <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.8** `OTF2_ErrorCode OTF2_SnapWriter_MpilrecvRequest ( OTF2_SnapWriter  
* writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp  
snapTime, OTF2_TimeStamp origEventTime, uint64_t requestID )`

Records an MpiIrecvRequest snapshot record.

This record exists for each [MpiIrecvRequest](#) event where an corresponding [MpiIrecv](#) or [MpiRequestCancelled](#) event did not occur on this location before the snapshot. Or the corresponding [MpiIrecv](#) did occurred (the [MpiIrecvSnap](#) record exists

## APPENDIX E. FILE DOCUMENTATION

in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. This could either be an [MpiRecv](#) or an [MpiIrecv](#) event.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>requestID</i>	ID of the requested receive

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.9** `OTF2_StatusCode OTF2_SnapWriter_Mpilsend ( OTF2_SnapWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp snapTime,  
OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef  
communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID )`

Records an Mpilsend snapshot record.

This record exists for each [Mpilsend](#) event where an corresponding [MpilsendComplete](#) or [MpiRequestCancelled](#) event did not occur on this location before the snapshot. Or the corresponding [MpilsendComplete](#) did occurred (the [MpilsendCompleteSnap](#) record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. (This could either be an [MpiRecv](#) or an [MpiIrecv](#) event.)

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>receiver</i>	MPI rank of receiver in <code>communicator</code> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_COMM</a> is available.

## E.32 otf2/OTF2\_SnapWriter.h File Reference

---

<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.32.3.10 OTF2\_ErrorCode OTF2\_SnapWriter\_MpilsendComplete (**  
**OTF2\_SnapWriter \* *writer*, OTF2\_AttributeList \* *attributeList*,**  
**OTF2\_TimeStamp *snapTime*, OTF2\_TimeStamp *origEventTime*,**  
**uint64\_t *requestID* )**

Records an MpilsendComplete snapshot record.

This record exists for each [\*Mpilsend\*](#) event where the corresponding [\*MpilsendComplete\*](#) event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an [\*MpiRecv\*](#) or an [\*Mpilrecv\*](#) event.) .

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happended.
<i>requestID</i>	ID of the related request

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.32.3.11** `OTF2_ErrorCode OTF2_SnapWriter_MpiRecv ( OTF2_SnapWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp snapTime,  
OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef  
communicator, uint32_t msgTag, uint64_t msgLength )`

Records an MpiRecv snapshot record.

This record exists for each [MpiRecv](#) event where the matching send message event did not occur on the remote location before the snapshot. This could either be an [MpiSend](#) or an [MpiSendComplete](#) event. Or an [MpiRecvRequest](#) occurred before this event but the corresponding [MpiRecv](#) event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing [MpiRecvRequest](#) is not yet known.

#### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>sender</i>	MPI rank of sender in <i>communicator</i> .
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

#### Since

Version 1.2

#### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.12** `OTF2_ErrorCode OTF2_SnapWriter_MpiSend ( OTF2_SnapWriter *  
writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp snapTime,  
OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef  
communicator, uint32_t msgTag, uint64_t msgLength )`

Records an MpiSend snapshot record.

This record exists for each [MpiSend](#) event where the matching receive message event did not occur on the remote location before the snapshot. This could either

## E.32 otf2/OTF2\_SnapWriter.h File Reference

---

be an [MpiRecv](#) or an [MpiIrecv](#) event. Note that it may so, that a previous [MpiIsend](#) with the same envelope than this one is neither completed not canceled yet, thus the matching receive may already occurred, but the matching couldn't be done yet.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>receiver</i>	MPI rank of receiver in communicator.
<i>communi-cator</i>	Communicator ID. References a <a href="#">Comm</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_-COMM</a> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.13** `OTF2_ErrorCode OTF2_SnapWriter_OmpAcquireLock (`  
    `OTF2_SnapWriter * writer, OTF2_AttributeList * attributeList,`  
    `OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime,`  
    `uint32_t lockID, uint32_t acquisitionOrder )`

Records an OmpAcquireLock snapshot record.

This record exists for each [OmpAcquireLock](#) event where the corresponding [OmpReleaseLock](#) did not occurred before this snapshot yet.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>lockID</i>	ID of the lock.

---

## APPENDIX E. FILE DOCUMENTATION

---

<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.
-------------------------	---

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.32.3.14** `OTF2_StatusCode OTF2_SnapWriter_OmpFork ( OTF2_SnapWriter * writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint32_t numberOfRequestedThreads )`

Records an OmpFork snapshot record.

This record exists for each [\*OmpFork\*](#) event where the corresponding [\*OmpJoin\*](#) did not occurred before this snapshot.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happended.
<i>numberOfRequestedThreads</i>	Requested size of the team.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

## E.32 otf2/OTF2\_SnapWriter.h File Reference

---

**E.32.3.15** `OTF2_ErrorCode OTF2_SnapWriter_OmpTaskCreate (`  
    `OTF2_SnapWriter * writer, OTF2_AttributeList * attributeList,`  
    `OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime,`  
    `uint64_t taskID )`

Records an OmpTaskCreate snapshot record.

This record exists for each *OmpTaskCreate* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the newly created task instance.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**E.32.3.16** `OTF2_ErrorCode OTF2_SnapWriter_OmpTaskSwitch (`  
    `OTF2_SnapWriter * writer, OTF2_AttributeList * attributeList,`  
    `OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime,`  
    `uint64_t taskID )`

Records an OmpTaskSwitch snapshot record.

This record exists for each *OmpTaskSwitch* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happended.
<i>taskID</i>	Identifier of the now active task instance.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.17** **OTF2\_ErrorCode** **OTF2\_SnapWriter\_ParameterInt** ( **OTF2\_SnapWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *snapTime*, **OTF2\_TimeStamp** *origEventTime*, **OTF2\_ParameterRef** *parameter*, **int64\_t** *value* )

Records an ParameterInt snapshot record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happened.
<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.18** **OTF2\_ErrorCode** **OTF2\_SnapWriter\_ParameterString** ( **OTF2\_SnapWriter** \* *writer*, **OTF2\_AttributeList** \* *attributeList*, **OTF2\_TimeStamp** *snapTime*, **OTF2\_TimeStamp** *origEventTime*, **OTF2\_ParameterRef** *parameter*, **OTF2\_StringRef** *string* )

Records an ParameterString snapshot record.



## E.32 otf2/OTF2\_SnapWriter.h File Reference

---

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.
<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>string</i>	Value: Handle of a string definition References a <a href="#">String</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_STRING</a> is available.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.19** `OTF2_StatusCode OTF2_SnapWriter_ParameterUnsignedInt (`  
    `OTF2_SnapWriter * writer, OTF2_AttributeList * attributeList,`  
    `OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime,`  
    `OTF2_ParameterRef parameter, uint64_t value )`

Records an ParameterUnsignedInt snapshot record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.

## APPENDIX E. FILE DOCUMENTATION

---

<i>parameter</i>	Parameter ID. References a <a href="#">Parameter</a> definition and will be mapped to the global definition if a mapping table of type <a href="#">OTF2_MAPPING_PARAMETER</a> is available.
<i>value</i>	Value of the recorded parameter.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.20** **OTF2\_ErrorCode** OTF2\_SnapWriter\_SnapshotEnd ( OTF2\_SnapWriter \*  
*writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *snapTime*,  
uint64\_t *contReadPos* )

Records an SnapshotEnd snapshot record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use [OTF2\\_EvtReader\\_Seek](#) with `contReadPos` as the position.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>contRead-Pos</i>	Position to continue reading in the event trace.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**E.32.3.21** **OTF2\_ErrorCode** OTF2\_SnapWriter\_SnapshotStart ( OTF2\_SnapWriter  
\* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp  
*snapTime*, uint64\_t *numberOfRecords* )

Records an SnapshotStart snapshot record.

### E.33 otf2/OTF2\_Thumbnail.h File Reference

---

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one [SnapshotStart](#) record and closes with one [SnapshotEnd](#) record. All snapshot records inbetween are ordered by the `origEventTime`, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

#### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>numberOfRecord</i>	Number of snapshot event records in this snapshot. Excluding the <a href="#">SnapshotEnd</a> record.

#### Since

Version 1.2

#### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

### E.33 otf2/OTF2\_Thumbnail.h File Reference

This lowest user-visible layer provides write routines to read and write thumbnail data.

```
#include <stdint.h>
```

```
#include <otf2/OTF2_GeneralDefinitions.h>
```

#### Typedefs

- typedef struct OTF2\_ThumbReader\_struct [OTF2\\_ThumbReader](#)  
*Keeps all necessary information about the event reader. See OTF2\_ThumbReader\_struct for detailed information.*
- typedef struct OTF2\_ThumbWriter\_struct [OTF2\\_ThumbWriter](#)  
*Keeps all necessary information about the thumb writer. See OTF2\_ThumbWriter\_struct for detailed information.*

## APPENDIX E. FILE DOCUMENTATION

### Functions

- [OTF2\\_ErrorCode](#) [OTF2\\_ThumbReader\\_GetHeader](#) ([OTF2\\_ThumbReader](#) \*reader, char \*\*const name, char \*\*const description, [OTF2\\_ThumbnailType](#) \*type, uint32\_t \*numberOfSamples, uint32\_t \*numberOfMetrics, uint64\_t \*\*refsToDefs)

*Reads a thumbnail header.*

- [OTF2\\_ErrorCode](#) [OTF2\\_ThumbReader\\_ReadSample](#) ([OTF2\\_ThumbReader](#) \*reader, uint64\_t \*baseline, uint32\_t numberOfMetrics, uint64\_t \*metricSamples)

*Reads a thumbnail sample.*

- [OTF2\\_ErrorCode](#) [OTF2\\_ThumbWriter\\_WriteSample](#) ([OTF2\\_ThumbWriter](#) \*writer, uint64\_t baseline, uint32\_t numberOfMetrics, const uint64\_t \*metricSamples)

*Writes a thumbnail sample.*

### E.33.1 Detailed Description

This lowest user-visible layer provides write routines to read and write thumbnail data.

### E.33.2 Function Documentation

**E.33.2.1** [OTF2\\_ErrorCode](#) [OTF2\\_ThumbReader\\_GetHeader](#) ( [OTF2\\_ThumbReader](#) \* *reader*, char \*\*const *name*, char \*\*const *description*, [OTF2\\_ThumbnailType](#) \* *type*, uint32\_t \* *numberOfSamples*, uint32\_t \* *numberOfMetrics*, uint64\_t \*\* *refsToDefs* )

Reads a thumbnail header.

A thumbnail header contains some meta information for a thumbnail.

### Parameters

	<i>reader</i>	Reader object.
out	<i>name</i>	Name of thumbnail.
out	<i>description</i>	Description of thumbnail.
out	<i>type</i>	Type of thumbnail.
out	<i>numberOfSamples</i>	Number of samples.
out	<i>numberOfMetrics</i>	Number of metrics.
out	<i>refsToDefs</i>	The sorted set of references to definitions used in an thumbnail sample.

### E.33 otf2/OTF2\_Thumbnail.h File Reference

---

#### Since

Version 1.2

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.33.2.2** **OTF2\_ErrorCode** **OTF2\_ThumbReader\_ReadSample** (  
    **OTF2\_ThumbReader** \* *reader*, **uint64\_t** \* *baseline*, **uint32\_t**  
    *numberOfMetrics*, **uint64\_t** \* *metricSamples* )

Reads a thumbnail sample.

#### Parameters

	<i>reader</i>	Reader object.
out	<i>baseline</i>	Baseline for this sample. If zero, the baseline is the sum of all metric values in this sample.
	<i>numberOfMetrics</i>	Number of metric sample values.
out	<i>metricSamples</i>	Metric sample values.

#### Since

Version 1.2

#### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**E.33.2.3** **OTF2\_ErrorCode** **OTF2\_ThumbWriter\_WriteSample** (  
    **OTF2\_ThumbWriter** \* *writer*, **uint64\_t** *baseline*, **uint32\_t** *numberOfMetrics*,  
    **const uint64\_t** \* *metricSamples* )

Writes a thumbnail sample.

#### Parameters

	<i>writer</i>	Writer object.
	<i>baseline</i>	Baseline for this sample. If zero, the baseline is the sum of all metric values in this sample.
	<i>numberOfMetrics</i>	Number of metric sample values.

## APPENDIX E. FILE DOCUMENTATION

---

<i>metricSamples</i>	Metric sample values.
----------------------	-----------------------

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

# Index

- Controlling OTF2 flush behavior in writing mode, [91](#)
  - [OTF2\\_PostFlushCallback](#), [92](#)
  - [OTF2\\_PreFlushCallback](#), [93](#)
- How to use the attribute list for writing additional attributes to event records, [91](#)
- List of all definition records, [22](#)
- List of all event records, [39](#)
- List of all marker records, [72](#)
- List of all snapshot records, [73](#)
- Memory pooling for OTF2, [93](#)
  - [OTF2\\_MemoryAllocate](#), [94](#)
  - [OTF2\\_MemoryFreeAll](#), [94](#)
- Operating OTF2 in an collective context, [95](#)
  - [OTF2\\_Collectives\\_Barrier](#), [97](#)
  - [OTF2\\_Collectives\\_Bcast](#), [98](#)
  - [OTF2\\_Collectives\\_CreateLocalComm](#), [98](#)
  - [OTF2\\_Collectives\\_FreeLocalComm](#), [98](#)
  - [OTF2\\_Collectives\\_Gather](#), [99](#)
  - [OTF2\\_Collectives\\_Gatherv](#), [99](#)
  - [OTF2\\_Collectives\\_GetRank](#), [99](#)
  - [OTF2\\_Collectives\\_GetSize](#), [100](#)
  - [OTF2\\_Collectives\\_Release](#), [100](#)
  - [OTF2\\_Collectives\\_Scatter](#), [100](#)
  - [OTF2\\_Collectives\\_Scatterv](#), [101](#)
- OTF2 callbacks, [91](#)
- OTF2 config tool, [17](#)
- OTF2 estimator tool, [20](#)
- OTF2 marker tool, [19](#)
- OTF2 print tool, [18](#)
- OTF2 records, [21](#)
- OTF2 snapshots tool, [19](#)
- OTF2 usage examples, [86](#)
- [otf2/otf2.h](#), [136](#)
- [otf2/OTF2\\_Archive.h](#), [136](#)
- [otf2/OTF2\\_AttributeList.h](#), [167](#)
- [otf2/OTF2\\_Callbacks.h](#), [192](#)
- [otf2/OTF2\\_Definitions.h](#), [195](#)
- [otf2/OTF2\\_DefReader.h](#), [209](#)
- [otf2/OTF2\\_DefReaderCallbacks.h](#), [212](#)
- [otf2/OTF2\\_DefWriter.h](#), [250](#)
- [otf2/OTF2\\_ErrorCodes.h](#), [127](#)
- [otf2/OTF2\\_Events.h](#), [269](#)
- [otf2/OTF2\\_EventSizeEstimator.h](#), [275](#)
- [otf2/OTF2\\_EvtReader.h](#), [310](#)
- [otf2/OTF2\\_EvtReaderCallbacks.h](#), [315](#)
- [otf2/OTF2\\_EvtWriter.h](#), [402](#)
- [otf2/OTF2\\_GeneralDefinitions.h](#), [448](#)
- [otf2/OTF2\\_GlobalDefReader.h](#), [461](#)
- [otf2/OTF2\\_GlobalDefReaderCallbacks.h](#), [463](#)
- [otf2/OTF2\\_GlobalDefWriter.h](#), [502](#)
- [otf2/OTF2\\_GlobalEvtReader.h](#), [522](#)
- [otf2/OTF2\\_GlobalEvtReaderCallbacks.h](#), [525](#)
- [otf2/OTF2\\_GlobalSnapReader.h](#), [611](#)
- [otf2/OTF2\\_GlobalSnapReaderCallbacks.h](#), [613](#)
- [otf2/OTF2\\_IdMap.h](#), [649](#)
- [otf2/OTF2\\_Marker.h](#), [655](#)
- [otf2/OTF2\\_MarkerReader.h](#), [657](#)
- [otf2/OTF2\\_MarkerReaderCallbacks.h](#), [659](#)
- [otf2/OTF2\\_MarkerWriter.h](#), [665](#)
- [otf2/OTF2\\_MPI\\_Collectives.h](#), [667](#)
- [otf2/OTF2\\_Reader.h](#), [670](#)

otf2/OTF2\_SnapReader.h, [704](#)  
otf2/OTF2\_SnapReaderCallbacks.h, [707](#)  
otf2/OTF2\_SnapWriter.h, [741](#)  
otf2/OTF2\_Thumbnail.h, [759](#)  
OTF2\_ABORT  
    OTF2\_ErrorCodes.h, [132](#)  
OTF2\_BASE\_BINARY  
    OTF2\_Definitions.h, [203](#)  
OTF2\_BASE\_DECIMAL  
    OTF2\_Definitions.h, [203](#)  
OTF2\_CALLBACK\_ERROR  
    OTF2\_GeneralDefinitions.h, [456](#)  
OTF2\_CALLBACK\_INTERRUPT  
    OTF2\_GeneralDefinitions.h, [456](#)  
OTF2\_CALLBACK\_SUCCESS  
    OTF2\_GeneralDefinitions.h, [456](#)  
OTF2\_CART\_PERIODIC\_FALSE  
    OTF2\_Definitions.h, [200](#)  
OTF2\_CART\_PERIODIC\_TRUE  
    OTF2\_Definitions.h, [200](#)  
OTF2\_COLLECTIVE\_OP\_ALLGATHER  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_ALLGATHERV  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_ALLOCATE  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_ALLREDUCE  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_ALLTOALL  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_ALLTOALLV  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_ALLTOALLW  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_BARRIER  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_BCAST  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE\_-  
    AND\_ALLOCATE  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_DEALLOCATE  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_DESTROY\_-  
    HANDLE  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_DESTROY\_-  
    HANDLE\_AND\_DEALLOCATE  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_EXSCAN  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_GATHER  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_GATHERV  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_REDUCE  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_REDUCE\_-  
    SCATTER  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_REDUCE\_-  
    SCATTER\_BLOCK  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_SCAN  
    OTF2\_Events.h, [273](#)  
OTF2\_COLLECTIVE\_OP\_SCATTER  
    OTF2\_Events.h, [272](#)  
OTF2\_COLLECTIVE\_OP\_SCATTERV  
    OTF2\_Events.h, [272](#)  
OTF2\_COMPRESSION\_NONE  
    OTF2\_GeneralDefinitions.h, [456](#)  
OTF2\_COMPRESSION\_UNDEFINED  
    OTF2\_GeneralDefinitions.h, [456](#)  
OTF2\_COMPRESSION\_ZLIB  
    OTF2\_GeneralDefinitions.h, [456](#)  
OTF2\_Definitions.h  
    OTF2\_BASE\_BINARY, [203](#)  
    OTF2\_BASE\_DECIMAL, [203](#)  
    OTF2\_CART\_PERIODIC\_FALSE,  
        [200](#)  
    OTF2\_CART\_PERIODIC\_TRUE, [200](#)  
    OTF2\_GROUP\_FLAG\_GLOBAL\_-  
        MEMBERS, [201](#)  
    OTF2\_GROUP\_FLAG\_NONE, [201](#)  
    OTF2\_GROUP\_TYPE\_COMM\_GROUP,  
        [201](#)



## INDEX

---

OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS, 201  
OTF2\_GROUP\_TYPE\_COMM\_SELF, 202  
OTF2\_GROUP\_TYPE\_LOCATIONS, 201  
OTF2\_GROUP\_TYPE\_METRIC, 201  
OTF2\_GROUP\_TYPE\_REGIONS, 201  
OTF2\_GROUP\_TYPE\_UNKNOWN, 201  
OTF2\_LOCATION\_GROUP\_TYPE\_-  
PROCESS, 202  
OTF2\_LOCATION\_GROUP\_TYPE\_-  
UNKNOWN, 202  
OTF2\_LOCATION\_TYPE\_CPU\_THREAD, 202  
OTF2\_LOCATION\_TYPE\_GPU, 202  
OTF2\_LOCATION\_TYPE\_METRIC, 202  
OTF2\_LOCATION\_TYPE\_UNKNOWN, 202  
OTF2\_METRIC\_ABSOLUTE\_LAST, 203  
OTF2\_METRIC\_ABSOLUTE\_NEXT, 203  
OTF2\_METRIC\_ABSOLUTE\_POINT, 203  
OTF2\_METRIC\_ACCUMULATED\_-  
LAST, 203  
OTF2\_METRIC\_ACCUMULATED\_-  
NEXT, 203  
OTF2\_METRIC\_ACCUMULATED\_-  
POINT, 203  
OTF2\_METRIC\_ACCUMULATED\_-  
START, 203  
OTF2\_METRIC\_ASYNCHRONOUS, 204  
OTF2\_METRIC\_RELATIVE\_LAST, 203  
OTF2\_METRIC\_RELATIVE\_NEXT, 203  
OTF2\_METRIC\_RELATIVE\_POINT, 203  
OTF2\_METRIC\_SYNCHRONOUS, 204  
OTF2\_METRIC\_SYNCHRONOUS\_-  
STRICT, 204  
OTF2\_METRIC\_TIMING\_LAST, 205  
OTF2\_METRIC\_TIMING\_MASK, 205  
OTF2\_METRIC\_TIMING\_NEXT, 205  
OTF2\_METRIC\_TIMING\_POINT, 205  
OTF2\_METRIC\_TIMING\_START, 205  
OTF2\_METRIC\_TYPE\_OTHER, 205  
OTF2\_METRIC\_TYPE\_PAPI, 205  
OTF2\_METRIC\_TYPE\_RUSAGE, 205  
OTF2\_METRIC\_TYPE\_USER, 205  
OTF2\_METRIC\_VALUE\_ABSOLUTE, 206  
OTF2\_METRIC\_VALUE\_ACCUMULATED, 206  
OTF2\_METRIC\_VALUE\_MASK, 206  
OTF2\_METRIC\_VALUE\_RELATIVE, 206  
OTF2\_PARAMETER\_TYPE\_INT64, 206  
OTF2\_PARAMETER\_TYPE\_STRING, 206  
OTF2\_PARAMETER\_TYPE\_UINT64, 206  
OTF2\_RECORDER\_KIND\_ABSTRACT, 206  
OTF2\_RECORDER\_KIND\_CPU, 206  
OTF2\_RECORDER\_KIND\_GPU, 206  
OTF2\_RECORDER\_KIND\_UNKNOWN, 206  
OTF2\_REGION\_FLAG\_DYNAMIC, 207  
OTF2\_REGION\_FLAG\_NONE, 207  
OTF2\_REGION\_FLAG\_PHASE, 207  
OTF2\_REGION\_ROLE\_ARTIFICIAL, 208  
OTF2\_REGION\_ROLE\_ATOMIC, 208

OTF2\_REGION\_ROLE\_BARRIER, 208  
 OTF2\_REGION\_ROLE\_CODE, 207  
 OTF2\_REGION\_ROLE\_COLL\_ALL2ALL, 208  
 OTF2\_REGION\_ROLE\_COLL\_ALL2ONE, 208  
 OTF2\_REGION\_ROLE\_COLL\_ONE2ALL, 208  
 OTF2\_REGION\_ROLE\_COLL\_OTHER, 208  
 OTF2\_REGION\_ROLE\_CRITICAL, 208  
 OTF2\_REGION\_ROLE\_CRITICAL\_-SBLOCK, 208  
 OTF2\_REGION\_ROLE\_DATA\_TRANSFER, 208  
 OTF2\_REGION\_ROLE\_FILE\_IO, 208  
 OTF2\_REGION\_ROLE\_FLUSH, 208  
 OTF2\_REGION\_ROLE\_FUNCTION, 207  
 OTF2\_REGION\_ROLE\_IMPLICIT\_-BARRIER, 208  
 OTF2\_REGION\_ROLE\_LOOP, 207  
 OTF2\_REGION\_ROLE\_MASTER, 208  
 OTF2\_REGION\_ROLE\_ORDERED, 208  
 OTF2\_REGION\_ROLE\_ORDERED\_-SBLOCK, 208  
 OTF2\_REGION\_ROLE\_PARALLEL, 207  
 OTF2\_REGION\_ROLE\_POINT2POINT, 208  
 OTF2\_REGION\_ROLE\_RMA, 208  
 OTF2\_REGION\_ROLE\_SECTION, 207  
 OTF2\_REGION\_ROLE\_SECTIONS, 207  
 OTF2\_REGION\_ROLE\_SINGLE, 207  
 OTF2\_REGION\_ROLE\_SINGLE\_-SBLOCK, 207  
 OTF2\_REGION\_ROLE\_TASK, 208  
 OTF2\_REGION\_ROLE\_TASK\_CREATE, 208  
 OTF2\_REGION\_ROLE\_TASK\_WAIT, 208  
 OTF2\_REGION\_ROLE\_THREAD\_-CREATE, 208  
 OTF2\_REGION\_ROLE\_THREAD\_-WAIT, 209  
 OTF2\_REGION\_ROLE\_UNKNOWN, 207  
 OTF2\_REGION\_ROLE\_WORKSHARE, 207  
 OTF2\_REGION\_ROLE\_WRAPPER, 207  
 OTF2\_SCOPE\_GROUP, 204  
 OTF2\_SCOPE\_LOCATION, 204  
 OTF2\_SCOPE\_LOCATION\_GROUP, 204  
 OTF2\_SCOPE\_SYSTEM\_TREE\_-NODE, 204  
 OTF2\_SYSTEM\_TREE\_DOMAIN\_-CACHE, 209  
 OTF2\_SYSTEM\_TREE\_DOMAIN\_-CORE, 209  
 OTF2\_SYSTEM\_TREE\_DOMAIN\_-MACHINE, 209  
 OTF2\_SYSTEM\_TREE\_DOMAIN\_-NUMA, 209  
 OTF2\_SYSTEM\_TREE\_DOMAIN\_-PU, 209  
 OTF2\_SYSTEM\_TREE\_DOMAIN\_-SHARED\_MEMORY, 209  
 OTF2\_SYSTEM\_TREE\_DOMAIN\_-SOCKET, 209  
 OTF2\_DEPRECATED  
 OTF2\_ErrorCodes.h, 131  
 OTF2\_ERROR\_COLLECTIVE\_CALLBACK  
 OTF2\_ErrorCodes.h, 135  
 OTF2\_ERROR\_DUPLICATE\_MAPPING\_-TABLE  
 OTF2\_ErrorCodes.h, 135  
 OTF2\_ERROR\_E2BIG  
 OTF2\_ErrorCodes.h, 132  
 OTF2\_ERROR\_EACCES

## INDEX

---

OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">132</a>
OTF2_ERROR_EADDRNOTAVAIL	OTF2_ERROR_EISCONN
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">132</a>
OTF2_ERROR_EAFNOSUPPORT	OTF2_ERROR_EISDIR
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">132</a>
OTF2_ERROR_EAGAIN	OTF2_ERROR_ELOOP
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">132</a>
OTF2_ERROR_EALREADY	OTF2_ERROR_EMFILE
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">132</a>
OTF2_ERROR_EBADF	OTF2_ERROR_EMLINK
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">132</a>
OTF2_ERROR_EBADMSG	OTF2_ERROR_MSGSIZE
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EBUSY	OTF2_ERROR_EMULTIHOP
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_ECANCELED	OTF2_ERROR_ENAMETOOLONG
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_ECHILD	OTF2_ERROR_END_OF_BUFFER
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ECONNREFUSED	OTF2_ERROR_END_OF_FUNCTION
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ECONNRESET	OTF2_ERROR_ENETDOWN
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EDEADLK	OTF2_ERROR_ENETRESET
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EDESTADDRREQ	OTF2_ERROR_ENETUNREACH
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EDOM	OTF2_ERROR_ENFILE
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EDQUOT	OTF2_ERROR_ENOBUFS
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EEXIST	OTF2_ERROR_ENODATA
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EFAULT	OTF2_ERROR_ENODEV
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EFBIG	OTF2_ERROR_ENOENT
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EINPROGRESS	OTF2_ERROR_ENOEXEC
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EINTR	OTF2_ERROR_ENOLCK
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EINVAL	OTF2_ERROR_ENOLINK
OTF2_ErrorCodes.h, <a href="#">132</a>	OTF2_ErrorCodes.h, <a href="#">133</a>
OTF2_ERROR_EIO	OTF2_ERROR_ENOMEM

OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOMSG	OTF2_ERROR_ESPIPE
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOPROTOPT	OTF2_ERROR_ESRCH
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOSPC	OTF2_ERROR_ESTALE
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOSR	OTF2_ERROR_ETIME
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOSTR	OTF2_ERROR_ETIMEDOUT
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOSYS	OTF2_ERROR_ETXTBSY
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOTCONN	OTF2_ERROR_EWOULDBLOCK
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOTDIR	OTF2_ERROR_EXDEV
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOTEMPTY	OTF2_ERROR_FILE_CAN_NOT_OPEN
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_ENOTSOCK	OTF2_ERROR_FILE_COMPRESSION_-
OTF2_ErrorCodes.h, <a href="#">133</a>	NOT_SUPPORTED
OTF2_ERROR_ENOTSUP	OTF2_ErrorCodes.h, <a href="#">135</a>
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ERROR_FILE_INTERACTION
OTF2_ERROR_ENOTTY	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ERROR_FILE_SUBSTRATE_NOT_-
OTF2_ERROR_ENXIO	SUPPORTED
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">135</a>
OTF2_ERROR_EOPNOTSUPP	OTF2_ERROR_INDEX_OUT_OF_BOUNDS
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_EOVERFLOW	OTF2_ERROR_INTEGRITY_FAULT
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ERROR_EPERM	OTF2_ERROR_INTERRUPTED_BY_-
OTF2_ErrorCodes.h, <a href="#">133</a>	CALLBACK
OTF2_ERROR_EPIPE	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ERROR_INVALID
OTF2_ERROR_EPROTO	OTF2_ErrorCodes.h, <a href="#">132</a>
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ERROR_INVALID_ARGUMENT
OTF2_ERROR_EPROTONOSUPPORT	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ErrorCodes.h, <a href="#">133</a>	OTF2_ERROR_INVALID_CALL
OTF2_ERROR_EPROTOTYPE	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ErrorCodes.h, <a href="#">134</a>	OTF2_ERROR_INVALID_DATA
OTF2_ERROR_ERANGE	OTF2_ErrorCodes.h, <a href="#">134</a>
OTF2_ErrorCodes.h, <a href="#">134</a>	OTF2_ERROR_INVALID_FILE_MODE_-
OTF2_ERROR_EROFS	TRANSITION

## INDEX

---

- OTF2\_ErrorCodes.h, [135](#)
- OTF2\_ERROR\_INVALID\_LINENO
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ERROR\_INVALID\_RECORD
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ERROR\_INVALID\_SIZE\_GIVEN
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ERROR\_MEM\_ALLOC\_FAILED
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ERROR\_MEM\_FAULT
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ERROR\_PROCESSED\_WITH\_FAULTS
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ERROR\_PROPERTY\_EXISTS
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ERROR\_PROPERTY\_NAME\_INVALID
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ERROR\_PROPERTY\_NOT\_FOUND
  - OTF2\_ErrorCodes.h, [135](#)
- OTF2\_ERROR\_PROPERTY\_VALUE\_INVALID
  - OTF2\_ErrorCodes.h, [135](#)
- OTF2\_ERROR\_UNKNOWN\_TYPE
  - OTF2\_ErrorCodes.h, [134](#)
- OTF2\_ErrorCodes.h
  - OTF2\_ABORT, [132](#)
  - OTF2\_DEPRECATED, [131](#)
  - OTF2\_ERROR\_COLLECTIVE\_CALLBACK, [134](#)
  - OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE, [135](#)
  - OTF2\_ERROR\_E2BIG, [132](#)
  - OTF2\_ERROR\_EACCES, [132](#)
  - OTF2\_ERROR\_EADDRNOTAVAIL, [132](#)
  - OTF2\_ERROR\_EAFNOSUPPORT, [132](#)
  - OTF2\_ERROR\_EAGAIN, [132](#)
  - OTF2\_ERROR\_EALREADY, [132](#)
  - OTF2\_ERROR\_EBADF, [132](#)
  - OTF2\_ERROR\_EBADMSG, [132](#)
  - OTF2\_ERROR\_EBUSY, [132](#)
  - OTF2\_ERROR\_ECANCELED, [132](#)
  - OTF2\_ERROR\_ECHILD, [132](#)
  - OTF2\_ERROR\_ECONNREFUSED, [132](#)
  - OTF2\_ERROR\_ECONNRESET, [132](#)
  - OTF2\_ERROR\_EDEADLK, [132](#)
  - OTF2\_ERROR\_EDESTADDRREQ, [132](#)
  - OTF2\_ERROR\_EDOM, [132](#)
  - OTF2\_ERROR\_EDQUOT, [132](#)
  - OTF2\_ERROR\_EEXIST, [132](#)
  - OTF2\_ERROR\_EFAULT, [132](#)
  - OTF2\_ERROR\_EFBIG, [132](#)
  - OTF2\_ERROR\_EINPROGRESS, [132](#)
  - OTF2\_ERROR\_EINTR, [132](#)
  - OTF2\_ERROR\_EINVAL, [132](#)
  - OTF2\_ERROR\_EIO, [132](#)
  - OTF2\_ERROR\_EISCONN, [132](#)
  - OTF2\_ERROR\_EISDIR, [132](#)
  - OTF2\_ERROR\_ELOOP, [132](#)
  - OTF2\_ERROR\_EMFILE, [132](#)
  - OTF2\_ERROR\_EMLINK, [132](#)
  - OTF2\_ERROR EMSGSIZE, [133](#)
  - OTF2\_ERROR\_EMULTIHOP, [133](#)
  - OTF2\_ERROR\_ENAMETOOLONG, [133](#)
  - OTF2\_ERROR\_END\_OF\_BUFFER, [134](#)
  - OTF2\_ERROR\_END\_OF\_FUNCTION, [134](#)
  - OTF2\_ERROR\_ENETDOWN, [133](#)
  - OTF2\_ERROR\_ENETRESET, [133](#)
  - OTF2\_ERROR\_ENETUNREACH, [133](#)
  - OTF2\_ERROR\_ENFILE, [133](#)
  - OTF2\_ERROR\_ENOBUFS, [133](#)
  - OTF2\_ERROR\_ENODATA, [133](#)
  - OTF2\_ERROR\_ENODEV, [133](#)
  - OTF2\_ERROR\_ENOENT, [133](#)
  - OTF2\_ERROR\_ENOEXEC, [133](#)
  - OTF2\_ERROR\_ENOLCK, [133](#)
  - OTF2\_ERROR\_ENOLINK, [133](#)
  - OTF2\_ERROR\_ENOMEM, [133](#)
  - OTF2\_ERROR\_ENOMSG, [133](#)
  - OTF2\_ERROR\_ENOPROTOPT, [133](#)

- 
- OTF2\_ERROR\_ENOSPC, [133](#)
  - OTF2\_ERROR\_ENOSR, [133](#)
  - OTF2\_ERROR\_ENOSTR, [133](#)
  - OTF2\_ERROR\_ENOSYS, [133](#)
  - OTF2\_ERROR\_ENOTCONN, [133](#)
  - OTF2\_ERROR\_ENOTDIR, [133](#)
  - OTF2\_ERROR\_ENOTEMPTY, [133](#)
  - OTF2\_ERROR\_ENOTSOCK, [133](#)
  - OTF2\_ERROR\_ENOTSUP, [133](#)
  - OTF2\_ERROR\_ENOTTY, [133](#)
  - OTF2\_ERROR\_ENXIO, [133](#)
  - OTF2\_ERROR\_EOPNOTSUPP, [133](#)
  - OTF2\_ERROR\_EOVERFLOW, [133](#)
  - OTF2\_ERROR\_EPERM, [133](#)
  - OTF2\_ERROR\_EPIPE, [133](#)
  - OTF2\_ERROR\_EPROTO, [133](#)
  - OTF2\_ERROR\_EPROTONOSUPPORT, [133](#)
  - OTF2\_ERROR\_EPROTOTYPE, [134](#)
  - OTF2\_ERROR\_ERANGE, [134](#)
  - OTF2\_ERROR\_EROFS, [134](#)
  - OTF2\_ERROR\_ESPIPE, [134](#)
  - OTF2\_ERROR\_ESRCH, [134](#)
  - OTF2\_ERROR\_ESTALE, [134](#)
  - OTF2\_ERROR\_ETIME, [134](#)
  - OTF2\_ERROR\_ETIMEDOUT, [134](#)
  - OTF2\_ERROR\_ETXTBSY, [134](#)
  - OTF2\_ERROR\_EWOULDBLOCK, [134](#)
  - OTF2\_ERROR\_EXDEV, [134](#)
  - OTF2\_ERROR\_FILE\_CAN\_NOT\_OPEN, [134](#)
  - OTF2\_ERROR\_FILE\_COMPRESSION\_NOT\_SUPPORTED, [135](#)
  - OTF2\_ERROR\_FILE\_INTERACTION, [134](#)
  - OTF2\_ERROR\_FILE\_SUBSTRATE\_NOT\_SUPPORTED, [135](#)
  - OTF2\_ERROR\_INDEX\_OUT\_OF\_BOUNDS, [134](#)
  - OTF2\_ERROR\_INTEGRITY\_FAULT, [134](#)
  - OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK, [134](#)
  - OTF2\_ERROR\_INVALID, [132](#)
  - OTF2\_ERROR\_INVALID\_ARGUMENT, [134](#)
  - OTF2\_ERROR\_INVALID\_CALL, [134](#)
  - OTF2\_ERROR\_INVALID\_DATA, [134](#)
  - OTF2\_ERROR\_INVALID\_FILE\_MODE\_TRANSITION, [135](#)
  - OTF2\_ERROR\_INVALID\_LINENO, [134](#)
  - OTF2\_ERROR\_INVALID\_RECORD, [134](#)
  - OTF2\_ERROR\_INVALID\_SIZE\_GIVEN, [134](#)
  - OTF2\_ERROR\_MEM\_ALLOC\_FAILED, [134](#)
  - OTF2\_ERROR\_MEM\_FAULT, [134](#)
  - OTF2\_ERROR\_PROCESSED\_WITH\_FAULTS, [134](#)
  - OTF2\_ERROR\_PROPERTY\_EXISTS, [134](#)
  - OTF2\_ERROR\_PROPERTY\_NAME\_INVALID, [134](#)
  - OTF2\_ERROR\_PROPERTY\_NOT\_FOUND, [135](#)
  - OTF2\_ERROR\_PROPERTY\_VALUE\_INVALID, [135](#)
  - OTF2\_ERROR\_UNKNOWN\_TYPE, [134](#)
  - OTF2\_SUCCESS, [132](#)
  - OTF2\_WARNING, [132](#)
  - OTF2\_Events.h
  - OTF2\_COLLECTIVE\_OP\_ALLGATHER, [272](#)
  - OTF2\_COLLECTIVE\_OP\_ALLGATHERV, [272](#)
  - OTF2\_COLLECTIVE\_OP\_ALLOCATE, [273](#)
  - OTF2\_COLLECTIVE\_OP\_ALLREDUCE, [272](#)
  - OTF2\_COLLECTIVE\_OP\_ALLTOALL, [272](#)
  - OTF2\_COLLECTIVE\_OP\_ALLTOALLV, [272](#)
-

## INDEX

---

OTF2\_COLLECTIVE\_OP\_ALLTOALLWOTF2\_RMA\_ATOMIC\_TYPE\_FETCH\_-  
272 AND\_ADD, 274

OTF2\_COLLECTIVE\_OP\_BARRIER, OTF2\_RMA\_ATOMIC\_TYPE\_FETCH\_-  
272 AND\_INCREMENT, 274

OTF2\_COLLECTIVE\_OP\_BCAST, OTF2\_RMA\_ATOMIC\_TYPE\_INCREMENT,  
272 274

OTF2\_COLLECTIVE\_OP\_CREATE\_- OTF2\_RMA\_ATOMIC\_TYPE\_SWAP,  
HANDLE, 273 274

OTF2\_COLLECTIVE\_OP\_CREATE\_- OTF2\_RMA\_ATOMIC\_TYPE\_TEST\_-  
HANDLE\_AND\_ALLOCATE, AND\_SET, 274  
273

OTF2\_COLLECTIVE\_OP\_DEALLOCATE, OTF2\_RMA\_SYNC\_LEVEL\_MEMORY,  
273 275

OTF2\_COLLECTIVE\_OP\_DESTROY\_- OTF2\_RMA\_SYNC\_LEVEL\_NONE,  
HANDLE, 273 275

OTF2\_COLLECTIVE\_OP\_DESTROY\_- OTF2\_RMA\_SYNC\_LEVEL\_PROCESS,  
HANDLE\_AND\_DEALLOCATE, 275  
273

OTF2\_COLLECTIVE\_OP\_EXSCAN, OTF2\_RMA\_SYNC\_TYPE\_MEMORY,  
273 275

OTF2\_COLLECTIVE\_OP\_GATHER, OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_-  
272 OUT, 275

OTF2\_COLLECTIVE\_OP\_GATHEROTF2\_FILEMODE\_MODIFY  
272 OTF2\_GeneralDefinitions.h, 456

OTF2\_COLLECTIVE\_OP\_REDUCEOTF2\_FILEMODE\_READ  
273 OTF2\_GeneralDefinitions.h, 456

OTF2\_COLLECTIVE\_OP\_REDUCEOTF2\_FILEMODE\_WRITE  
SCATTER, 273 OTF2\_GeneralDefinitions.h, 456

OTF2\_COLLECTIVE\_OP\_REDUCEOTF2\_FILETYPE\_ANCHOR  
SCATTER\_BLOCK, 273 OTF2\_GeneralDefinitions.h, 457

OTF2\_COLLECTIVE\_OP\_SCAN, OTF2\_FILETYPE\_EVENTS  
273 OTF2\_GeneralDefinitions.h, 457

OTF2\_COLLECTIVE\_OP\_SCATTEROTF2\_FILETYPE\_GLOBAL\_DEFS  
272 OTF2\_GeneralDefinitions.h, 457

OTF2\_COLLECTIVE\_OP\_SCATTEROTF2\_FILETYPE\_LOCAL\_DEFS  
272 OTF2\_GeneralDefinitions.h, 457

OTF2\_LOCK\_EXCLUSIVE, 273 OTF2\_FILETYPE\_MARKER  
OTF2\_LOCK\_SHARED, 273 OTF2\_GeneralDefinitions.h, 457

OTF2\_MEASUREMENT\_OFF, 274 OTF2\_FILETYPE\_SIONRANKMAP  
OTF2\_MEASUREMENT\_ON, 274 OTF2\_GeneralDefinitions.h, 457

OTF2\_RMA\_ATOMIC\_TYPE\_ACCUMULATEOTF2\_FILETYPE\_SNAPSHOTS  
274 OTF2\_GeneralDefinitions.h, 457

OTF2\_RMA\_ATOMIC\_TYPE\_COMPAREOTF2\_FILETYPE\_THUMBNAIL  
AND\_SWAP, 274 OTF2\_GeneralDefinitions.h, 457



OTF2\_FLUSH  
     OTF2\_GeneralDefinitions.h, 458  
 OTF2\_GeneralDefinitions.h  
     OTF2\_CALLBACK\_ERROR, 456  
     OTF2\_CALLBACK\_INTERRUPT, 456  
     OTF2\_CALLBACK\_SUCCESS, 456  
     OTF2\_COMPRESSION\_NONE, 456  
     OTF2\_COMPRESSION\_UNDEFINED, 456  
     OTF2\_COMPRESSION\_ZLIB, 456  
     OTF2\_FILEMODE\_MODIFY, 456  
     OTF2\_FILEMODE\_READ, 456  
     OTF2\_FILEMODE\_WRITE, 456  
     OTF2\_FILETYPE\_ANCHOR, 457  
     OTF2\_FILETYPE\_EVENTS, 457  
     OTF2\_FILETYPE\_GLOBAL\_DEFS, 457  
     OTF2\_FILETYPE\_LOCAL\_DEFS, 457  
     OTF2\_FILETYPE\_MARKER, 457  
     OTF2\_FILETYPE\_SIONRANKMAP, 457  
     OTF2\_FILETYPE\_SNAPSHOTS, 457  
     OTF2\_FILETYPE\_THUMBNAIL, 457  
     OTF2\_FLUSH, 458  
     OTF2\_MAPPING\_ATTRIBUTE, 458  
     OTF2\_MAPPING\_COMM, 458  
     OTF2\_MAPPING\_GROUP, 458  
     OTF2\_MAPPING\_LOCATION, 458  
     OTF2\_MAPPING\_MAX, 458  
     OTF2\_MAPPING\_METRIC, 458  
     OTF2\_MAPPING\_PARAMETER, 458  
     OTF2\_MAPPING\_REGION, 458  
     OTF2\_MAPPING\_RMA\_WIN, 458  
     OTF2\_MAPPING\_STRING, 458  
     OTF2\_NO\_FLUSH, 458  
     OTF2\_PARADIGM\_COMPILER, 459  
     OTF2\_PARADIGM\_CUDA, 459  
     OTF2\_PARADIGM\_GASPI, 459  
     OTF2\_PARADIGM\_HARDWARE, 459  
     OTF2\_PARADIGM\_HMPP, 459  
     OTF2\_PARADIGM\_MEASUREMENT\_SYSTEM, 459  
     OTF2\_PARADIGM\_MPI, 459  
     OTF2\_PARADIGM\_OMPSS, 459  
     OTF2\_PARADIGM\_OPENMP, 459  
     OTF2\_PARADIGM\_PTHREAD, 459  
     OTF2\_PARADIGM\_SHMEM, 459  
     OTF2\_PARADIGM\_UNKNOWN, 459  
     OTF2\_PARADIGM\_UPC, 459  
     OTF2\_PARADIGM\_USER, 459  
     OTF2\_SUBSTRATE\_NONE, 457  
     OTF2\_SUBSTRATE\_POSIX, 457  
     OTF2\_SUBSTRATE\_SION, 457  
     OTF2\_SUBSTRATE\_UNDEFINED, 457  
     OTF2\_THUMBNAIL\_TYPE\_ATTRIBUTES, 460  
     OTF2\_THUMBNAIL\_TYPE\_METRIC, 460  
     OTF2\_THUMBNAIL\_TYPE\_REGION, 460  
     OTF2\_TYPE\_ATTRIBUTE, 461  
     OTF2\_TYPE\_COMM, 461  
     OTF2\_TYPE\_DOUBLE, 460  
     OTF2\_TYPE\_FLOAT, 460  
     OTF2\_TYPE\_GROUP, 461  
     OTF2\_TYPE\_INT16, 460  
     OTF2\_TYPE\_INT32, 460  
     OTF2\_TYPE\_INT64, 460  
     OTF2\_TYPE\_INT8, 460  
     OTF2\_TYPE\_LOCATION, 461  
     OTF2\_TYPE\_METRIC, 461  
     OTF2\_TYPE\_NONE, 460  
     OTF2\_TYPE\_PARAMETER, 461  
     OTF2\_TYPE\_REGION, 461  
     OTF2\_TYPE\_RMA\_WIN, 461  
     OTF2\_TYPE\_STRING, 460  
     OTF2\_TYPE\_UINT16, 460  
     OTF2\_TYPE\_UINT32, 460  
     OTF2\_TYPE\_UINT64, 460  
     OTF2\_TYPE\_UINT8, 460  
     OTF2\_GROUP\_FLAG\_GLOBAL\_MEMBERS  
         OTF2\_Definitions.h, 201  
     OTF2\_GROUP\_FLAG\_NONE



## INDEX

---

OTF2\_Definitions.h, [201](#)  
OTF2\_GROUP\_TYPE\_COMM\_GROUP  
OTF2\_Definitions.h, [201](#)  
OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS  
OTF2\_Definitions.h, [201](#)  
OTF2\_GROUP\_TYPE\_COMM\_SELF  
OTF2\_Definitions.h, [202](#)  
OTF2\_GROUP\_TYPE\_LOCATIONS  
OTF2\_Definitions.h, [201](#)  
OTF2\_GROUP\_TYPE\_METRIC  
OTF2\_Definitions.h, [201](#)  
OTF2\_GROUP\_TYPE\_REGIONS  
OTF2\_Definitions.h, [201](#)  
OTF2\_GROUP\_TYPE\_UNKNOWN  
OTF2\_Definitions.h, [201](#)  
OTF2\_ID\_MAP\_DENSE  
OTF2\_IdMap.h, [651](#)  
OTF2\_ID\_MAP\_SPARSE  
OTF2\_IdMap.h, [651](#)  
OTF2\_IdMap.h  
OTF2\_ID\_MAP\_DENSE, [651](#)  
OTF2\_ID\_MAP\_SPARSE, [651](#)  
OTF2\_LOCATION\_GROUP\_TYPE\_PROCESS  
OTF2\_Definitions.h, [202](#)  
OTF2\_LOCATION\_GROUP\_TYPE\_UNKNOWN  
OTF2\_Definitions.h, [202](#)  
OTF2\_LOCATION\_TYPE\_CPU\_THREAD  
OTF2\_Definitions.h, [202](#)  
OTF2\_LOCATION\_TYPE\_GPU  
OTF2\_Definitions.h, [202](#)  
OTF2\_LOCATION\_TYPE\_METRIC  
OTF2\_Definitions.h, [202](#)  
OTF2\_LOCATION\_TYPE\_UNKNOWN  
OTF2\_Definitions.h, [202](#)  
OTF2\_LOCK\_EXCLUSIVE  
OTF2\_Events.h, [273](#)  
OTF2\_LOCK\_SHARED  
OTF2\_Events.h, [273](#)  
OTF2\_MAPPING\_ATTRIBUTE  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_COMM  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_GROUP  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_LOCATION  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_MAX  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_METRIC  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_PARAMETER  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_REGION  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_RMA\_WIN  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_MAPPING\_STRING  
OTF2\_GeneralDefinitions.h, [458](#)  
OTF2\_Marker.h  
OTF2\_MARKER\_SCOPE\_COMM,  
[657](#)  
OTF2\_MARKER\_SCOPE\_GLOBAL,  
[656](#)  
OTF2\_MARKER\_SCOPE\_GROUP,  
[657](#)  
OTF2\_MARKER\_SCOPE\_LOCATION,  
[656](#)  
OTF2\_MARKER\_SCOPE\_LOCATION\_-  
GROUP, [656](#)  
OTF2\_MARKER\_SCOPE\_SYSTEM\_-  
TREE\_NODE, [656](#)  
OTF2\_SEVERITY\_HIGH, [657](#)  
OTF2\_SEVERITY\_LOW, [657](#)  
OTF2\_SEVERITY\_MEDIUM, [657](#)  
OTF2\_SEVERITY\_NONE, [657](#)  
OTF2\_MARKER\_SCOPE\_COMM  
OTF2\_Marker.h, [657](#)  
OTF2\_MARKER\_SCOPE\_GLOBAL  
OTF2\_Marker.h, [656](#)  
OTF2\_MARKER\_SCOPE\_GROUP  
OTF2\_Marker.h, [657](#)  
OTF2\_MARKER\_SCOPE\_LOCATION  
OTF2\_Marker.h, [656](#)  
OTF2\_MARKER\_SCOPE\_LOCATION\_-  
GROUP  
OTF2\_Marker.h, [656](#)  
OTF2\_MARKER\_SCOPE\_SYSTEM\_-  
TREE\_NODE

OTF2_Marker.h, <a href="#">656</a>	OTF2_Definitions.h, <a href="#">205</a>
OTF2_MEASUREMENT_OFF	OTF2_METRIC_TYPE_RUSAGE
OTF2_Events.h, <a href="#">274</a>	OTF2_Definitions.h, <a href="#">205</a>
OTF2_MEASUREMENT_ON	OTF2_METRIC_TYPE_USER
OTF2_Events.h, <a href="#">274</a>	OTF2_Definitions.h, <a href="#">205</a>
OTF2_METRIC_ABSOLUTE_LAST	OTF2_METRIC_VALUE_ABSOLUTE
OTF2_Definitions.h, <a href="#">203</a>	OTF2_Definitions.h, <a href="#">206</a>
OTF2_METRIC_ABSOLUTE_NEXT	OTF2_METRIC_VALUE_ACCUMULATED
OTF2_Definitions.h, <a href="#">203</a>	OTF2_Definitions.h, <a href="#">206</a>
OTF2_METRIC_ABSOLUTE_POINT	OTF2_METRIC_VALUE_MASK
OTF2_Definitions.h, <a href="#">203</a>	OTF2_Definitions.h, <a href="#">206</a>
OTF2_METRIC_ACCUMULATED_LAST	OTF2_METRIC_VALUE_RELATIVE
OTF2_Definitions.h, <a href="#">203</a>	OTF2_Definitions.h, <a href="#">206</a>
OTF2_METRIC_ACCUMULATED_NEXT	OTF2_NO_FLUSH
OTF2_Definitions.h, <a href="#">203</a>	OTF2_GeneralDefinitions.h, <a href="#">458</a>
OTF2_METRIC_ACCUMULATED_POINT	OTF2_PARADIGM_COMPILER
OTF2_Definitions.h, <a href="#">203</a>	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_METRIC_ACCUMULATED_START	OTF2_PARADIGM_CUDA
OTF2_Definitions.h, <a href="#">203</a>	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_METRIC_ASYNCHRONOUS	OTF2_PARADIGM_GASPI
OTF2_Definitions.h, <a href="#">204</a>	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_METRIC_RELATIVE_LAST	OTF2_PARADIGM_HARDWARE
OTF2_Definitions.h, <a href="#">203</a>	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_METRIC_RELATIVE_NEXT	OTF2_PARADIGM_HMPP
OTF2_Definitions.h, <a href="#">203</a>	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_METRIC_RELATIVE_POINT	OTF2_PARADIGM_MEASUREMENT_ -
OTF2_Definitions.h, <a href="#">203</a>	SYSTEM
OTF2_METRIC_SYNCHRONOUS	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_Definitions.h, <a href="#">204</a>	OTF2_PARADIGM_MPI
OTF2_METRIC_SYNCHRONOUS_STRICT	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_Definitions.h, <a href="#">204</a>	OTF2_PARADIGM_OMPSS
OTF2_METRIC_TIMING_LAST	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_Definitions.h, <a href="#">205</a>	OTF2_PARADIGM_OPENMP
OTF2_METRIC_TIMING_MASK	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_Definitions.h, <a href="#">205</a>	OTF2_PARADIGM_PTHREAD
OTF2_METRIC_TIMING_NEXT	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_Definitions.h, <a href="#">205</a>	OTF2_PARADIGM_SHMEM
OTF2_METRIC_TIMING_POINT	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_Definitions.h, <a href="#">205</a>	OTF2_PARADIGM_UNKNOWN
OTF2_METRIC_TIMING_START	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_Definitions.h, <a href="#">205</a>	OTF2_PARADIGM_UPC
OTF2_METRIC_TYPE_OTHER	OTF2_GeneralDefinitions.h, <a href="#">459</a>
OTF2_Definitions.h, <a href="#">205</a>	OTF2_PARADIGM_USER
OTF2_METRIC_TYPE_PAPI	OTF2_GeneralDefinitions.h, <a href="#">459</a>

## INDEX

---

OTF2_PARAMETER_TYPE_INT64	OTF2_REGION_ROLE_FLUSH
OTF2_Definitions.h, <a href="#">206</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_PARAMETER_TYPE_STRING	OTF2_REGION_ROLE_FUNCTION
OTF2_Definitions.h, <a href="#">206</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_PARAMETER_TYPE_UINT64	OTF2_REGION_ROLE_IMPLICIT_BARRIER
OTF2_Definitions.h, <a href="#">206</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_RECORDER_KIND_ABSTRACT	OTF2_REGION_ROLE_LOOP
OTF2_Definitions.h, <a href="#">206</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_RECORDER_KIND_CPU	OTF2_REGION_ROLE_MASTER
OTF2_Definitions.h, <a href="#">206</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_RECORDER_KIND_GPU	OTF2_REGION_ROLE_ORDERED
OTF2_Definitions.h, <a href="#">206</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_RECORDER_KIND_UNKNOWN	OTF2_REGION_ROLE_ORDERED_SBLOCK
OTF2_Definitions.h, <a href="#">206</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_REGION_FLAG_DYNAMIC	OTF2_REGION_ROLE_PARALLEL
OTF2_Definitions.h, <a href="#">207</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_REGION_FLAG_NONE	OTF2_REGION_ROLE_POINT2POINT
OTF2_Definitions.h, <a href="#">207</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_REGION_FLAG_PHASE	OTF2_REGION_ROLE_RMA
OTF2_Definitions.h, <a href="#">207</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_REGION_ROLE_ARTIFICIAL	OTF2_REGION_ROLE_SECTION
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_REGION_ROLE_ATOMIC	OTF2_REGION_ROLE_SECTIONS
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_REGION_ROLE_BARRIER	OTF2_REGION_ROLE_SINGLE
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_REGION_ROLE_CODE	OTF2_REGION_ROLE_SINGLE_SBLOCK
OTF2_Definitions.h, <a href="#">207</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_REGION_ROLE_COLL_ALL2ALL	OTF2_REGION_ROLE_TASK
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_REGION_ROLE_COLL_ALL2ONE	OTF2_REGION_ROLE_TASK_CREATE
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_REGION_ROLE_COLL_ONE2ALL	OTF2_REGION_ROLE_TASK_WAIT
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_REGION_ROLE_COLL_OTHER	OTF2_REGION_ROLE_THREAD_CREATE
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">208</a>
OTF2_REGION_ROLE_CRITICAL	OTF2_REGION_ROLE_THREAD_WAIT
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">209</a>
OTF2_REGION_ROLE_CRITICAL_SBLOCK	OTF2_REGION_ROLE_UNKNOWN
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_REGION_ROLE_DATA_TRANSFER	OTF2_REGION_ROLE_WORKSHARE
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">207</a>
OTF2_REGION_ROLE_FILE_IO	OTF2_REGION_ROLE_WRAPPER
OTF2_Definitions.h, <a href="#">208</a>	OTF2_Definitions.h, <a href="#">207</a>

OTF2\_RMA\_ATOMIC\_TYPE\_ACCUMULATE\_OTF2\_SEVERITY\_MEDIUM  
     OTF2\_Events.h, [274](#)                      OTF2\_Marker.h, [657](#)  
 OTF2\_RMA\_ATOMIC\_TYPE\_COMPARE\_OTF2\_SEVERITY\_NONE  
     AND\_SWAP                      OTF2\_Marker.h, [657](#)  
     OTF2\_Events.h, [274](#)                      OTF2\_SUBSTRATE\_NONE  
 OTF2\_RMA\_ATOMIC\_TYPE\_FETCH\_-      OTF2\_GeneralDefinitions.h, [457](#)  
     AND\_ADD                      OTF2\_SUBSTRATE\_POSIX  
     OTF2\_Events.h, [274](#)                      OTF2\_GeneralDefinitions.h, [457](#)  
 OTF2\_RMA\_ATOMIC\_TYPE\_FETCH\_-OTF2\_SUBSTRATE\_SION  
     AND\_INCREMENT                      OTF2\_GeneralDefinitions.h, [457](#)  
     OTF2\_Events.h, [274](#)                      OTF2\_SUBSTRATE\_UNDEFINED  
 OTF2\_RMA\_ATOMIC\_TYPE\_INCREMENTOTF2\_GeneralDefinitions.h, [457](#)  
     OTF2\_Events.h, [274](#)                      OTF2\_SUCCESS  
 OTF2\_RMA\_ATOMIC\_TYPE\_SWAP              OTF2\_ErrorCodes.h, [132](#)  
     OTF2\_Events.h, [274](#)                      OTF2\_SYSTEM\_TREE\_DOMAIN\_CACHE  
 OTF2\_RMA\_ATOMIC\_TYPE\_TEST\_-              OTF2\_Definitions.h, [209](#)  
     AND\_SET                      OTF2\_SYSTEM\_TREE\_DOMAIN\_CORE  
     OTF2\_Events.h, [274](#)                      OTF2\_Definitions.h, [209](#)  
 OTF2\_RMA\_SYNC\_LEVEL\_MEMORYOTF2\_SYSTEM\_TREE\_DOMAIN\_MACHINE  
     OTF2\_Events.h, [275](#)                      OTF2\_Definitions.h, [209](#)  
 OTF2\_RMA\_SYNC\_LEVEL\_NONE      OTF2\_SYSTEM\_TREE\_DOMAIN\_NUMA  
     OTF2\_Events.h, [275](#)                      OTF2\_Definitions.h, [209](#)  
 OTF2\_RMA\_SYNC\_LEVEL\_PROCESS OTF2\_SYSTEM\_TREE\_DOMAIN\_PU  
     OTF2\_Events.h, [275](#)                      OTF2\_Definitions.h, [209](#)  
 OTF2\_RMA\_SYNC\_TYPE\_MEMORY OTF2\_SYSTEM\_TREE\_DOMAIN\_SHARED\_-  
     OTF2\_Events.h, [275](#)                      MEMORY  
 OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_-      OTF2\_Definitions.h, [209](#)  
     IN                      OTF2\_SYSTEM\_TREE\_DOMAIN\_SOCKET  
     OTF2\_Events.h, [275](#)                      OTF2\_Definitions.h, [209](#)  
 OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_- OTF2\_THUMBNAIL\_TYPE\_ATTRIBUTES  
     OUT                      OTF2\_GeneralDefinitions.h, [460](#)  
     OTF2\_Events.h, [275](#)                      OTF2\_THUMBNAIL\_TYPE\_METRIC  
 OTF2\_SCOPE\_GROUP                      OTF2\_GeneralDefinitions.h, [460](#)  
     OTF2\_Definitions.h, [204](#)                      OTF2\_THUMBNAIL\_TYPE\_REGION  
 OTF2\_SCOPE\_LOCATION                      OTF2\_GeneralDefinitions.h, [460](#)  
     OTF2\_Definitions.h, [204](#)                      OTF2\_TYPE\_ATTRIBUTE  
 OTF2\_SCOPE\_LOCATION\_GROUP              OTF2\_GeneralDefinitions.h, [461](#)  
     OTF2\_Definitions.h, [204](#)                      OTF2\_TYPE\_COMM  
 OTF2\_SCOPE\_SYSTEM\_TREE\_NODE      OTF2\_GeneralDefinitions.h, [461](#)  
     OTF2\_Definitions.h, [204](#)                      OTF2\_TYPE\_DOUBLE  
 OTF2\_SEVERITY\_HIGH                      OTF2\_GeneralDefinitions.h, [460](#)  
     OTF2\_Marker.h, [657](#)                      OTF2\_TYPE\_FLOAT  
 OTF2\_SEVERITY\_LOW                      OTF2\_GeneralDefinitions.h, [460](#)  
     OTF2\_Marker.h, [657](#)                      OTF2\_TYPE\_GROUP

## INDEX

---

OTF2\_GeneralDefinitions.h, [461](#)  
OTF2\_TYPE\_INT16  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_INT32  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_INT64  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_INT8  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_LOCATION  
    OTF2\_GeneralDefinitions.h, [461](#)  
OTF2\_TYPE\_METRIC  
    OTF2\_GeneralDefinitions.h, [461](#)  
OTF2\_TYPE\_NONE  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_PARAMETER  
    OTF2\_GeneralDefinitions.h, [461](#)  
OTF2\_TYPE\_REGION  
    OTF2\_GeneralDefinitions.h, [461](#)  
OTF2\_TYPE\_RMA\_WIN  
    OTF2\_GeneralDefinitions.h, [461](#)  
OTF2\_TYPE\_STRING  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_UINT16  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_UINT32  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_UINT64  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_TYPE\_UINT8  
    OTF2\_GeneralDefinitions.h, [460](#)  
OTF2\_WARNING  
    OTF2\_ErrorCodes.h, [132](#)  
OTF2\_Archive  
    OTF2\_Archive.h, [142](#)  
OTF2\_Archive.h  
    OTF2\_Archive, [142](#)  
    OTF2\_Archive\_Close, [143](#)  
    OTF2\_Archive\_CloseDefFiles, [143](#)  
    OTF2\_Archive\_CloseDefReader, [143](#)  
    OTF2\_Archive\_CloseDefWriter, [144](#)  
    OTF2\_Archive\_CloseEvtFiles, [144](#)  
    OTF2\_Archive\_CloseEvtReader, [144](#)  
    OTF2\_Archive\_CloseEvtWriter, [145](#)  
    OTF2\_Archive\_CloseGlobalDefReader, [145](#)  
    OTF2\_Archive\_CloseGlobalDefWriter, [145](#)  
    OTF2\_Archive\_CloseGlobalEvtReader, [146](#)  
    OTF2\_Archive\_CloseGlobalSnapReader, [146](#)  
    OTF2\_Archive\_CloseMarkerReader, [146](#)  
    OTF2\_Archive\_CloseMarkerWriter, [147](#)  
    OTF2\_Archive\_CloseSnapFiles, [147](#)  
    OTF2\_Archive\_CloseSnapReader, [148](#)  
    OTF2\_Archive\_CloseSnapWriter, [148](#)  
    OTF2\_Archive\_CloseThumbReader, [149](#)  
    OTF2\_Archive\_GetChunkSize, [149](#)  
    OTF2\_Archive\_GetCompression, [149](#)  
    OTF2\_Archive\_GetCreator, [150](#)  
    OTF2\_Archive\_GetDefReader, [150](#)  
    OTF2\_Archive\_GetDefWriter, [150](#)  
    OTF2\_Archive\_GetDescription, [151](#)  
    OTF2\_Archive\_GetEvtReader, [151](#)  
    OTF2\_Archive\_GetEvtWriter, [151](#)  
    OTF2\_Archive\_GetFileSubstrate, [152](#)  
    OTF2\_Archive\_GetGlobalDefReader, [152](#)  
    OTF2\_Archive\_GetGlobalDefWriter, [152](#)  
    OTF2\_Archive\_GetGlobalEvtReader, [153](#)  
    OTF2\_Archive\_GetGlobalSnapReader, [153](#)  
    OTF2\_Archive\_GetMachineName, [153](#)  
    OTF2\_Archive\_GetMarkerReader, [154](#)  
    OTF2\_Archive\_GetMarkerWriter, [154](#)  
    OTF2\_Archive\_GetNumberOfGlobalDefinitions, [154](#)  
    OTF2\_Archive\_GetNumberOfLocations, [155](#)  
    OTF2\_Archive\_GetNumberOfSnapshots, [155](#)

OTF2\_Archive\_GetNumberOfThumbFiles  
 156  
 OTF2\_Archive\_GetProperty, 156  
 OTF2\_Archive\_GetPropertyNames,  
 156  
 OTF2\_Archive\_GetSnapReader, 157  
 OTF2\_Archive\_GetSnapWriter, 157  
 OTF2\_Archive\_GetThumbReader, 158  
 OTF2\_Archive\_GetThumbWriter, 158  
 OTF2\_Archive\_GetTraceId, 159  
 OTF2\_Archive\_GetVersion, 159  
 OTF2\_Archive\_Open, 159  
 OTF2\_Archive\_OpenDefFiles, 161  
 OTF2\_Archive\_OpenEvtFiles, 161  
 OTF2\_Archive\_OpenSnapFiles, 161  
 OTF2\_Archive\_SelectLocation, 162  
 OTF2\_Archive\_SetBoolProperty, 162  
 OTF2\_Archive\_SetCollectiveCallbacks,  
 163  
 OTF2\_Archive\_SetCreator, 163  
 OTF2\_Archive\_SetDescription, 164  
 OTF2\_Archive\_SetFlushCallbacks,  
 164  
 OTF2\_Archive\_SetMachineName, 164  
 OTF2\_Archive\_SetMemoryCallbacks,  
 165  
 OTF2\_Archive\_SetNumberOfSnapshots,  
 165  
 OTF2\_Archive\_SetProperty, 166  
 OTF2\_Archive\_SetSerialCollectiveCallbacks,  
 166  
 OTF2\_Archive\_SwitchFileMode, 167  
 OTF2\_CHUNK\_SIZE\_DEFINITION  
 DEFAULT, 142  
 OTF2\_CHUNK\_SIZE\_EVENTS\_-  
 DEFAULT, 142  
 OTF2\_Archive\_Close  
 OTF2\_Archive.h, 143  
 OTF2\_Archive\_CloseDefFiles  
 OTF2\_Archive.h, 143  
 OTF2\_Archive\_CloseDefReader  
 OTF2\_Archive.h, 143  
 OTF2\_Archive\_CloseDefWriter  
 OTF2\_Archive.h, 144  
 OTF2\_Archive\_CloseEvtFiles  
 OTF2\_Archive.h, 144  
 OTF2\_Archive\_CloseEvtReader  
 OTF2\_Archive.h, 144  
 OTF2\_Archive\_CloseEvtWriter  
 OTF2\_Archive.h, 145  
 OTF2\_Archive\_CloseGlobalDefReader  
 OTF2\_Archive.h, 145  
 OTF2\_Archive\_CloseGlobalDefWriter  
 OTF2\_Archive.h, 145  
 OTF2\_Archive\_CloseGlobalEvtReader  
 OTF2\_Archive.h, 146  
 OTF2\_Archive\_CloseGlobalSnapReader  
 OTF2\_Archive.h, 146  
 OTF2\_Archive\_CloseMarkerReader  
 OTF2\_Archive.h, 146  
 OTF2\_Archive\_CloseMarkerWriter  
 OTF2\_Archive.h, 147  
 OTF2\_Archive\_CloseSnapFiles  
 OTF2\_Archive.h, 147  
 OTF2\_Archive\_CloseSnapReader  
 OTF2\_Archive.h, 148  
 OTF2\_Archive\_CloseSnapWriter  
 OTF2\_Archive.h, 148  
 OTF2\_Archive\_CloseThumbReader  
 OTF2\_Archive.h, 149  
 OTF2\_Archive\_GetChunkSize  
 OTF2\_Archive.h, 149  
 OTF2\_Archive\_GetCompression  
 OTF2\_Archive.h, 149  
 OTF2\_Archive\_GetCreator  
 OTF2\_Archive.h, 150  
 OTF2\_Archive\_GetDefReader  
 OTF2\_Archive.h, 150  
 OTF2\_Archive\_GetDefWriter  
 OTF2\_Archive.h, 150  
 OTF2\_Archive\_GetDescription  
 OTF2\_Archive.h, 151  
 OTF2\_Archive\_GetEvtReader  
 OTF2\_Archive.h, 151  
 OTF2\_Archive\_GetEvtWriter  
 OTF2\_Archive.h, 151  
 OTF2\_Archive\_GetFileSubstrate  
 OTF2\_Archive.h, 152



## INDEX

---

- OTF2\_Archive\_GetGlobalDefReader  
OTF2\_Archive.h, [152](#)
- OTF2\_Archive\_GetGlobalDefWriter  
OTF2\_Archive.h, [152](#)
- OTF2\_Archive\_GetGlobalEvtReader  
OTF2\_Archive.h, [153](#)
- OTF2\_Archive\_GetGlobalSnapReader  
OTF2\_Archive.h, [153](#)
- OTF2\_Archive\_GetMachineName  
OTF2\_Archive.h, [153](#)
- OTF2\_Archive\_GetMarkerReader  
OTF2\_Archive.h, [154](#)
- OTF2\_Archive\_GetMarkerWriter  
OTF2\_Archive.h, [154](#)
- OTF2\_Archive\_GetNumberOfGlobalDefinitions  
OTF2\_Archive.h, [154](#)
- OTF2\_Archive\_GetNumberOfLocations  
OTF2\_Archive.h, [155](#)
- OTF2\_Archive\_GetNumberOfSnapshots  
OTF2\_Archive.h, [155](#)
- OTF2\_Archive\_GetNumberOfThumbnails  
OTF2\_Archive.h, [156](#)
- OTF2\_Archive\_GetProperty  
OTF2\_Archive.h, [156](#)
- OTF2\_Archive\_GetPropertyNames  
OTF2\_Archive.h, [156](#)
- OTF2\_Archive\_GetSnapReader  
OTF2\_Archive.h, [157](#)
- OTF2\_Archive\_GetSnapWriter  
OTF2\_Archive.h, [157](#)
- OTF2\_Archive\_GetThumbReader  
OTF2\_Archive.h, [158](#)
- OTF2\_Archive\_GetThumbWriter  
OTF2\_Archive.h, [158](#)
- OTF2\_Archive\_GetTraceId  
OTF2\_Archive.h, [159](#)
- OTF2\_Archive\_GetVersion  
OTF2\_Archive.h, [159](#)
- OTF2\_Archive\_Open  
OTF2\_Archive.h, [159](#)
- OTF2\_Archive\_OpenDefFiles  
OTF2\_Archive.h, [161](#)
- OTF2\_Archive\_OpenEvtFiles  
OTF2\_Archive.h, [161](#)
- OTF2\_Archive\_OpenSnapFiles  
OTF2\_Archive.h, [161](#)
- OTF2\_Archive\_SelectLocation  
OTF2\_Archive.h, [162](#)
- OTF2\_Archive\_SetBoolProperty  
OTF2\_Archive.h, [162](#)
- OTF2\_Archive\_SetCollectiveCallbacks  
OTF2\_Archive.h, [163](#)
- OTF2\_Archive\_SetCreator  
OTF2\_Archive.h, [163](#)
- OTF2\_Archive\_SetDescription  
OTF2\_Archive.h, [164](#)
- OTF2\_Archive\_SetFlushCallbacks  
OTF2\_Archive.h, [164](#)
- OTF2\_Archive\_SetMachineName  
OTF2\_Archive.h, [164](#)
- OTF2\_Archive\_SetMemoryCallbacks  
OTF2\_Archive.h, [165](#)
- OTF2\_Archive\_SetNumberOfSnapshots  
OTF2\_Archive.h, [165](#)
- OTF2\_Archive\_SetProperty  
OTF2\_Archive.h, [166](#)
- OTF2\_Archive\_SetSerialCollectiveCallbacks  
OTF2\_Archive.h, [166](#)
- OTF2\_Archive\_SwitchFileMode  
OTF2\_Archive.h, [167](#)
- OTF2\_AttributeList.h  
OTF2\_AttributeList\_AddAttribute, [172](#)  
OTF2\_AttributeList\_AddAttributeRef,  
[173](#)  
OTF2\_AttributeList\_AddCommRef,  
[173](#)  
OTF2\_AttributeList\_AddDouble, [174](#)  
OTF2\_AttributeList\_AddFloat, [174](#)  
OTF2\_AttributeList\_AddGroupRef,  
[174](#)  
OTF2\_AttributeList\_AddInt16, [175](#)  
OTF2\_AttributeList\_AddInt32, [175](#)  
OTF2\_AttributeList\_AddInt64, [176](#)  
OTF2\_AttributeList\_AddInt8, [176](#)  
OTF2\_AttributeList\_AddLocationRef,  
[176](#)  
OTF2\_AttributeList\_AddMetricRef,  
[177](#)

OTF2\_AttributeList\_AddParameterRef, 177  
 OTF2\_AttributeList\_AddRegionRef, 177  
 OTF2\_AttributeList\_AddRmaWinRef, 178  
 OTF2\_AttributeList\_AddString, 178  
 OTF2\_AttributeList\_AddStringRef, 179  
 OTF2\_AttributeList\_AddUInt16, 179  
 OTF2\_AttributeList\_AddUInt32, 180  
 OTF2\_AttributeList\_AddUInt64, 180  
 OTF2\_AttributeList\_AddUInt8, 180  
 OTF2\_AttributeList\_Delete, 181  
 OTF2\_AttributeList\_GetAttributeByID, 181  
 OTF2\_AttributeList\_GetAttributeByIndex, 181  
 OTF2\_AttributeList\_GetAttributeRef, 182  
 OTF2\_AttributeList\_GetCommRef, 182  
 OTF2\_AttributeList\_GetDouble, 183  
 OTF2\_AttributeList\_GetFloat, 183  
 OTF2\_AttributeList\_GetGroupRef, 183  
 OTF2\_AttributeList\_GetInt16, 184  
 OTF2\_AttributeList\_GetInt32, 184  
 OTF2\_AttributeList\_GetInt64, 185  
 OTF2\_AttributeList\_GetInt8, 185  
 OTF2\_AttributeList\_GetLocationRef, 185  
 OTF2\_AttributeList\_GetMetricRef, 186  
 OTF2\_AttributeList\_GetNumberOfElements, 186  
 OTF2\_AttributeList\_GetParameterRef, 187  
 OTF2\_AttributeList\_GetRegionRef, 187  
 OTF2\_AttributeList\_GetRmaWinRef, 187  
 OTF2\_AttributeList\_GetString, 188  
 OTF2\_AttributeList\_GetStringRef, 188  
 OTF2\_AttributeList\_GetUInt16, 189  
 OTF2\_AttributeList\_GetUInt32, 189  
 OTF2\_AttributeList\_GetUInt64, 190  
 OTF2\_AttributeList\_GetUInt8, 190  
 OTF2\_AttributeList\_New, 191  
 OTF2\_AttributeList\_PopAttribute, 191  
 OTF2\_AttributeList\_RemoveAllAttributes, 191  
 OTF2\_AttributeList\_RemoveAttribute, 191  
 OTF2\_AttributeList\_TestAttributeByID, 192  
 OTF2\_AttributeList\_AddAttribute  
 OTF2\_AttributeList.h, 172  
 OTF2\_AttributeList\_AddAttributeRef  
 OTF2\_AttributeList.h, 173  
 OTF2\_AttributeList\_AddCommRef  
 OTF2\_AttributeList.h, 173  
 OTF2\_AttributeList\_AddDouble  
 OTF2\_AttributeList.h, 174  
 OTF2\_AttributeList\_AddFloat  
 OTF2\_AttributeList.h, 174  
 OTF2\_AttributeList\_AddGroupRef  
 OTF2\_AttributeList.h, 174  
 OTF2\_AttributeList\_AddInt16  
 OTF2\_AttributeList.h, 175  
 OTF2\_AttributeList\_AddInt32  
 OTF2\_AttributeList.h, 175  
 OTF2\_AttributeList\_AddInt64  
 OTF2\_AttributeList.h, 176  
 OTF2\_AttributeList\_AddInt8  
 OTF2\_AttributeList.h, 176  
 OTF2\_AttributeList\_AddLocationRef  
 OTF2\_AttributeList.h, 176  
 OTF2\_AttributeList\_AddMetricRef  
 OTF2\_AttributeList.h, 177  
 OTF2\_AttributeList\_AddParameterRef  
 OTF2\_AttributeList.h, 177  
 OTF2\_AttributeList\_AddRegionRef  
 OTF2\_AttributeList.h, 177  
 OTF2\_AttributeList\_AddRmaWinRef  
 OTF2\_AttributeList.h, 178  
 OTF2\_AttributeList\_AddString  
 OTF2\_AttributeList.h, 178



## INDEX

---

- OTF2\_AttributeList\_AddStringRef  
OTF2\_AttributeList.h, [179](#)
- OTF2\_AttributeList\_AddUint16  
OTF2\_AttributeList.h, [179](#)
- OTF2\_AttributeList\_AddUint32  
OTF2\_AttributeList.h, [180](#)
- OTF2\_AttributeList\_AddUint64  
OTF2\_AttributeList.h, [180](#)
- OTF2\_AttributeList\_AddUint8  
OTF2\_AttributeList.h, [180](#)
- OTF2\_AttributeList\_Delete  
OTF2\_AttributeList.h, [181](#)
- OTF2\_AttributeList\_GetAttributeByID  
OTF2\_AttributeList.h, [181](#)
- OTF2\_AttributeList\_GetAttributeByIndex  
OTF2\_AttributeList.h, [181](#)
- OTF2\_AttributeList\_GetAttributeRef  
OTF2\_AttributeList.h, [182](#)
- OTF2\_AttributeList\_GetCommRef  
OTF2\_AttributeList.h, [182](#)
- OTF2\_AttributeList\_GetDouble  
OTF2\_AttributeList.h, [183](#)
- OTF2\_AttributeList\_GetFloat  
OTF2\_AttributeList.h, [183](#)
- OTF2\_AttributeList\_GetGroupRef  
OTF2\_AttributeList.h, [183](#)
- OTF2\_AttributeList\_GetInt16  
OTF2\_AttributeList.h, [184](#)
- OTF2\_AttributeList\_GetInt32  
OTF2\_AttributeList.h, [184](#)
- OTF2\_AttributeList\_GetInt64  
OTF2\_AttributeList.h, [185](#)
- OTF2\_AttributeList\_GetInt8  
OTF2\_AttributeList.h, [185](#)
- OTF2\_AttributeList\_GetLocationRef  
OTF2\_AttributeList.h, [185](#)
- OTF2\_AttributeList\_GetMetricRef  
OTF2\_AttributeList.h, [186](#)
- OTF2\_AttributeList\_GetNumberOfElements  
OTF2\_AttributeList.h, [186](#)
- OTF2\_AttributeList\_GetParameterRef  
OTF2\_AttributeList.h, [187](#)
- OTF2\_AttributeList\_GetRegionRef  
OTF2\_AttributeList.h, [187](#)
- OTF2\_AttributeList\_GetRmaWinRef  
OTF2\_AttributeList.h, [187](#)
- OTF2\_AttributeList\_GetString  
OTF2\_AttributeList.h, [188](#)
- OTF2\_AttributeList\_GetStringRef  
OTF2\_AttributeList.h, [188](#)
- OTF2\_AttributeList\_GetUint16  
OTF2\_AttributeList.h, [189](#)
- OTF2\_AttributeList\_GetUint32  
OTF2\_AttributeList.h, [189](#)
- OTF2\_AttributeList\_GetUint64  
OTF2\_AttributeList.h, [190](#)
- OTF2\_AttributeList\_GetUint8  
OTF2\_AttributeList.h, [190](#)
- OTF2\_AttributeList\_New  
OTF2\_AttributeList.h, [191](#)
- OTF2\_AttributeList\_PopAttribute  
OTF2\_AttributeList.h, [191](#)
- OTF2\_AttributeList\_RemoveAllAttributes  
OTF2\_AttributeList.h, [191](#)
- OTF2\_AttributeList\_RemoveAttribute  
OTF2\_AttributeList.h, [191](#)
- OTF2\_AttributeList\_TestAttributeByID  
OTF2\_AttributeList.h, [192](#)
- OTF2\_AttributeValue, [121](#)
- OTF2\_CallbackCode  
OTF2\_GeneralDefinitions.h, [455](#)
- OTF2\_CartPeriodicity\_enum  
OTF2\_Definitions.h, [200](#)
- OTF2\_CHUNK\_SIZE\_DEFINITIONS\_  
DEFAULT  
OTF2\_Archive.h, [142](#)
- OTF2\_CHUNK\_SIZE\_EVENTS\_DEFAULT  
OTF2\_Archive.h, [142](#)
- OTF2\_CollectiveCallbacks, [123](#)
- OTF2\_CollectiveContext, [123](#)
- OTF2\_CollectiveOp\_enum  
OTF2\_Events.h, [272](#)
- OTF2\_Collectives\_Barrier  
Operating OTF2 in an collective con-  
text, [97](#)
- OTF2\_Collectives\_Bcast  
Operating OTF2 in an collective con-  
text, [98](#)

- OTF2\_Collectives\_CreateLocalComm
  - Operating OTF2 in an collective context, [98](#)
- OTF2\_Collectives\_FreeLocalComm
  - Operating OTF2 in an collective context, [98](#)
- OTF2\_Collectives\_Gather
  - Operating OTF2 in an collective context, [99](#)
- OTF2\_Collectives\_Gatherv
  - Operating OTF2 in an collective context, [99](#)
- OTF2\_Collectives\_GetRank
  - Operating OTF2 in an collective context, [99](#)
- OTF2\_Collectives\_GetSize
  - Operating OTF2 in an collective context, [100](#)
- OTF2\_Collectives\_Release
  - Operating OTF2 in an collective context, [100](#)
- OTF2\_Collectives\_Scatter
  - Operating OTF2 in an collective context, [100](#)
- OTF2\_Collectives\_Scatterv
  - Operating OTF2 in an collective context, [101](#)
- OTF2\_Compression\_enum
  - OTF2\_GeneralDefinitions.h, [456](#)
- OTF2\_Definitions.h
  - OTF2\_CartPeriodicity\_enum, [200](#)
  - OTF2\_GroupFlag\_enum, [200](#)
  - OTF2\_GroupType\_enum, [201](#)
  - OTF2\_LocationGroupType\_enum, [200](#)
  - OTF2\_LocationType\_enum, [202](#)
  - OTF2\_MetricBase\_enum, [202](#)
  - OTF2\_MetricMode\_enum, [203](#)
  - OTF2\_MetricOccurrence\_enum, [203](#)
  - OTF2\_MetricScope\_enum, [204](#)
  - OTF2\_MetricTiming\_enum, [204](#)
  - OTF2\_MetricType\_enum, [205](#)
  - OTF2\_MetricValueProperty\_enum, [205](#)
  - OTF2\_ParameterType\_enum, [206](#)
  - OTF2\_RecorderKind\_enum, [206](#)
  - OTF2\_RegionFlag\_enum, [206](#)
  - OTF2\_RegionRole\_enum, [207](#)
  - OTF2\_SystemTreeDomain\_enum, [209](#)
- OTF2\_DefReader.h
  - OTF2\_DefReader\_GetLocationID, [210](#)
  - OTF2\_DefReader\_ReadDefinitions, [210](#)
  - OTF2\_DefReader\_SetCallbacks, [211](#)
- OTF2\_DefReader\_GetLocationID
  - OTF2\_DefReader.h, [210](#)
- OTF2\_DefReader\_ReadDefinitions
  - OTF2\_DefReader.h, [210](#)
- OTF2\_DefReader\_SetCallbacks
  - OTF2\_DefReader.h, [211](#)
- OTF2\_DefReaderCallback\_Attribute
  - OTF2\_DefReaderCallbacks.h, [218](#)
- OTF2\_DefReaderCallback\_Callpath
  - OTF2\_DefReaderCallbacks.h, [219](#)
- OTF2\_DefReaderCallback\_Callsite
  - OTF2\_DefReaderCallbacks.h, [219](#)
- OTF2\_DefReaderCallback\_CartCoordinate
  - OTF2\_DefReaderCallbacks.h, [220](#)
- OTF2\_DefReaderCallback\_CartDimension
  - OTF2\_DefReaderCallbacks.h, [221](#)
- OTF2\_DefReaderCallback\_CartTopology
  - OTF2\_DefReaderCallbacks.h, [221](#)
- OTF2\_DefReaderCallback\_ClockOffset
  - OTF2\_DefReaderCallbacks.h, [222](#)
- OTF2\_DefReaderCallback\_Comm
  - OTF2\_DefReaderCallbacks.h, [223](#)
- OTF2\_DefReaderCallback\_Group
  - OTF2\_DefReaderCallbacks.h, [224](#)
- OTF2\_DefReaderCallback\_Location
  - OTF2\_DefReaderCallbacks.h, [224](#)
- OTF2\_DefReaderCallback\_LocationGroup
  - OTF2\_DefReaderCallbacks.h, [225](#)
- OTF2\_DefReaderCallback\_LocationGroupProperty
  - OTF2\_DefReaderCallbacks.h, [226](#)
- OTF2\_DefReaderCallback\_LocationProperty
  - OTF2\_DefReaderCallbacks.h, [226](#)
- OTF2\_DefReaderCallback\_MappingTable
  - OTF2\_DefReaderCallbacks.h, [227](#)
- OTF2\_DefReaderCallback\_MetricClass

## INDEX

---

- OTF2\_DefReaderCallbacks.h, [227](#)
- OTF2\_DefReaderCallback\_MetricClassRecorder [225](#)
- OTF2\_DefReaderCallbacks.h, [228](#)
- OTF2\_DefReaderCallback\_MetricInstance
- OTF2\_DefReaderCallbacks.h, [229](#)
- OTF2\_DefReaderCallback\_MetricMember
- OTF2\_DefReaderCallbacks.h, [230](#)
- OTF2\_DefReaderCallback\_Parameter
- OTF2\_DefReaderCallbacks.h, [231](#)
- OTF2\_DefReaderCallback\_Region
- OTF2\_DefReaderCallbacks.h, [231](#)
- OTF2\_DefReaderCallback\_RmaWin
- OTF2\_DefReaderCallbacks.h, [232](#)
- OTF2\_DefReaderCallback\_String
- OTF2\_DefReaderCallbacks.h, [233](#)
- OTF2\_DefReaderCallback\_SystemTreeNode
- OTF2\_DefReaderCallbacks.h, [233](#)
- OTF2\_DefReaderCallback\_SystemTreeNodeDomain
- OTF2\_DefReaderCallbacks.h, [234](#)
- OTF2\_DefReaderCallback\_SystemTreeNodeProperty
- OTF2\_DefReaderCallbacks.h, [234](#)
- OTF2\_DefReaderCallback\_Unknown
- OTF2\_DefReaderCallbacks.h, [235](#)
- OTF2\_DefReaderCallbacks.h
- OTF2\_DefReaderCallback\_Attribute,
- [218](#)
- OTF2\_DefReaderCallback\_Callpath,
- [219](#)
- OTF2\_DefReaderCallback\_Callsite,
- [219](#)
- OTF2\_DefReaderCallback\_CartCoordinate,
- [220](#)
- OTF2\_DefReaderCallback\_CartDimension,
- [221](#)
- OTF2\_DefReaderCallback\_CartTopology,
- [221](#)
- OTF2\_DefReaderCallback\_ClockOffset,
- [222](#)
- OTF2\_DefReaderCallback\_Comm,
- [223](#)
- OTF2\_DefReaderCallback\_Group, [224](#)
- OTF2\_DefReaderCallback\_Location,
- [224](#)
- OTF2\_DefReaderCallback\_LocationGroup,
- [225](#)
- OTF2\_DefReaderCallback\_LocationGroupProperty,
- [226](#)
- OTF2\_DefReaderCallback\_LocationProperty,
- [226](#)
- OTF2\_DefReaderCallback\_MappingTable,
- [227](#)
- OTF2\_DefReaderCallback\_MetricClass,
- [227](#)
- OTF2\_DefReaderCallback\_MetricClassRecorder,
- [228](#)
- OTF2\_DefReaderCallback\_MetricInstance,
- [229](#)
- OTF2\_DefReaderCallback\_MetricMember,
- [230](#)
- OTF2\_DefReaderCallback\_Parameter,
- [231](#)
- OTF2\_DefReaderCallback\_Region,
- [231](#)
- OTF2\_DefReaderCallback\_RmaWin,
- [232](#)
- OTF2\_DefReaderCallback\_String, [233](#)
- OTF2\_DefReaderCallback\_SystemTreeNode,
- [233](#)
- OTF2\_DefReaderCallback\_SystemTreeNodeDomain,
- [234](#)
- OTF2\_DefReaderCallback\_SystemTreeNodeProperty,
- [234](#)
- OTF2\_DefReaderCallback\_Unknown,
- [235](#)
- OTF2\_DefReaderCallbacks\_Clear, [236](#)
- OTF2\_DefReaderCallbacks\_Delete,
- [236](#)
- OTF2\_DefReaderCallbacks\_New, [236](#)
- OTF2\_DefReaderCallbacks\_SetAttributeCallback,
- [236](#)
- OTF2\_DefReaderCallbacks\_SetCallpathCallback,
- [237](#)
- OTF2\_DefReaderCallbacks\_SetCallsiteCallback,
- [237](#)
- OTF2\_DefReaderCallbacks\_SetCartCoordinateCallback,
- [238](#)

- OTF2\_DefReaderCallbacks\_SetCartCoordinateCallback  
238  
OTF2\_DefReaderCallbacks.h, 236
- OTF2\_DefReaderCallbacks\_SetCartTopologyCallback  
239  
OTF2\_DefReaderCallbacks.h, 236
- OTF2\_DefReaderCallbacks\_SetClockOffsetCallback  
239  
OTF2\_DefReaderCallbacks.h, 236
- OTF2\_DefReaderCallbacks\_SetCommCallback  
240  
OTF2\_DefReaderCallbacks.h, 236
- OTF2\_DefReaderCallbacks\_SetGroupCallback  
240  
OTF2\_DefReaderCallbacks.h, 237
- OTF2\_DefReaderCallbacks\_SetLocationCallback  
241  
OTF2\_DefReaderCallbacks.h, 237
- OTF2\_DefReaderCallbacks\_SetLocationCallback  
241  
OTF2\_DefReaderCallbacks.h, 238
- OTF2\_DefReaderCallbacks\_SetLocationCallback  
242  
OTF2\_DefReaderCallbacks.h, 238
- OTF2\_DefReaderCallbacks\_SetLocationCallback  
242  
OTF2\_DefReaderCallbacks.h, 239
- OTF2\_DefReaderCallbacks\_SetMappingCallback  
243  
OTF2\_DefReaderCallbacks.h, 239
- OTF2\_DefReaderCallbacks\_SetMetricCallback  
244  
OTF2\_DefReaderCallbacks.h, 240
- OTF2\_DefReaderCallbacks\_SetMetricCallback  
244  
OTF2\_DefReaderCallbacks.h, 240
- OTF2\_DefReaderCallbacks\_SetMetricCallback  
245  
OTF2\_DefReaderCallbacks.h, 241
- OTF2\_DefReaderCallbacks\_SetMetricCallback  
245  
OTF2\_DefReaderCallbacks.h, 241
- OTF2\_DefReaderCallbacks\_SetParameterCallback  
246  
OTF2\_DefReaderCallbacks.h, 242
- OTF2\_DefReaderCallbacks\_SetRegionCallback  
246  
OTF2\_DefReaderCallbacks.h, 242
- OTF2\_DefReaderCallbacks\_SetRmaCallback  
247  
OTF2\_DefReaderCallbacks.h, 243
- OTF2\_DefReaderCallbacks\_SetStringCallback  
247  
OTF2\_DefReaderCallbacks.h, 244
- OTF2\_DefReaderCallbacks\_SetSystemCallback  
248  
OTF2\_DefReaderCallbacks.h, 244
- OTF2\_DefReaderCallbacks\_SetSystemCallback  
248  
OTF2\_DefReaderCallbacks.h, 245
- OTF2\_DefReaderCallbacks\_SetSystemCallback  
249  
OTF2\_DefReaderCallbacks.h, 245
- OTF2\_DefReaderCallbacks\_SetUnknownCallback  
250  
OTF2\_DefReaderCallbacks.h, 246

## INDEX

---

OTF2\_DefReaderCallbacks\_SetRegionCallback, OTF2\_DefWriter\_WriteMetricMember,  
OTF2\_DefReaderCallbacks.h, [246](#) [264](#)  
OTF2\_DefReaderCallbacks\_SetRmaWinCallback, OTF2\_DefWriter\_WriteParameter, [265](#)  
OTF2\_DefReaderCallbacks.h, [247](#) OTF2\_DefWriter\_WriteRegion, [266](#)  
OTF2\_DefReaderCallbacks\_SetStringCallback, OTF2\_DefWriter\_WriteRmaWin, [266](#)  
OTF2\_DefReaderCallbacks.h, [247](#) OTF2\_DefWriter\_WriteString, [267](#)  
OTF2\_DefReaderCallbacks\_SetSystemTreeNodeCallback, OTF2\_DefWriter\_WriteSystemTreeNode,  
OTF2\_DefReaderCallbacks.h, [248](#) [267](#)  
OTF2\_DefReaderCallbacks\_SetSystemTreeNodeDomainCallback, OTF2\_DefWriter\_WriteSystemTreeNodeDomain,  
OTF2\_DefReaderCallbacks.h, [248](#) [268](#)  
OTF2\_DefReaderCallbacks\_SetSystemTreeNodePropertyCallback, OTF2\_DefWriter\_WriteSystemTreeNodeProperty,  
OTF2\_DefReaderCallbacks.h, [249](#) [269](#)  
OTF2\_DefReaderCallbacks\_SetUnknownCallback, OTF2\_DefWriter\_GetLocationID  
OTF2\_DefReaderCallbacks.h, [250](#) OTF2\_DefWriter.h, [253](#)  
OTF2\_DefWriter.h OTF2\_DefWriter\_WriteAttribute  
OTF2\_DefWriter\_GetLocationID, [253](#) OTF2\_DefWriter.h, [254](#)  
OTF2\_DefWriter\_WriteAttribute, [254](#) OTF2\_DefWriter\_WriteCallpath  
OTF2\_DefWriter\_WriteCallpath, [254](#) OTF2\_DefWriter.h, [254](#)  
OTF2\_DefWriter\_WriteCallsite, [255](#) OTF2\_DefWriter\_WriteCallsite  
OTF2\_DefWriter\_WriteCartCoordinate, OTF2\_DefWriter.h, [255](#)  
[255](#) OTF2\_DefWriter\_WriteCartCoordinate  
OTF2\_DefWriter\_WriteCartDimension, OTF2\_DefWriter.h, [255](#)  
[256](#) OTF2\_DefWriter\_WriteCartDimension  
OTF2\_DefWriter\_WriteCartTopology, OTF2\_DefWriter.h, [256](#)  
[257](#) OTF2\_DefWriter\_WriteCartTopology  
OTF2\_DefWriter\_WriteClockOffset, OTF2\_DefWriter.h, [257](#)  
[257](#) OTF2\_DefWriter\_WriteClockOffset  
OTF2\_DefWriter\_WriteComm, [258](#) OTF2\_DefWriter.h, [257](#)  
OTF2\_DefWriter\_WriteGroup, [258](#) OTF2\_DefWriter\_WriteComm  
OTF2\_DefWriter\_WriteLocation, [259](#) OTF2\_DefWriter.h, [258](#)  
OTF2\_DefWriter\_WriteLocationGroup, OTF2\_DefWriter\_WriteGroup  
[260](#) OTF2\_DefWriter.h, [258](#)  
OTF2\_DefWriter\_WriteLocationGroupProperty, OTF2\_DefWriter\_WriteLocation  
[260](#) OTF2\_DefWriter.h, [259](#)  
OTF2\_DefWriter\_WriteLocationProperty, OTF2\_DefWriter\_WriteLocationGroup  
[261](#) OTF2\_DefWriter.h, [260](#)  
OTF2\_DefWriter\_WriteMappingTable, OTF2\_DefWriter\_WriteLocationGroupProperty  
[262](#) OTF2\_DefWriter.h, [260](#)  
OTF2\_DefWriter\_WriteMetricClass, OTF2\_DefWriter\_WriteLocationProperty  
[262](#) OTF2\_DefWriter.h, [261](#)  
OTF2\_DefWriter\_WriteMetricClassReference, OTF2\_DefWriter\_WriteMappingTable  
[263](#) OTF2\_DefWriter.h, [262](#)  
OTF2\_DefWriter\_WriteMetricInstance, OTF2\_DefWriter\_WriteMetricClass  
[263](#) OTF2\_DefWriter.h, [262](#)

OTF2\_DefWriter\_WriteMetricClassRecorder  
     OTF2\_DefWriter.h, [263](#)  
 OTF2\_DefWriter\_WriteMetricInstance  
     OTF2\_DefWriter.h, [263](#)  
 OTF2\_DefWriter\_WriteMetricMember  
     OTF2\_DefWriter.h, [264](#)  
 OTF2\_DefWriter\_WriteParameter  
     OTF2\_DefWriter.h, [265](#)  
 OTF2\_DefWriter\_WriteRegion  
     OTF2\_DefWriter.h, [266](#)  
 OTF2\_DefWriter\_WriteRmaWin  
     OTF2\_DefWriter.h, [266](#)  
 OTF2\_DefWriter\_WriteString  
     OTF2\_DefWriter.h, [267](#)  
 OTF2\_DefWriter\_WriteSystemTreeNode  
     OTF2\_DefWriter.h, [267](#)  
 OTF2\_DefWriter\_WriteSystemTreeNodeDomain  
     OTF2\_DefWriter.h, [268](#)  
 OTF2\_DefWriter\_WriteSystemTreeNodeProperty  
     OTF2\_DefWriter.h, [269](#)  
 OTF2\_Error\_GetDescription  
     OTF2\_ErrorCodes.h, [135](#)  
 OTF2\_Error\_GetName  
     OTF2\_ErrorCodes.h, [135](#)  
 OTF2\_Error\_RegisterCallback  
     OTF2\_ErrorCodes.h, [135](#)  
 OTF2\_ErrorCallback  
     OTF2\_ErrorCodes.h, [131](#)  
 OTF2\_ErrorCode  
     OTF2\_ErrorCodes.h, [131](#)  
 OTF2\_ErrorCodes.h  
     OTF2\_Error\_GetDescription, [135](#)  
     OTF2\_Error\_GetName, [135](#)  
     OTF2\_Error\_RegisterCallback, [135](#)  
     OTF2\_ErrorCallback, [131](#)  
     OTF2\_ErrorCode, [131](#)  
 OTF2\_Events.h  
     OTF2\_CollectiveOp\_enum, [272](#)  
     OTF2\_LockType\_enum, [273](#)  
     OTF2\_MeasurementMode\_enum, [273](#)  
     OTF2\_RmaAtomicType\_enum, [274](#)  
     OTF2\_RmaSyncLevel\_enum, [275](#)  
     OTF2\_RmaSyncType\_enum, [275](#)  
 OTF2\_EventSizeEstimator.h  
     OTF2\_EventSizeEstimator\_Delete, [281](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfAttributeList,  
         [282](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfBufferFlushEvent,  
         [282](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfEnterEvent,  
         [282](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfLeaveEvent,  
         [283](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMeasurementOnOffEvent,  
         [283](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMetricEvent,  
         [284](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiCollectiveBeginEvent,  
         [284](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiCollectiveEndEvent,  
         [284](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiIrecvEvent,  
         [285](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiIrecvRequestEvent,  
         [285](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiIsendCompleteEvent,  
         [286](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiIsendEvent,  
         [286](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiRecvEvent,  
         [286](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiRequestCancelledEvent,  
         [287](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiRequestTestEvent,  
         [287](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfMpiSendEvent,  
         [288](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfOmpAcquireLockEvent,  
         [288](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfOmpForkEvent,  
         [288](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfOmpJoinEvent,  
         [289](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfOmpReleaseLockEvent,  
         [289](#)  
     OTF2\_EventSizeEstimator\_GetSizeOfOmpTaskCompleteEvent,  
         [290](#)



## INDEX

---

OTF2\_EventSizeEstimator\_GetSizeOfOmpTaskCreateEvent, 290  
OTF2\_EventSizeEstimator\_GetSizeOfOmpTaskSwitchEvent, 291  
OTF2\_EventSizeEstimator\_GetSizeOfParameterIntEvent, 291  
OTF2\_EventSizeEstimator\_GetSizeOfParameterStringEvent, 292  
OTF2\_EventSizeEstimator\_GetSizeOfParameterUnsignedIntEvent, 292  
OTF2\_EventSizeEstimator\_GetSizeOfRmaAcquireLockEvent, 292  
OTF2\_EventSizeEstimator\_GetSizeOfRmaAtomicEvent, 293  
OTF2\_EventSizeEstimator\_GetSizeOfRmaCollectiveBeginEvent, 293  
OTF2\_EventSizeEstimator\_GetSizeOfRmaCollectiveEndEvent, 293  
OTF2\_EventSizeEstimator\_GetSizeOfRmaGetEvent, 294  
OTF2\_EventSizeEstimator\_GetSizeOfRmaGroupSyncEvent, 294  
OTF2\_EventSizeEstimator\_GetSizeOfRmaOpCompleteBlockingEvent, 295  
OTF2\_EventSizeEstimator\_GetSizeOfRmaOpCompleteNonBlockingEvent, 295  
OTF2\_EventSizeEstimator\_GetSizeOfRmaOpCompleteRemoteEvent, 295  
OTF2\_EventSizeEstimator\_GetSizeOfRmaOpTestEvent, 296  
OTF2\_EventSizeEstimator\_GetSizeOfRmaPutEvent, 296  
OTF2\_EventSizeEstimator\_GetSizeOfRmaReleaseLockEvent, 297  
OTF2\_EventSizeEstimator\_GetSizeOfRmaRequestLockEvent, 297  
OTF2\_EventSizeEstimator\_GetSizeOfRmaSyncEvent, 297  
OTF2\_EventSizeEstimator\_GetSizeOfRmaTryLockEvent, 298  
OTF2\_EventSizeEstimator\_GetSizeOfRmaWaitChangeSizeEvent, 298  
OTF2\_EventSizeEstimator\_GetSizeOfRmaWaitCreateEvent, 299  
OTF2\_EventSizeEstimator\_GetSizeOfRmaWinDestroyEvent, 299  
OTF2\_EventSizeEstimator\_GetSizeOfThreadAcquireLockEvent, 299  
OTF2\_EventSizeEstimator\_GetSizeOfThreadBeginEvent, 300  
OTF2\_EventSizeEstimator\_GetSizeOfThreadCreateEvent, 300  
OTF2\_EventSizeEstimator\_GetSizeOfThreadEndEvent, 301  
OTF2\_EventSizeEstimator\_GetSizeOfThreadForkEvent, 301  
OTF2\_EventSizeEstimator\_GetSizeOfThreadJoinEvent, 301  
OTF2\_EventSizeEstimator\_GetSizeOfThreadReleaseLockEvent, 302  
OTF2\_EventSizeEstimator\_GetSizeOfThreadTaskCompleteEvent, 302  
OTF2\_EventSizeEstimator\_GetSizeOfThreadTaskCreateEvent, 303  
OTF2\_EventSizeEstimator\_GetSizeOfThreadTaskSwitchEvent, 303  
OTF2\_EventSizeEstimator\_GetSizeOfThreadTeamBeginEvent, 303  
OTF2\_EventSizeEstimator\_GetSizeOfThreadTeamEndEvent, 304  
OTF2\_EventSizeEstimator\_GetSizeOfThreadWaitEvent, 304  
OTF2\_EventSizeEstimator\_GetSizeOfTimestamp, 305  
OTF2\_EventSizeEstimator\_New, 305  
OTF2\_EventSizeEstimator\_SetNumberOfAttributeDefinitions, 305  
OTF2\_EventSizeEstimator\_SetNumberOfCommDefinitions, 306  
OTF2\_EventSizeEstimator\_SetNumberOfGroupDefinitions, 306  
OTF2\_EventSizeEstimator\_SetNumberOfLocationDefinitions, 307  
OTF2\_EventSizeEstimator\_SetNumberOfMetricDefinitions, 307  
OTF2\_EventSizeEstimator\_SetNumberOfParameterDefinitions, 308

- 
- OTF2\_EventSizeEstimator\_SetNumberOfEvents [308](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfOmpJoinEvent [OTF2\\_EventSizeEstimator.h, 289](#)
  - OTF2\_EventSizeEstimator\_SetNumberOfEvents [309](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfOmpReleaseLockEvent [OTF2\\_EventSizeEstimator.h, 289](#)
  - OTF2\_EventSizeEstimator\_SetNumberOfEvents [309](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfOmpTaskCompleteEvent [OTF2\\_EventSizeEstimator.h, 290](#)
  - OTF2\_EventSizeEstimator\_Delete [OTF2\\_EventSizeEstimator.h, 281](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfOmpTaskCreateEvent [OTF2\\_EventSizeEstimator.h, 290](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfAttributeEvent [OTF2\\_EventSizeEstimator.h, 282](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfOmpTaskSwitchEvent [OTF2\\_EventSizeEstimator.h, 291](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfBufferEvent [OTF2\\_EventSizeEstimator.h, 282](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfParameterIntEvent [OTF2\\_EventSizeEstimator.h, 291](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfEntryPointEvent [OTF2\\_EventSizeEstimator.h, 282](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfParameterStringEvent [OTF2\\_EventSizeEstimator.h, 292](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfLeaderEvent [OTF2\\_EventSizeEstimator.h, 283](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfParameterUnsignedIntEvent [OTF2\\_EventSizeEstimator.h, 292](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMemoryEvent [OTF2\\_EventSizeEstimator.h, 283](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaAcquireLockEvent [OTF2\\_EventSizeEstimator.h, 292](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMemoryEvent [OTF2\\_EventSizeEstimator.h, 284](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaAtomicEvent [OTF2\\_EventSizeEstimator.h, 293](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 284](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaCollectiveBeginEvent [OTF2\\_EventSizeEstimator.h, 293](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 284](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaCollectiveEndEvent [OTF2\\_EventSizeEstimator.h, 293](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 285](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaGetEvent [OTF2\\_EventSizeEstimator.h, 294](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 285](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaGroupSyncEvent [OTF2\\_EventSizeEstimator.h, 294](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 286](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaOpCompleteBlockingEvent [OTF2\\_EventSizeEstimator.h, 295](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 286](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaOpCompleteNonBlockingEvent [OTF2\\_EventSizeEstimator.h, 295](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 286](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaOpCompleteRemoteEvent [OTF2\\_EventSizeEstimator.h, 295](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 287](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaOpTestEvent [OTF2\\_EventSizeEstimator.h, 296](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 287](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaPutEvent [OTF2\\_EventSizeEstimator.h, 296](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfMPIEvent [OTF2\\_EventSizeEstimator.h, 288](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaReleaseLockEvent [OTF2\\_EventSizeEstimator.h, 297](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfOmpEvent [OTF2\\_EventSizeEstimator.h, 288](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaRequestLockEvent [OTF2\\_EventSizeEstimator.h, 297](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfOmpEvent [OTF2\\_EventSizeEstimator.h, 288](#)
  - OTF2\_EventSizeEstimator\_GetSizeOfRmaSyncEvent [OTF2\\_EventSizeEstimator.h, 297](#)
-



## INDEX

---

OTF2\_EventSizeEstimator\_GetSizeOfRmaWin 307  
OTF2\_EventSizeEstimator.h, 298  
OTF2\_EventSizeEstimator\_SetNumberOfLocationDefinitions  
OTF2\_EventSizeEstimator.h, 307  
OTF2\_EventSizeEstimator\_GetSizeOfRmaWin 307  
OTF2\_EventSizeEstimator.h, 298  
OTF2\_EventSizeEstimator\_SetNumberOfMetricDefinitions  
OTF2\_EventSizeEstimator.h, 307  
OTF2\_EventSizeEstimator\_GetSizeOfRmaWin 308  
OTF2\_EventSizeEstimator.h, 299  
OTF2\_EventSizeEstimator\_SetNumberOfParameterDefinitions  
OTF2\_EventSizeEstimator.h, 308  
OTF2\_EventSizeEstimator\_GetSizeOfRmaWin 308  
OTF2\_EventSizeEstimator.h, 299  
OTF2\_EventSizeEstimator\_SetNumberOfRegionDefinitions  
OTF2\_EventSizeEstimator.h, 308  
OTF2\_EventSizeEstimator\_GetSizeOfThread 309  
OTF2\_EventSizeEstimator.h, 299  
OTF2\_EventSizeEstimator\_SetNumberOfRmaWinDefinitions  
OTF2\_EventSizeEstimator.h, 309  
OTF2\_EventSizeEstimator\_GetSizeOfThread 309  
OTF2\_EventSizeEstimator.h, 300  
OTF2\_EventSizeEstimator\_SetNumberOfStringDefinitions  
OTF2\_EventSizeEstimator.h, 309  
OTF2\_EventSizeEstimator\_GetSizeOfThread 309  
OTF2\_EventSizeEstimator.h, 300  
OTF2\_EvtReader.h  
OTF2\_EventSizeEstimator.h, 300  
OTF2\_EvtReader\_ApplyClockOffsets,  
OTF2\_EventSizeEstimator\_GetSizeOfThreadEndEvent  
OTF2\_EventSizeEstimator.h, 301  
OTF2\_EvtReader\_ApplyMappingTables,  
OTF2\_EventSizeEstimator\_GetSizeOfThreadEvent  
OTF2\_EventSizeEstimator.h, 301  
OTF2\_EvtReader\_GetLocationID, 312  
OTF2\_EventSizeEstimator\_GetSizeOfThreadJoinEvent  
OTF2\_EventSizeEstimator.h, 301  
OTF2\_EvtReader\_GetPos, 312  
OTF2\_EventSizeEstimator\_GetSizeOfThreadReadEvent  
OTF2\_EventSizeEstimator.h, 302  
OTF2\_EvtReader\_ReadEvents, 313  
OTF2\_EventSizeEstimator\_GetSizeOfThreadReadEventBackward,  
OTF2\_EventSizeEstimator.h, 302  
313  
OTF2\_EventSizeEstimator\_GetSizeOfThreadSeek  
OTF2\_EventSizeEstimator.h, 302  
OTF2\_EvtReader\_SetCallbacks, 314  
OTF2\_EventSizeEstimator\_GetSizeOfThreadTimestampRewrite,  
OTF2\_EventSizeEstimator.h, 303  
314  
OTF2\_EventSizeEstimator\_GetSizeOfThreadWaitEvent  
OTF2\_EventSizeEstimator.h, 303  
OTF2\_EvtReader\_ApplyClockOffsets  
OTF2\_EventSizeEstimator.h, 303  
OTF2\_EvtReader.h, 311  
OTF2\_EventSizeEstimator\_GetSizeOfThreadWaitEvent  
OTF2\_EventSizeEstimator.h, 303  
OTF2\_EvtReader\_ApplyMappingTables  
OTF2\_EventSizeEstimator.h, 303  
OTF2\_EvtReader.h, 312  
OTF2\_EventSizeEstimator\_GetSizeOfThreadWaitEvent  
OTF2\_EventSizeEstimator.h, 304  
OTF2\_EvtReader\_GetLocationID  
OTF2\_EventSizeEstimator.h, 304  
OTF2\_EvtReader.h, 312  
OTF2\_EventSizeEstimator\_GetSizeOfThreadWaitEvent  
OTF2\_EventSizeEstimator.h, 304  
OTF2\_EvtReader\_GetPos  
OTF2\_EventSizeEstimator.h, 304  
OTF2\_EvtReader.h, 312  
OTF2\_EventSizeEstimator\_GetSizeOfThreadWaitEvent  
OTF2\_EventSizeEstimator.h, 305  
OTF2\_EvtReader\_ReadEvents  
OTF2\_EventSizeEstimator.h, 305  
OTF2\_EvtReader.h, 313  
OTF2\_EventSizeEstimator\_New  
OTF2\_EventSizeEstimator.h, 305  
OTF2\_EvtReader\_ReadEventsBackward  
OTF2\_EventSizeEstimator.h, 305  
OTF2\_EvtReader.h, 313  
OTF2\_EventSizeEstimator\_SetNumberOfThreadWaitEvent  
OTF2\_EventSizeEstimator.h, 305  
OTF2\_EvtReader\_SetCallbacks  
OTF2\_EventSizeEstimator.h, 306  
OTF2\_EvtReader.h, 314  
OTF2\_EventSizeEstimator\_SetNumberOfThreadWaitEvent  
OTF2\_EventSizeEstimator.h, 306  
OTF2\_EvtReader\_TimeStampRewrite  
OTF2\_EventSizeEstimator.h, 306  
OTF2\_EvtReader.h, 314

OTF2_EvtReaderCallback_BufferFlush	OTF2_EvtReaderCallback_ParameterInt
OTF2_EvtReaderCallbacks.h, 328	OTF2_EvtReaderCallbacks.h, 344
OTF2_EvtReaderCallback_Enter	OTF2_EvtReaderCallback_ParameterString
OTF2_EvtReaderCallbacks.h, 329	OTF2_EvtReaderCallbacks.h, 345
OTF2_EvtReaderCallback_Leave	OTF2_EvtReaderCallback_ParameterUnsignedInt
OTF2_EvtReaderCallbacks.h, 330	OTF2_EvtReaderCallbacks.h, 346
OTF2_EvtReaderCallback_Measurement	OTF2_EvtReaderCallback_RmaAcquireLock
OTF2_EvtReaderCallbacks.h, 330	OTF2_EvtReaderCallbacks.h, 346
OTF2_EvtReaderCallback_Metric	OTF2_EvtReaderCallback_RmaAtomic
OTF2_EvtReaderCallbacks.h, 331	OTF2_EvtReaderCallbacks.h, 347
OTF2_EvtReaderCallback_MpiCollectiveBegin	OTF2_EvtReaderCallback_RmaCollectiveBegin
OTF2_EvtReaderCallbacks.h, 332	OTF2_EvtReaderCallbacks.h, 348
OTF2_EvtReaderCallback_MpiCollectiveEnd	OTF2_EvtReaderCallback_RmaCollectiveEnd
OTF2_EvtReaderCallbacks.h, 332	OTF2_EvtReaderCallbacks.h, 349
OTF2_EvtReaderCallback_MpiIrecv	OTF2_EvtReaderCallback_RmaGet
OTF2_EvtReaderCallbacks.h, 333	OTF2_EvtReaderCallbacks.h, 349
OTF2_EvtReaderCallback_MpiIrecvRequest	OTF2_EvtReaderCallback_RmaGroupSync
OTF2_EvtReaderCallbacks.h, 334	OTF2_EvtReaderCallbacks.h, 350
OTF2_EvtReaderCallback_MpiIsend	OTF2_EvtReaderCallback_RmaOpCompleteBlocking
OTF2_EvtReaderCallbacks.h, 335	OTF2_EvtReaderCallbacks.h, 351
OTF2_EvtReaderCallback_MpiIsendComplete	OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking
OTF2_EvtReaderCallbacks.h, 335	OTF2_EvtReaderCallbacks.h, 352
OTF2_EvtReaderCallback_MpiRecv	OTF2_EvtReaderCallback_RmaOpCompleteRemote
OTF2_EvtReaderCallbacks.h, 336	OTF2_EvtReaderCallbacks.h, 352
OTF2_EvtReaderCallback_MpiRequestCancel	OTF2_EvtReaderCallback_RmaOpTest
OTF2_EvtReaderCallbacks.h, 337	OTF2_EvtReaderCallbacks.h, 353
OTF2_EvtReaderCallback_MpiRequestTest	OTF2_EvtReaderCallback_RmaPut
OTF2_EvtReaderCallbacks.h, 338	OTF2_EvtReaderCallbacks.h, 354
OTF2_EvtReaderCallback_MpiSend	OTF2_EvtReaderCallback_RmaReleaseLock
OTF2_EvtReaderCallbacks.h, 338	OTF2_EvtReaderCallbacks.h, 355
OTF2_EvtReaderCallback_OmpAcquireLock	OTF2_EvtReaderCallback_RmaRequestLock
OTF2_EvtReaderCallbacks.h, 339	OTF2_EvtReaderCallbacks.h, 355
OTF2_EvtReaderCallback_OmpFork	OTF2_EvtReaderCallback_RmaSync
OTF2_EvtReaderCallbacks.h, 340	OTF2_EvtReaderCallbacks.h, 356
OTF2_EvtReaderCallback_OmpJoin	OTF2_EvtReaderCallback_RmaTryLock
OTF2_EvtReaderCallbacks.h, 341	OTF2_EvtReaderCallbacks.h, 357
OTF2_EvtReaderCallback_OmpReleaseLock	OTF2_EvtReaderCallback_RmaWaitChange
OTF2_EvtReaderCallbacks.h, 341	OTF2_EvtReaderCallbacks.h, 358
OTF2_EvtReaderCallback_OmpTaskComplete	OTF2_EvtReaderCallback_RmaWinCreate
OTF2_EvtReaderCallbacks.h, 342	OTF2_EvtReaderCallbacks.h, 358
OTF2_EvtReaderCallback_OmpTaskCreate	OTF2_EvtReaderCallback_RmaWinDestroy
OTF2_EvtReaderCallbacks.h, 343	OTF2_EvtReaderCallbacks.h, 359
OTF2_EvtReaderCallback_OmpTaskSwitch	OTF2_EvtReaderCallback_ThreadAcquireLock
OTF2_EvtReaderCallbacks.h, 343	OTF2_EvtReaderCallbacks.h, 360

## INDEX

---

- OTF2\_EvtReaderCallback\_ThreadBegin  
OTF2\_EvtReaderCallbacks.h, [361](#)
- OTF2\_EvtReaderCallback\_ThreadCreate  
OTF2\_EvtReaderCallbacks.h, [361](#)
- OTF2\_EvtReaderCallback\_ThreadEnd  
OTF2\_EvtReaderCallbacks.h, [362](#)
- OTF2\_EvtReaderCallback\_ThreadFork  
OTF2\_EvtReaderCallbacks.h, [363](#)
- OTF2\_EvtReaderCallback\_ThreadJoin  
OTF2\_EvtReaderCallbacks.h, [363](#)
- OTF2\_EvtReaderCallback\_ThreadReleaseLock  
OTF2\_EvtReaderCallbacks.h, [364](#)
- OTF2\_EvtReaderCallback\_ThreadTaskComplete  
OTF2\_EvtReaderCallbacks.h, [365](#)
- OTF2\_EvtReaderCallback\_ThreadTaskCreate  
OTF2\_EvtReaderCallbacks.h, [366](#)
- OTF2\_EvtReaderCallback\_ThreadTaskSwitch  
OTF2\_EvtReaderCallbacks.h, [366](#)
- OTF2\_EvtReaderCallback\_ThreadTeamBegin  
OTF2\_EvtReaderCallbacks.h, [367](#)
- OTF2\_EvtReaderCallback\_ThreadTeamEnd  
OTF2\_EvtReaderCallbacks.h, [368](#)
- OTF2\_EvtReaderCallback\_ThreadWait  
OTF2\_EvtReaderCallbacks.h, [369](#)
- OTF2\_EvtReaderCallback\_Unknown  
OTF2\_EvtReaderCallbacks.h, [369](#)
- OTF2\_EvtReaderCallbacks.h  
OTF2\_EvtReaderCallback\_BufferFlush,  
[328](#)  
OTF2\_EvtReaderCallback\_Enter, [329](#)  
OTF2\_EvtReaderCallback\_Leave, [330](#)  
OTF2\_EvtReaderCallback\_MeasurementOnOff,  
[330](#)  
OTF2\_EvtReaderCallback\_Metric, [331](#)  
OTF2\_EvtReaderCallback\_MpiCollectiveBegin,  
[332](#)  
OTF2\_EvtReaderCallback\_MpiCollectiveEnd,  
[332](#)  
OTF2\_EvtReaderCallback\_MpiIrecv,  
[333](#)  
OTF2\_EvtReaderCallback\_MpiIrecvRequest,  
[334](#)  
OTF2\_EvtReaderCallback\_MpiIsend,  
[335](#)  
OTF2\_EvtReaderCallback\_MpiIsendComplete,  
[335](#)  
OTF2\_EvtReaderCallback\_MpiRecv,  
[336](#)  
OTF2\_EvtReaderCallback\_MpiRequestCancelled,  
[337](#)  
OTF2\_EvtReaderCallback\_MpiRequestTest,  
[338](#)  
OTF2\_EvtReaderCallback\_MpiSend,  
[338](#)  
OTF2\_EvtReaderCallback\_OmpAcquireLock,  
[339](#)  
OTF2\_EvtReaderCallback\_OmpFork,  
[340](#)  
OTF2\_EvtReaderCallback\_OmpJoin,  
[341](#)  
OTF2\_EvtReaderCallback\_OmpReleaseLock,  
[341](#)  
OTF2\_EvtReaderCallback\_OmpTaskComplete,  
[342](#)  
OTF2\_EvtReaderCallback\_OmpTaskCreate,  
[343](#)  
OTF2\_EvtReaderCallback\_OmpTaskSwitch,  
[343](#)  
OTF2\_EvtReaderCallback\_ParameterInt,  
[344](#)  
OTF2\_EvtReaderCallback\_ParameterString,  
[345](#)  
OTF2\_EvtReaderCallback\_ParameterUnsignedInt,  
[346](#)  
OTF2\_EvtReaderCallback\_RmaAcquireLock,  
[346](#)  
OTF2\_EvtReaderCallback\_RmaAtomic,  
[347](#)  
OTF2\_EvtReaderCallback\_RmaCollectiveBegin,  
[348](#)  
OTF2\_EvtReaderCallback\_RmaCollectiveEnd,  
[349](#)  
OTF2\_EvtReaderCallback\_RmaGet,  
[349](#)  
OTF2\_EvtReaderCallback\_RmaGroupSync,  
[350](#)  
OTF2\_EvtReaderCallback\_RmaOpCompleteBlocking,  
[351](#)

## INDEX

---

OTF2_EvtReaderCallback_RmaOpComplete, <a href="#">352</a>	OTF2_EvtReaderCallback_ThreadTeamEnd, <a href="#">368</a>
OTF2_EvtReaderCallback_RmaOpComplete, <a href="#">352</a>	OTF2_EvtReaderCallback_ThreadWait, <a href="#">369</a>
OTF2_EvtReaderCallback_RmaOpTest, <a href="#">353</a>	OTF2_EvtReaderCallback_Unknown, <a href="#">369</a>
OTF2_EvtReaderCallback_RmaPut, <a href="#">354</a>	OTF2_EvtReaderCallbacks_Clear, <a href="#">370</a>
OTF2_EvtReaderCallback_RmaReleaseLock, <a href="#">355</a>	OTF2_EvtReaderCallbacks_Delete, <a href="#">370</a>
OTF2_EvtReaderCallback_RmaRequestLock, <a href="#">355</a>	OTF2_EvtReaderCallbacks_New, <a href="#">370</a>
OTF2_EvtReaderCallback_RmaSync, <a href="#">356</a>	OTF2_EvtReaderCallbacks_SetBufferFlushCallback, <a href="#">370</a>
OTF2_EvtReaderCallback_RmaTryLock, <a href="#">357</a>	OTF2_EvtReaderCallbacks_SetEnterCallback, <a href="#">371</a>
OTF2_EvtReaderCallback_RmaWaitChange, <a href="#">358</a>	OTF2_EvtReaderCallbacks_SetLeaveCallback, <a href="#">372</a>
OTF2_EvtReaderCallback_RmaWinCreate, <a href="#">358</a>	OTF2_EvtReaderCallbacks_SetMeasurementOnOffCallback, <a href="#">372</a>
OTF2_EvtReaderCallback_RmaWinDestroy, <a href="#">359</a>	OTF2_EvtReaderCallbacks_SetMetricCallback, <a href="#">373</a>
OTF2_EvtReaderCallback_ThreadAcquireLock, <a href="#">360</a>	OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback, <a href="#">373</a>
OTF2_EvtReaderCallback_ThreadBegin, <a href="#">361</a>	OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback, <a href="#">374</a>
OTF2_EvtReaderCallback_ThreadCreate, <a href="#">361</a>	OTF2_EvtReaderCallbacks_SetMpiIrecvCallback, <a href="#">374</a>
OTF2_EvtReaderCallback_ThreadEnd, <a href="#">362</a>	OTF2_EvtReaderCallbacks_SetMpiIrecvRequestCallback, <a href="#">375</a>
OTF2_EvtReaderCallback_ThreadFork, <a href="#">363</a>	OTF2_EvtReaderCallbacks_SetMpiIsendCallback, <a href="#">375</a>
OTF2_EvtReaderCallback_ThreadJoin, <a href="#">363</a>	OTF2_EvtReaderCallbacks_SetMpiIsendCompleteCallback, <a href="#">376</a>
OTF2_EvtReaderCallback_ThreadReleaseLock, <a href="#">364</a>	OTF2_EvtReaderCallbacks_SetMpiRecvCallback, <a href="#">377</a>
OTF2_EvtReaderCallback_ThreadTaskComplete, <a href="#">365</a>	OTF2_EvtReaderCallbacks_SetMpiRequestCancelledCallback, <a href="#">377</a>
OTF2_EvtReaderCallback_ThreadTaskCreate, <a href="#">366</a>	OTF2_EvtReaderCallbacks_SetMpiRequestTestCallback, <a href="#">378</a>
OTF2_EvtReaderCallback_ThreadTaskSwitch, <a href="#">366</a>	OTF2_EvtReaderCallbacks_SetMpiSendCallback, <a href="#">378</a>
OTF2_EvtReaderCallback_ThreadTeamBegin, <a href="#">367</a>	OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback, <a href="#">379</a>
	OTF2_EvtReaderCallbacks_SetOmpForkCallback, <a href="#">379</a>

---

## INDEX

---

OTF2\_EvtReaderCallbacks\_SetOmpJoinCallback, 380  
OTF2\_EvtReaderCallbacks\_SetOmpJoinCallback, 392  
OTF2\_EvtReaderCallbacks\_SetOmpReleaseLockCallback, 380  
OTF2\_EvtReaderCallbacks\_SetOmpReleaseLockCallback, 393  
OTF2\_EvtReaderCallbacks\_SetOmpTaskCompleteCallback, 381  
OTF2\_EvtReaderCallbacks\_SetOmpTaskCompleteCallback, 394  
OTF2\_EvtReaderCallbacks\_SetOmpTaskCreateCallback, 382  
OTF2\_EvtReaderCallbacks\_SetOmpTaskCreateCallback, 394  
OTF2\_EvtReaderCallbacks\_SetOmpTaskSwitchCallback, 382  
OTF2\_EvtReaderCallbacks\_SetThreadAcquireLockCallback, 395  
OTF2\_EvtReaderCallbacks\_SetParameterCallback, 383  
OTF2\_EvtReaderCallbacks\_SetThreadBeginCallback, 395  
OTF2\_EvtReaderCallbacks\_SetParameterCallback, 383  
OTF2\_EvtReaderCallbacks\_SetThreadCreateCallback, 396  
OTF2\_EvtReaderCallbacks\_SetParameterCallback, 384  
OTF2\_EvtReaderCallbacks\_SetThreadEndCallback, 396  
OTF2\_EvtReaderCallbacks\_SetRmaAcquireLockCallback, 384  
OTF2\_EvtReaderCallbacks\_SetThreadForkCallback, 397  
OTF2\_EvtReaderCallbacks\_SetRmaAtomCallback, 385  
OTF2\_EvtReaderCallbacks\_SetThreadJoinCallback, 397  
OTF2\_EvtReaderCallbacks\_SetRmaCollectCallback, 386  
OTF2\_EvtReaderCallbacks\_SetThreadReleaseLockCallback, 398  
OTF2\_EvtReaderCallbacks\_SetRmaCollectCallback, 386  
OTF2\_EvtReaderCallbacks\_SetThreadTaskCompleteCallback, 398  
OTF2\_EvtReaderCallbacks\_SetRmaGetCallback, 387  
OTF2\_EvtReaderCallbacks\_SetThreadTaskCreateCallback, 399  
OTF2\_EvtReaderCallbacks\_SetRmaGroupSyncCallback, 387  
OTF2\_EvtReaderCallbacks\_SetThreadTaskSwitchCallback, 400  
OTF2\_EvtReaderCallbacks\_SetRmaOpCompleteCallback, 388  
OTF2\_EvtReaderCallbacks\_SetThreadTeamBeginCallback, 400  
OTF2\_EvtReaderCallbacks\_SetRmaOpCompleteCallback, 388  
OTF2\_EvtReaderCallbacks\_SetThreadTeamEndCallback, 401  
OTF2\_EvtReaderCallbacks\_SetRmaOpCompleteCallback, 389  
OTF2\_EvtReaderCallbacks\_SetThreadWaitCallback, 401  
OTF2\_EvtReaderCallbacks\_SetRmaOpTestCallback, 390  
OTF2\_EvtReaderCallbacks\_SetUnknownCallback, 402  
OTF2\_EvtReaderCallbacks\_SetRmaOpTestCallback, 390  
OTF2\_EvtReaderCallbacks\_Clear, 390  
OTF2\_EvtReaderCallbacks.h, 370  
OTF2\_EvtReaderCallbacks\_SetRmaOpTestCallback, 391  
OTF2\_EvtReaderCallbacks\_Delete, 370  
OTF2\_EvtReaderCallbacks.h, 370  
OTF2\_EvtReaderCallbacks\_SetRmaOpTestCallback, 391  
OTF2\_EvtReaderCallbacks\_New, 370  
OTF2\_EvtReaderCallbacks.h, 370  
OTF2\_EvtReaderCallbacks\_SetRmaOpTestCallback, 392  
OTF2\_EvtReaderCallbacks\_SetBufferFlushCallback, 370  
OTF2\_EvtReaderCallbacks.h, 370



---

OTF2_EvtReaderCallbacks_SetEnterCallback	OTF2_EvtReaderCallbacks_SetParameterStringCallback
OTF2_EvtReaderCallbacks.h, 371	OTF2_EvtReaderCallbacks.h, 383
OTF2_EvtReaderCallbacks_SetLeaveCallback	OTF2_EvtReaderCallbacks_SetParameterUnsignedIntCallback
OTF2_EvtReaderCallbacks.h, 372	OTF2_EvtReaderCallbacks.h, 384
OTF2_EvtReaderCallbacks_SetMeasurementCallback	OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback
OTF2_EvtReaderCallbacks.h, 372	OTF2_EvtReaderCallbacks.h, 384
OTF2_EvtReaderCallbacks_SetMetricCallback	OTF2_EvtReaderCallbacks_SetRmaAtomicCallback
OTF2_EvtReaderCallbacks.h, 373	OTF2_EvtReaderCallbacks.h, 385
OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback	OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback
OTF2_EvtReaderCallbacks.h, 373	OTF2_EvtReaderCallbacks.h, 386
OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback	OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback
OTF2_EvtReaderCallbacks.h, 374	OTF2_EvtReaderCallbacks.h, 386
OTF2_EvtReaderCallbacks_SetMpiRecvCallback	OTF2_EvtReaderCallbacks_SetRmaGetCallback
OTF2_EvtReaderCallbacks.h, 374	OTF2_EvtReaderCallbacks.h, 387
OTF2_EvtReaderCallbacks_SetMpiRecvReduceCallback	OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback
OTF2_EvtReaderCallbacks.h, 375	OTF2_EvtReaderCallbacks.h, 387
OTF2_EvtReaderCallbacks_SetMpiSendCallback	OTF2_EvtReaderCallbacks_SetRmaOpCompleteBlockingCallback
OTF2_EvtReaderCallbacks.h, 375	OTF2_EvtReaderCallbacks.h, 388
OTF2_EvtReaderCallbacks_SetMpiSendRecvCallback	OTF2_EvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback
OTF2_EvtReaderCallbacks.h, 376	OTF2_EvtReaderCallbacks.h, 388
OTF2_EvtReaderCallbacks_SetMpiRecvReduceCallback	OTF2_EvtReaderCallbacks_SetRmaOpCompleteRemoteCallback
OTF2_EvtReaderCallbacks.h, 377	OTF2_EvtReaderCallbacks.h, 389
OTF2_EvtReaderCallbacks_SetMpiRequestCancelCallback	OTF2_EvtReaderCallbacks_SetRmaOpTestCallback
OTF2_EvtReaderCallbacks.h, 377	OTF2_EvtReaderCallbacks.h, 390
OTF2_EvtReaderCallbacks_SetMpiRequestReduceCallback	OTF2_EvtReaderCallbacks_SetRmaPutCallback
OTF2_EvtReaderCallbacks.h, 378	OTF2_EvtReaderCallbacks.h, 390
OTF2_EvtReaderCallbacks_SetMpiSendRecvCallback	OTF2_EvtReaderCallbacks_SetRmaReleaseLockCallback
OTF2_EvtReaderCallbacks.h, 378	OTF2_EvtReaderCallbacks.h, 391
OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback	OTF2_EvtReaderCallbacks_SetRmaRequestLockCallback
OTF2_EvtReaderCallbacks.h, 379	OTF2_EvtReaderCallbacks.h, 391
OTF2_EvtReaderCallbacks_SetOmpForkCallback	OTF2_EvtReaderCallbacks_SetRmaSyncCallback
OTF2_EvtReaderCallbacks.h, 379	OTF2_EvtReaderCallbacks.h, 392
OTF2_EvtReaderCallbacks_SetOmpJoinCallback	OTF2_EvtReaderCallbacks_SetRmaTryLockCallback
OTF2_EvtReaderCallbacks.h, 380	OTF2_EvtReaderCallbacks.h, 392
OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback	OTF2_EvtReaderCallbacks_SetRmaWaitChangeCallback
OTF2_EvtReaderCallbacks.h, 380	OTF2_EvtReaderCallbacks.h, 393
OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback	OTF2_EvtReaderCallbacks_SetRmaWinCreateCallback
OTF2_EvtReaderCallbacks.h, 381	OTF2_EvtReaderCallbacks.h, 394
OTF2_EvtReaderCallbacks_SetOmpTaskWaitCallback	OTF2_EvtReaderCallbacks_SetRmaWinDestroyCallback
OTF2_EvtReaderCallbacks.h, 382	OTF2_EvtReaderCallbacks.h, 394
OTF2_EvtReaderCallbacks_SetOmpTaskWaitForCallback	OTF2_EvtReaderCallbacks_SetThreadAcquireLockCallback
OTF2_EvtReaderCallbacks.h, 382	OTF2_EvtReaderCallbacks.h, 395
OTF2_EvtReaderCallbacks_SetParameterIntegerCallback	OTF2_EvtReaderCallbacks_SetThreadBeginCallback
OTF2_EvtReaderCallbacks.h, 383	OTF2_EvtReaderCallbacks.h, 395

---

## INDEX

---

OTF2\_EvtReaderCallbacks\_SetThreadCreateCallback, 416  
OTF2\_EvtReaderCallbacks.h, 396  
OTF2\_EvtReaderCallbacks\_SetThreadEndCallback, 417  
OTF2\_EvtReaderCallbacks.h, 396  
OTF2\_EvtReaderCallbacks\_SetThreadForkCallback, 418  
OTF2\_EvtReaderCallbacks.h, 397  
OTF2\_EvtReaderCallbacks\_SetThreadJoinCallback, 419  
OTF2\_EvtReaderCallbacks.h, 397  
OTF2\_EvtReaderCallbacks\_SetThreadReleaseLockCallback, 420  
OTF2\_EvtReaderCallbacks.h, 398  
OTF2\_EvtReaderCallbacks\_SetThreadTaskCompleteCallback, 421  
OTF2\_EvtReaderCallbacks.h, 398  
OTF2\_EvtReaderCallbacks\_SetThreadTaskCreateCallback, 422  
OTF2\_EvtReaderCallbacks.h, 399  
OTF2\_EvtReaderCallbacks\_SetThreadTaskSwitchCallback, 423  
OTF2\_EvtReaderCallbacks.h, 400  
OTF2\_EvtReaderCallbacks\_SetThreadTeamBeginCallback, 423  
OTF2\_EvtReaderCallbacks.h, 400  
OTF2\_EvtReaderCallbacks\_SetThreadTeamEndCallback, 424  
OTF2\_EvtReaderCallbacks.h, 401  
OTF2\_EvtReaderCallbacks\_SetThreadWaitCallback, 425  
OTF2\_EvtReaderCallbacks.h, 401  
OTF2\_EvtReaderCallbacks\_SetUnknownCallback, 426  
OTF2\_EvtReaderCallbacks.h, 402  
OTF2\_EvtWriter.h  
OTF2\_EvtWriter\_BufferFlush, 409  
OTF2\_EvtWriter\_ClearRewindPoint, 410  
OTF2\_EvtWriter\_Enter, 410  
OTF2\_EvtWriter\_GetLocationID, 411  
OTF2\_EvtWriter\_GetNumberOfEvents, 411  
OTF2\_EvtWriter\_GetUserData, 412  
OTF2\_EvtWriter\_Leave, 412  
OTF2\_EvtWriter\_MeasurementOnOff, 412  
OTF2\_EvtWriter\_Metric, 413  
OTF2\_EvtWriter\_MpiCollectiveBegin, 414  
OTF2\_EvtWriter\_MpiCollectiveEnd, 414  
OTF2\_EvtWriter\_MpiIrecv, 415  
OTF2\_EvtWriter\_MpiIrecvRequest, 416  
OTF2\_EvtWriter\_MpiIsend, 416  
OTF2\_EvtWriter\_MpiIsendComplete, 417  
OTF2\_EvtWriter\_MpiRecv, 417  
OTF2\_EvtWriter\_MpiRequestCancelled, 418  
OTF2\_EvtWriter\_MpiRequestTest, 419  
OTF2\_EvtWriter\_MpiSend, 419  
OTF2\_EvtWriter\_OmpAcquireLock, 420  
OTF2\_EvtWriter\_OmpFork, 421  
OTF2\_EvtWriter\_OmpJoin, 421  
OTF2\_EvtWriter\_OmpReleaseLock, 422  
OTF2\_EvtWriter\_OmpTaskComplete, 423  
OTF2\_EvtWriter\_OmpTaskCreate, 423  
OTF2\_EvtWriter\_OmpTaskSwitch, 424  
OTF2\_EvtWriter\_ParameterInt, 425  
OTF2\_EvtWriter\_ParameterString, 425  
OTF2\_EvtWriter\_ParameterUnsignedInt, 426  
OTF2\_EvtWriter\_Rewind, 427  
OTF2\_EvtWriter\_RmaAcquireLock, 427  
OTF2\_EvtWriter\_RmaAtomic, 428  
OTF2\_EvtWriter\_RmaCollectiveBegin, 429  
OTF2\_EvtWriter\_RmaCollectiveEnd, 429  
OTF2\_EvtWriter\_RmaGet, 430  
OTF2\_EvtWriter\_RmaGroupSync, 430  
OTF2\_EvtWriter\_RmaOpCompleteBlocking, 431  
OTF2\_EvtWriter\_RmaOpCompleteNonBlocking, 432  
OTF2\_EvtWriter\_RmaOpCompleteRemote, 432  
OTF2\_EvtWriter\_RmaOpTest, 433  
OTF2\_EvtWriter\_RmaPut, 434  
OTF2\_EvtWriter\_RmaReleaseLock, 434

OTF2\_EvtWriter\_RmaRequestLock,      OTF2\_EvtWriter.h, 412  
     435      OTF2\_EvtWriter\_Leave  
 OTF2\_EvtWriter\_RmaSync, 436      OTF2\_EvtWriter.h, 412  
 OTF2\_EvtWriter\_RmaTryLock, 436 OTF2\_EvtWriter\_MeasurementOnOff  
 OTF2\_EvtWriter\_RmaWaitChange,      OTF2\_EvtWriter.h, 412  
     437      OTF2\_EvtWriter\_Metric  
 OTF2\_EvtWriter\_RmaWinCreate, 437      OTF2\_EvtWriter.h, 413  
 OTF2\_EvtWriter\_RmaWinDestroy,      OTF2\_EvtWriter\_MpiCollectiveBegin  
     438      OTF2\_EvtWriter.h, 414  
 OTF2\_EvtWriter\_SetLocationID, 439 OTF2\_EvtWriter\_MpiCollectiveEnd  
 OTF2\_EvtWriter\_SetUserData, 439      OTF2\_EvtWriter.h, 414  
 OTF2\_EvtWriter\_StoreRewindPoint, OTF2\_EvtWriter\_MpiIrecv  
     439      OTF2\_EvtWriter.h, 415  
 OTF2\_EvtWriter\_ThreadAcquireLock, OTF2\_EvtWriter\_MpiIrecvRequest  
     440      OTF2\_EvtWriter.h, 416  
 OTF2\_EvtWriter\_ThreadBegin, 440 OTF2\_EvtWriter\_MpiIsend  
 OTF2\_EvtWriter\_ThreadCreate, 441      OTF2\_EvtWriter.h, 416  
 OTF2\_EvtWriter\_ThreadEnd, 441      OTF2\_EvtWriter\_MpiIsendComplete  
 OTF2\_EvtWriter\_ThreadFork, 442      OTF2\_EvtWriter.h, 417  
 OTF2\_EvtWriter\_ThreadJoin, 443      OTF2\_EvtWriter\_MpiRecv  
 OTF2\_EvtWriter\_ThreadReleaseLock,      OTF2\_EvtWriter.h, 417  
     443      OTF2\_EvtWriter\_MpiRequestCancelled  
 OTF2\_EvtWriter\_ThreadTaskComplete,      OTF2\_EvtWriter.h, 418  
     444      OTF2\_EvtWriter\_MpiRequestTest  
 OTF2\_EvtWriter\_ThreadTaskCreate,      OTF2\_EvtWriter.h, 419  
     444      OTF2\_EvtWriter\_MpiSend  
 OTF2\_EvtWriter\_ThreadTaskSwitch,      OTF2\_EvtWriter.h, 419  
     445      OTF2\_EvtWriter\_OmpAcquireLock  
 OTF2\_EvtWriter\_ThreadTeamBegin,      OTF2\_EvtWriter.h, 420  
     446      OTF2\_EvtWriter\_OmpFork  
 OTF2\_EvtWriter\_ThreadTeamEnd,      OTF2\_EvtWriter.h, 421  
     446      OTF2\_EvtWriter\_OmpJoin  
 OTF2\_EvtWriter\_ThreadWait, 447      OTF2\_EvtWriter.h, 421  
 OTF2\_EvtWriter\_BufferFlush      OTF2\_EvtWriter\_OmpReleaseLock  
     OTF2\_EvtWriter.h, 409      OTF2\_EvtWriter.h, 422  
 OTF2\_EvtWriter\_ClearRewindPoint      OTF2\_EvtWriter\_OmpTaskComplete  
     OTF2\_EvtWriter.h, 410      OTF2\_EvtWriter.h, 423  
 OTF2\_EvtWriter\_Enter      OTF2\_EvtWriter\_OmpTaskCreate  
     OTF2\_EvtWriter.h, 410      OTF2\_EvtWriter.h, 423  
 OTF2\_EvtWriter\_GetLocationID      OTF2\_EvtWriter\_OmpTaskSwitch  
     OTF2\_EvtWriter.h, 411      OTF2\_EvtWriter.h, 424  
 OTF2\_EvtWriter\_GetNumberOfEvents      OTF2\_EvtWriter\_ParameterInt  
     OTF2\_EvtWriter.h, 411      OTF2\_EvtWriter.h, 425  
 OTF2\_EvtWriter\_GetUserData      OTF2\_EvtWriter\_ParameterString

---



## INDEX

---

OTF2\_EvtWriter.h, [425](#)  
OTF2\_EvtWriter\_ParameterUnsignedInt  
OTF2\_EvtWriter.h, [426](#)  
OTF2\_EvtWriter\_Rewind  
OTF2\_EvtWriter.h, [427](#)  
OTF2\_EvtWriter\_RmaAcquireLock  
OTF2\_EvtWriter.h, [427](#)  
OTF2\_EvtWriter\_RmaAtomic  
OTF2\_EvtWriter.h, [428](#)  
OTF2\_EvtWriter\_RmaCollectiveBegin  
OTF2\_EvtWriter.h, [429](#)  
OTF2\_EvtWriter\_RmaCollectiveEnd  
OTF2\_EvtWriter.h, [429](#)  
OTF2\_EvtWriter\_RmaGet  
OTF2\_EvtWriter.h, [430](#)  
OTF2\_EvtWriter\_RmaGroupSync  
OTF2\_EvtWriter.h, [430](#)  
OTF2\_EvtWriter\_RmaOpCompleteBlocking  
OTF2\_EvtWriter.h, [431](#)  
OTF2\_EvtWriter\_RmaOpCompleteNonBlocking  
OTF2\_EvtWriter.h, [432](#)  
OTF2\_EvtWriter\_RmaOpCompleteRemote  
OTF2\_EvtWriter.h, [432](#)  
OTF2\_EvtWriter\_RmaOpTest  
OTF2\_EvtWriter.h, [433](#)  
OTF2\_EvtWriter\_RmaPut  
OTF2\_EvtWriter.h, [434](#)  
OTF2\_EvtWriter\_RmaReleaseLock  
OTF2\_EvtWriter.h, [434](#)  
OTF2\_EvtWriter\_RmaRequestLock  
OTF2\_EvtWriter.h, [435](#)  
OTF2\_EvtWriter\_RmaSync  
OTF2\_EvtWriter.h, [436](#)  
OTF2\_EvtWriter\_RmaTryLock  
OTF2\_EvtWriter.h, [436](#)  
OTF2\_EvtWriter\_RmaWaitChange  
OTF2\_EvtWriter.h, [437](#)  
OTF2\_EvtWriter\_RmaWinCreate  
OTF2\_EvtWriter.h, [437](#)  
OTF2\_EvtWriter\_RmaWinDestroy  
OTF2\_EvtWriter.h, [438](#)  
OTF2\_EvtWriter\_SetLocationID  
OTF2\_EvtWriter.h, [439](#)  
OTF2\_EvtWriter\_SetUserData  
OTF2\_EvtWriter.h, [439](#)  
OTF2\_EvtWriter\_StoreRewindPoint  
OTF2\_EvtWriter.h, [439](#)  
OTF2\_EvtWriter\_ThreadAcquireLock  
OTF2\_EvtWriter.h, [440](#)  
OTF2\_EvtWriter\_ThreadBegin  
OTF2\_EvtWriter.h, [440](#)  
OTF2\_EvtWriter\_ThreadCreate  
OTF2\_EvtWriter.h, [441](#)  
OTF2\_EvtWriter\_ThreadEnd  
OTF2\_EvtWriter.h, [441](#)  
OTF2\_EvtWriter\_ThreadFork  
OTF2\_EvtWriter.h, [442](#)  
OTF2\_EvtWriter\_ThreadJoin  
OTF2\_EvtWriter.h, [443](#)  
OTF2\_EvtWriter\_ThreadReleaseLock  
OTF2\_EvtWriter.h, [443](#)  
OTF2\_EvtWriter\_ThreadTaskComplete  
OTF2\_EvtWriter.h, [444](#)  
OTF2\_EvtWriter\_ThreadTaskCreate  
OTF2\_EvtWriter.h, [444](#)  
OTF2\_EvtWriter\_ThreadTaskSwitch  
OTF2\_EvtWriter.h, [445](#)  
OTF2\_EvtWriter\_ThreadTeamBegin  
OTF2\_EvtWriter.h, [446](#)  
OTF2\_EvtWriter\_ThreadTeamEnd  
OTF2\_EvtWriter.h, [446](#)  
OTF2\_EvtWriter\_ThreadWait  
OTF2\_EvtWriter.h, [447](#)  
OTF2\_FileMode\_enum  
OTF2\_GeneralDefinitions.h, [456](#)  
OTF2\_FileSubstrate\_enum  
OTF2\_GeneralDefinitions.h, [456](#)  
OTF2\_FileType\_enum  
OTF2\_GeneralDefinitions.h, [457](#)  
OTF2\_FlushCallbacks, [123](#)  
OTF2\_FlushType\_enum  
OTF2\_GeneralDefinitions.h, [457](#)  
OTF2\_GeneralDefinitions.h  
OTF2\_CallbackCode, [455](#)  
OTF2\_Compression\_enum, [456](#)  
OTF2\_FileMode\_enum, [456](#)  
OTF2\_FileSubstrate\_enum, [456](#)  
OTF2\_FileType\_enum, [457](#)

OTF2\_FlushType\_enum, [457](#)  
 OTF2\_MappingType\_enum, [458](#)  
 OTF2\_Paradigm\_enum, [458](#)  
 OTF2\_ThumbnailType\_enum, [459](#)  
 OTF2\_Type\_enum, [460](#)  
 OTF2\_GlobalDefReader.h  
   OTF2\_GlobalDefReader\_ReadDefinitions, [462](#)  
   OTF2\_GlobalDefReader\_SetCallbacks, [462](#)  
 OTF2\_GlobalDefReader\_ReadDefinitions  
   OTF2\_GlobalDefReader.h, [462](#)  
 OTF2\_GlobalDefReader\_SetCallbacks  
   OTF2\_GlobalDefReader.h, [462](#)  
 OTF2\_GlobalDefReaderCallback\_Attribute  
   OTF2\_GlobalDefReaderCallbacks.h, [469](#)  
 OTF2\_GlobalDefReaderCallback\_Callpath  
   OTF2\_GlobalDefReaderCallbacks.h, [470](#)  
 OTF2\_GlobalDefReaderCallback\_Callsite  
   OTF2\_GlobalDefReaderCallbacks.h, [470](#)  
 OTF2\_GlobalDefReaderCallback\_CartCoordinate  
   OTF2\_GlobalDefReaderCallbacks.h, [471](#)  
 OTF2\_GlobalDefReaderCallback\_CartDimension  
   OTF2\_GlobalDefReaderCallbacks.h, [472](#)  
 OTF2\_GlobalDefReaderCallback\_CartTopology  
   OTF2\_GlobalDefReaderCallbacks.h, [472](#)  
 OTF2\_GlobalDefReaderCallback\_ClockProperties  
   OTF2\_GlobalDefReaderCallbacks.h, [473](#)  
 OTF2\_GlobalDefReaderCallback\_Comm  
   OTF2\_GlobalDefReaderCallbacks.h, [474](#)  
 OTF2\_GlobalDefReaderCallback\_Group  
   OTF2\_GlobalDefReaderCallbacks.h, [475](#)  
 OTF2\_GlobalDefReaderCallback\_Location  
   OTF2\_GlobalDefReaderCallbacks.h, [475](#)  
 OTF2\_GlobalDefReaderCallback\_LocationGroup  
   OTF2\_GlobalDefReaderCallbacks.h, [476](#)  
 OTF2\_GlobalDefReaderCallback\_LocationGroupProperty  
   OTF2\_GlobalDefReaderCallbacks.h, [477](#)  
 OTF2\_GlobalDefReaderCallback\_LocationProperty  
   OTF2\_GlobalDefReaderCallbacks.h, [477](#)  
 OTF2\_GlobalDefReaderCallback\_MetricClass  
   OTF2\_GlobalDefReaderCallbacks.h, [478](#)  
 OTF2\_GlobalDefReaderCallback\_MetricClassRecorder  
   OTF2\_GlobalDefReaderCallbacks.h, [479](#)  
 OTF2\_GlobalDefReaderCallback\_MetricInstance  
   OTF2\_GlobalDefReaderCallbacks.h, [479](#)  
 OTF2\_GlobalDefReaderCallback\_MetricMember  
   OTF2\_GlobalDefReaderCallbacks.h, [480](#)  
 OTF2\_GlobalDefReaderCallback\_Parameter  
   OTF2\_GlobalDefReaderCallbacks.h, [481](#)  
 OTF2\_GlobalDefReaderCallback\_Region  
   OTF2\_GlobalDefReaderCallbacks.h, [482](#)  
 OTF2\_GlobalDefReaderCallback\_RmaWin  
   OTF2\_GlobalDefReaderCallbacks.h, [483](#)  
 OTF2\_GlobalDefReaderCallback\_String  
   OTF2\_GlobalDefReaderCallbacks.h, [483](#)  
 OTF2\_GlobalDefReaderCallback\_SystemTreeNode  
   OTF2\_GlobalDefReaderCallbacks.h, [484](#)  
 OTF2\_GlobalDefReaderCallback\_SystemTreeNodeDomain  
   OTF2\_GlobalDefReaderCallbacks.h, [484](#)  
 OTF2\_GlobalDefReaderCallback\_SystemTreeNodeProperty  
   OTF2\_GlobalDefReaderCallbacks.h, [485](#)  
 OTF2\_GlobalDefReaderCallback\_Unknown

## INDEX

---

- OTF2\_GlobalDefReaderCallbacks.h, 486
- OTF2\_GlobalDefReaderCallbacks.h
  - OTF2\_GlobalDefReaderCallback\_-Attribute, 469
  - OTF2\_GlobalDefReaderCallback\_-Callpath, 470
  - OTF2\_GlobalDefReaderCallback\_-Callsite, 470
  - OTF2\_GlobalDefReaderCallback\_-CartCoordinate, 471
  - OTF2\_GlobalDefReaderCallback\_-CartDimension, 472
  - OTF2\_GlobalDefReaderCallback\_-CartTopology, 472
  - OTF2\_GlobalDefReaderCallback\_-ClockProperties, 473
  - OTF2\_GlobalDefReaderCallback\_-Comm, 474
  - OTF2\_GlobalDefReaderCallback\_-Group, 475
  - OTF2\_GlobalDefReaderCallback\_-Location, 475
  - OTF2\_GlobalDefReaderCallback\_-LocationGroup, 476
  - OTF2\_GlobalDefReaderCallback\_-LocationGroupProperty, 477
  - OTF2\_GlobalDefReaderCallback\_-LocationProperty, 477
  - OTF2\_GlobalDefReaderCallback\_-MetricClass, 478
  - OTF2\_GlobalDefReaderCallback\_-MetricClassRecorder, 479
  - OTF2\_GlobalDefReaderCallback\_-MetricInstance, 479
  - OTF2\_GlobalDefReaderCallback\_-MetricMember, 480
  - OTF2\_GlobalDefReaderCallback\_-Parameter, 481
  - OTF2\_GlobalDefReaderCallback\_-Region, 482
  - OTF2\_GlobalDefReaderCallback\_-RmaWin, 483
- OTF2\_GlobalDefReaderCallback\_-String, 483
- OTF2\_GlobalDefReaderCallback\_-SystemTreeNode, 484
- OTF2\_GlobalDefReaderCallback\_-SystemTreeNodeDomain, 484
- OTF2\_GlobalDefReaderCallback\_-SystemTreeNodeProperty, 485
- OTF2\_GlobalDefReaderCallback\_-Unknown, 486
- OTF2\_GlobalDefReaderCallbacks\_-Clear, 486
- OTF2\_GlobalDefReaderCallbacks\_-Delete, 486
- OTF2\_GlobalDefReaderCallbacks\_-New, 487
- OTF2\_GlobalDefReaderCallbacks\_-SetAttributeCallback, 487
- OTF2\_GlobalDefReaderCallbacks\_-SetCallpathCallback, 487
- OTF2\_GlobalDefReaderCallbacks\_-SetCallsiteCallback, 488
- OTF2\_GlobalDefReaderCallbacks\_-SetCartCoordinateCallback, 489
- OTF2\_GlobalDefReaderCallbacks\_-SetCartDimensionCallback, 489
- OTF2\_GlobalDefReaderCallbacks\_-SetCartTopologyCallback, 490
- OTF2\_GlobalDefReaderCallbacks\_-SetClockPropertiesCallback, 490
- OTF2\_GlobalDefReaderCallbacks\_-SetCommCallback, 491
- OTF2\_GlobalDefReaderCallbacks\_-SetGroupCallback, 491
- OTF2\_GlobalDefReaderCallbacks\_-SetLocationCallback, 492
- OTF2\_GlobalDefReaderCallbacks\_-SetLocationGroupCallback, 493
- OTF2\_GlobalDefReaderCallbacks\_-SetLocationGroupPropertyCallback, 493
- OTF2\_GlobalDefReaderCallbacks\_-SetLocationPropertyCallback, 494

## INDEX

---

OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks.h,  
     SetMetricClassCallback, [494](#) [488](#)  
 OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetCartCoordinateCallback  
     SetMetricClassRecorderCallback, OTF2\_GlobalDefReaderCallbacks.h,  
     [495](#) [489](#)  
 OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetCartDimensionCallback  
     SetMetricInstanceCallback, [496](#) OTF2\_GlobalDefReaderCallbacks.h,  
     OTF2\_GlobalDefReaderCallbacks\_- [489](#)  
     SetMetricMemberCallback, [496](#) OTF2\_GlobalDefReaderCallbacks\_SetCartTopologyCallback  
 OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks.h,  
     SetParameterCallback, [497](#) [490](#)  
 OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetClockPropertiesCallback  
     SetRegionCallback, [497](#) OTF2\_GlobalDefReaderCallbacks.h,  
     OTF2\_GlobalDefReaderCallbacks\_- [490](#)  
     SetRmaWinCallback, [498](#) OTF2\_GlobalDefReaderCallbacks\_SetCommCallback  
 OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks.h,  
     SetStringCallback, [499](#) [491](#)  
 OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetGroupCallback  
     SetSystemTreeNodeCallback, [499](#) OTF2\_GlobalDefReaderCallbacks.h,  
     OTF2\_GlobalDefReaderCallbacks\_- [491](#)  
     SetSystemTreeNodeDomainCallback, OTF2\_GlobalDefReaderCallbacks\_SetLocationCallback  
     [500](#) OTF2\_GlobalDefReaderCallbacks.h,  
     OTF2\_GlobalDefReaderCallbacks\_- [492](#)  
     SetSystemTreeNodePropertyCallback, OTF2\_GlobalDefReaderCallbacks\_SetLocationGroupCallback  
     [500](#) OTF2\_GlobalDefReaderCallbacks.h,  
     OTF2\_GlobalDefReaderCallbacks\_- [493](#)  
     SetUnknownCallback, [501](#) OTF2\_GlobalDefReaderCallbacks\_SetLocationGroupPropertyCallback  
 OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks.h,  
 OTF2\_GlobalDefReaderCallbacks\_Clear [493](#)  
     OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetLocationPropertyCallback  
     [486](#) OTF2\_GlobalDefReaderCallbacks.h,  
 OTF2\_GlobalDefReaderCallbacks\_Delete [494](#)  
     OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetMetricClassCallback  
     [486](#) OTF2\_GlobalDefReaderCallbacks.h,  
 OTF2\_GlobalDefReaderCallbacks\_New [494](#)  
     OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetMetricClassRecorderCallback  
     [487](#) OTF2\_GlobalDefReaderCallbacks.h,  
 OTF2\_GlobalDefReaderCallbacks\_SetAttributeCallback [495](#)  
     OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetMetricInstanceCallback  
     [487](#) OTF2\_GlobalDefReaderCallbacks.h,  
 OTF2\_GlobalDefReaderCallbacks\_SetCallpathCallback [496](#)  
     OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetMetricMemberCallback  
     [487](#) OTF2\_GlobalDefReaderCallbacks.h,  
 OTF2\_GlobalDefReaderCallbacks\_SetCallsiteCallback [496](#)

## INDEX

---

OTF2\_GlobalDefReaderCallbacks\_SetParameterCallback, 510  
OTF2\_GlobalDefReaderCallbacks.h, 497  
OTF2\_GlobalDefReaderCallbacks\_SetRegionCallback, 511  
OTF2\_GlobalDefReaderCallbacks.h, 497  
OTF2\_GlobalDefReaderCallbacks\_SetRmaWinCallback, 513  
OTF2\_GlobalDefReaderCallbacks.h, 498  
OTF2\_GlobalDefReaderCallbacks\_SetStringCallback, 513  
OTF2\_GlobalDefReaderCallbacks.h, 499  
OTF2\_GlobalDefReaderCallbacks\_SetSystemTreeNodeCallback, 514  
OTF2\_GlobalDefReaderCallbacks.h, 499  
OTF2\_GlobalDefReaderCallbacks\_SetSystemTreeNodeDomainCallback, 515  
OTF2\_GlobalDefReaderCallbacks.h, 500  
OTF2\_GlobalDefReaderCallbacks\_SetSystemTreeNodePropertyCallback, 516  
OTF2\_GlobalDefReaderCallbacks.h, 500  
OTF2\_GlobalDefReaderCallbacks\_SetUnknownCallback, 518  
OTF2\_GlobalDefReaderCallbacks.h, 501  
OTF2\_GlobalDefWriter.h, 519  
OTF2\_GlobalDefWriter\_GetNumberOfDefinitions, 505  
OTF2\_GlobalDefWriter\_GetNumberOfLocations, 506  
OTF2\_GlobalDefWriter\_WriteAttribute, 506  
OTF2\_GlobalDefWriter\_WriteCallpath, 507  
OTF2\_GlobalDefWriter\_WriteCallsite, 507  
OTF2\_GlobalDefWriter\_WriteCartCoordinate, 508  
OTF2\_GlobalDefWriter\_WriteCartDirection, 509  
OTF2\_GlobalDefWriter\_WriteCartTopology, 509  
OTF2\_GlobalDefWriter\_WriteClockPart, 510  
OTF2\_GlobalDefWriter\_WriteComm, 510  
OTF2\_GlobalDefWriter\_WriteGroup, 511  
OTF2\_GlobalDefWriter\_WriteLocation, 512  
OTF2\_GlobalDefWriter\_WriteLocationGroup, 513  
OTF2\_GlobalDefWriter\_WriteLocationGroupProperty, 513  
OTF2\_GlobalDefWriter\_WriteLocationProperty, 514  
OTF2\_GlobalDefWriter\_WriteMetricClass, 514  
OTF2\_GlobalDefWriter\_WriteMetricClassRecorder, 515  
OTF2\_GlobalDefWriter\_WriteMetricInstance, 516  
OTF2\_GlobalDefWriter\_WriteMetricMember, 516  
OTF2\_GlobalDefWriter\_WriteParameter, 518  
OTF2\_GlobalDefWriter\_WriteRegion, 518  
OTF2\_GlobalDefWriter\_WriteRmaWin, 519  
OTF2\_GlobalDefWriter\_WriteString, 520  
OTF2\_GlobalDefWriter\_WriteSystemTreeNode, 520  
OTF2\_GlobalDefWriter\_WriteSystemTreeNodeDomain, 521  
OTF2\_GlobalDefWriter\_WriteSystemTreeNodeProperty, 521  
OTF2\_GlobalDefWriter\_GetNumberOfDefinitions, 505  
OTF2\_GlobalDefWriter.h, 505  
OTF2\_GlobalDefWriter\_GetNumberOfLocations, 506  
OTF2\_GlobalDefWriter.h, 506  
OTF2\_GlobalDefWriter\_WriteAttribute, 506  
OTF2\_GlobalDefWriter.h, 506  
OTF2\_GlobalDefWriter\_WriteCallpath, 507  
OTF2\_GlobalDefWriter.h, 507  
OTF2\_GlobalDefWriter\_WriteCallsite, 507  
OTF2\_GlobalDefWriter.h, 507

OTF2\_GlobalDefWriter\_WriteCartCoordinate OTF2\_GlobalEvtReader\_HasEvent,  
 OTF2\_GlobalDefWriter.h, 508 523  
 OTF2\_GlobalDefWriter\_WriteCartDimension OTF2\_GlobalEvtReader\_ReadEvent,  
 OTF2\_GlobalDefWriter.h, 509 523  
 OTF2\_GlobalDefWriter\_WriteCartTopology OTF2\_GlobalEvtReader\_ReadEvents,  
 OTF2\_GlobalDefWriter.h, 509 524  
 OTF2\_GlobalDefWriter\_WriteClockProperties OTF2\_GlobalEvtReader\_SetCallbacks,  
 OTF2\_GlobalDefWriter.h, 510 524  
 OTF2\_GlobalDefWriter\_WriteComm OTF2\_GlobalEvtReader\_HasEvent  
 OTF2\_GlobalDefWriter.h, 510 OTF2\_GlobalEvtReader.h, 523  
 OTF2\_GlobalDefWriter\_WriteGroup OTF2\_GlobalEvtReader\_ReadEvent  
 OTF2\_GlobalDefWriter.h, 511 OTF2\_GlobalEvtReader.h, 523  
 OTF2\_GlobalDefWriter\_WriteLocation OTF2\_GlobalEvtReader\_ReadEvents  
 OTF2\_GlobalDefWriter.h, 512 OTF2\_GlobalEvtReader.h, 524  
 OTF2\_GlobalDefWriter\_WriteLocationGroup OTF2\_GlobalEvtReader\_SetCallbacks  
 OTF2\_GlobalDefWriter.h, 513 OTF2\_GlobalEvtReader.h, 524  
 OTF2\_GlobalDefWriter\_WriteLocationGroupProperty OTF2\_GlobalEvtReaderCallback\_BufferFlush  
 OTF2\_GlobalDefWriter.h, 513 OTF2\_GlobalEvtReaderCallbacks.h,  
 538  
 OTF2\_GlobalDefWriter\_WriteLocationProperty OTF2\_GlobalEvtReaderCallback\_Enter  
 OTF2\_GlobalDefWriter.h, 514 OTF2\_GlobalEvtReaderCallbacks.h,  
 OTF2\_GlobalDefWriter\_WriteMetricClass 538  
 OTF2\_GlobalDefWriter.h, 514 OTF2\_GlobalEvtReaderCallback\_Leave  
 OTF2\_GlobalDefWriter\_WriteMetricClassRecorder OTF2\_GlobalEvtReaderCallbacks.h,  
 OTF2\_GlobalDefWriter.h, 515 539  
 OTF2\_GlobalDefWriter\_WriteMetricInstance OTF2\_GlobalEvtReaderCallback\_MeasurementOnOff  
 OTF2\_GlobalDefWriter.h, 516 OTF2\_GlobalEvtReaderCallbacks.h,  
 OTF2\_GlobalDefWriter\_WriteMetricMember 540  
 OTF2\_GlobalDefWriter.h, 516 OTF2\_GlobalEvtReaderCallback\_Metric  
 OTF2\_GlobalDefWriter\_WriteParameter OTF2\_GlobalEvtReaderCallbacks.h,  
 OTF2\_GlobalDefWriter.h, 518 540  
 OTF2\_GlobalDefWriter\_WriteRegion OTF2\_GlobalEvtReaderCallback\_MpiCollectiveBegin  
 OTF2\_GlobalDefWriter.h, 518 OTF2\_GlobalEvtReaderCallbacks.h,  
 OTF2\_GlobalDefWriter\_WriteRmaWin 541  
 OTF2\_GlobalDefWriter.h, 519 OTF2\_GlobalEvtReaderCallback\_MpiCollectiveEnd  
 OTF2\_GlobalDefWriter\_WriteString OTF2\_GlobalEvtReaderCallbacks.h,  
 OTF2\_GlobalDefWriter.h, 520 542  
 OTF2\_GlobalDefWriter\_WriteSystemTreeNode OTF2\_GlobalEvtReaderCallback\_MpiRecv  
 OTF2\_GlobalDefWriter.h, 520 OTF2\_GlobalEvtReaderCallbacks.h,  
 OTF2\_GlobalDefWriter\_WriteSystemTreeNodeDone 541  
 OTF2\_GlobalDefWriter.h, 521 OTF2\_GlobalEvtReaderCallback\_MpiRecvRequest  
 OTF2\_GlobalDefWriter\_WriteSystemTreeNodeProperty OTF2\_GlobalEvtReaderCallbacks.h,  
 OTF2\_GlobalDefWriter.h, 521 543  
 OTF2\_GlobalEvtReader.h OTF2\_GlobalEvtReaderCallback\_MpiIsend



## INDEX

---

OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_ParameterUnsignedInt  
544 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_MpiIsendComplete 554  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaAcquireLock  
545 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_MpiRecv 555  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaAtomic  
545 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_MpiRequestCancelled 555  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaCollectiveBegin  
546 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_MpiRequestTest 556  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaCollectiveEnd  
547 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_MpiSend 557  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaGet  
547 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_OmpAcquireLock 558  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaGroupSync  
548 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_OmpFork 558  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteBlocking  
549 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_OmpJoin 559  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteNonBlocking  
549 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_OmpReleaseLock 560  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteRemote  
550 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_OmpTaskComplete 561  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaOpTest  
551 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_OmpTaskCreate 561  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaPut  
551 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_OmpTaskSwitch 562  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaReleaseLock  
552 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_ParameterInt 563  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaRequestLock  
553 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallback\_ParameterString 563  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallback\_RmaSync  
553

---





## INDEX

---

- OTF2\_GlobalEvtReaderCallback\_-  
  OmpAcquireLock, [548](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  OmpFork, [549](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  OmpJoin, [549](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  OmpReleaseLock, [550](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  OmpTaskComplete, [551](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  OmpTaskCreate, [551](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  OmpTaskSwitch, [552](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ParameterInt, [553](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ParameterString, [553](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ParameterUnsignedInt, [554](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaAcquireLock, [555](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaAtomic, [555](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaCollectiveBegin, [556](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaCollectiveEnd, [557](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaGet, [558](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaGroupSync, [558](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaOpCompleteBlocking, [559](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaOpCompleteNonBlocking,  
    [560](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaOpCompleteRemote, [561](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaOpTest, [561](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaPut, [562](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaReleaseLock, [563](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaRequestLock, [563](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaSync, [564](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaTryLock, [565](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaWaitChange, [565](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaWinCreate, [566](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  RmaWinDestroy, [567](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadAcquireLock, [567](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadBegin, [568](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadCreate, [569](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadEnd, [569](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadFork, [570](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadJoin, [571](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadReleaseLock, [571](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadTaskComplete, [572](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadTaskCreate, [573](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadTaskSwitch, [573](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadTeamBegin, [574](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadTeamEnd, [575](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  ThreadWait, [575](#)
- OTF2\_GlobalEvtReaderCallback\_-  
  Unknown, [576](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
  Clear, [577](#)

- OTF2\_GlobalEvtReaderCallbacks\_-  
Delete, [577](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
New, [577](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetBufferFlushCallback, [577](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetEnterCallback, [578](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetLeaveCallback, [579](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMeasurementOnOffCallback,  
[579](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMetricCallback, [580](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiCollectiveBeginCallback,  
[580](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiCollectiveEndCallback,  
[581](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiIrecvCallback, [582](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiIrecvRequestCallback, [582](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiIsendCallback, [583](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiIsendCompleteCallback,  
[583](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiRecvCallback, [584](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiRequestCancelledCallback,  
[585](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiRequestTestCallback, [585](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiSendCallback, [586](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetOmpAcquireLockCallback,  
[586](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetOmpForkCallback, [587](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetOmpJoinCallback, [588](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetOmpReleaseLockCallback, [588](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetOmpTaskCompleteCallback,  
[589](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetOmpTaskCreateCallback, [589](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetOmpTaskSwitchCallback, [590](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetParameterIntCallback, [590](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetParameterStringCallback, [591](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetParameterUnsignedIntCallback,  
[592](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaAcquireLockCallback, [592](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaAtomicCallback, [593](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaCollectiveBeginCallback,  
[593](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaCollectiveEndCallback,  
[594](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaGetCallback, [595](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaGroupSyncCallback, [595](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaOpCompleteBlockingCallback,  
[596](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaOpCompleteNonBlockingCallback,  
[597](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaOpCompleteRemoteCallback,  
[597](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaOpTestCallback, [598](#)

## INDEX

---

- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaPutCallback, [598](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaReleaseLockCallback, [599](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaRequestLockCallback, [599](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaSyncCallback, [600](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaTryLockCallback, [601](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaWaitChangeCallback, [601](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaWinCreateCallback, [602](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaWinDestroyCallback, [602](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadAcquireLockCallback,  
[603](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadBeginCallback, [604](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadCreateCallback, [604](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadEndCallback, [605](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadForkCallback, [605](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadJoinCallback, [606](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadReleaseLockCallback,  
[607](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadTaskCompleteCallback,  
[607](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadTaskCreateCallback,  
[608](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadTaskSwitchCallback,  
[608](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadTeamBeginCallback,  
[609](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadTeamEndCallback, [610](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadWaitCallback, [610](#)
- OTF2\_GlobalEvtReaderCallbacks\_-  
SetUnknownCallback, [611](#)
- OTF2\_GlobalEvtReaderCallbacks\_Clear  
OTF2\_GlobalEvtReaderCallbacks.h,  
[577](#)
- OTF2\_GlobalEvtReaderCallbacks\_Delete  
OTF2\_GlobalEvtReaderCallbacks.h,  
[577](#)
- OTF2\_GlobalEvtReaderCallbacks\_New  
OTF2\_GlobalEvtReaderCallbacks.h,  
[577](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetBufferFlushCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[577](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetEnterCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[578](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetLeaveCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[579](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetMeasurementOnOffCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[579](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetMetricCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[580](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetMpiCollectiveBeginCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[580](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetMpiCollectiveEndCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[581](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetMpiIrecvCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[582](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetMpiIrecvRequestCallback  
OTF2\_GlobalEvtReaderCallbacks.h,  
[582](#)
- OTF2\_GlobalEvtReaderCallbacks\_SetMpiIsendCallback

## INDEX

OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetParameterUnsignedIntCallback  
583 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpiIsendCompleteCallback  
592 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaAcquireLockCallback  
583 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpiRecvCallback  
592 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaAtomicCallback  
584 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpiRequestCancelledCallback  
593 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaCollectiveBeginCallback  
585 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpiRequestTestCallback  
593 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaCollectiveEndCallback  
585 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpiSendCallback  
594 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaGetCallback  
586 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetOmpAcquireLockCallback  
595 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaGroupSyncCallback  
586 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetOmpForkCallback  
595 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteBlockingCallback  
587 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetOmpJoinCallback  
596 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteNonBlockingCallback  
588 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetOmpReleaseLockCallback  
597 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteRemoteCallback  
588 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskCompleteCallback  
597 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpTestCallback  
589 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskCreateCallback  
598 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaPutCallback  
589 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaRequestLockCallback  
598 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaReleaseLockCallback  
590 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetParameterIntCallback  
599 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaRequestLockCallback  
590 OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetParameterStringCallback  
599 OTF2\_GlobalEvtReaderCallbacks.h,  
591 OTF2\_GlobalEvtReaderCallbacks\_SetRmaSyncCallback

## INDEX

---

OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadTeamBeginCallback  
600 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaTryLockCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadTeamEndCallback  
601 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaWaitChangeCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadWaitCallback  
601 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaWinCreateCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetUnknownCallback  
602 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaWinDestroyCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalSnapReader.h  
602 OTF2\_GlobalSnapReader\_ReadSnapshots,  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadAcquireLockCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalSnapReader\_SetCallbacks,  
603 OTF2\_GlobalSnapReaderCallback\_Enter  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadBeginCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalSnapReader\_ReadSnapshots  
604 OTF2\_GlobalSnapReader.h, 612  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadCreateCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalSnapReader\_SetCallbacks  
OTF2\_GlobalSnapReader.h, 613  
604 OTF2\_GlobalSnapReaderCallback\_Enter  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadEndCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalSnapReaderCallbacks.h,  
619 OTF2\_GlobalSnapReaderCallback\_MeasurementOnOff  
605 OTF2\_GlobalSnapReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadForkCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 620  
605 OTF2\_GlobalSnapReaderCallback\_Metric  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadJoinCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 620  
606 OTF2\_GlobalSnapReaderCallback\_MpiCollectiveBegin  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadReleaseLockCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 621  
607 OTF2\_GlobalSnapReaderCallback\_MpiCollectiveEnd  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskCompleteCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 622  
607 OTF2\_GlobalSnapReaderCallback\_MpiIrecv  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskCreateCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 623  
608 OTF2\_GlobalSnapReaderCallback\_MpiIrecvRequest  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskWaitCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 624  
608 OTF2\_GlobalSnapReaderCallback\_MpiIsend

---



---

[OTF2\\_GlobalSnapReaderCallbacks.h](#)  
[625](#)

[OTF2\\_GlobalSnapReaderCallback\\_MpiIsendComplete](#), [619](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[626](#) [MeasurementOnOff](#), [620](#)

[OTF2\\_GlobalSnapReaderCallback\\_MpiRecv](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [Metric](#), [620](#)  
[626](#)

[OTF2\\_GlobalSnapReaderCallback\\_MpiSend](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [MpiCollectiveBegin](#), [621](#)  
[627](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[MpiCollectiveEnd](#), [622](#)

[OTF2\\_GlobalSnapReaderCallback\\_OmpAcquireLock](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [MpiIrecv](#), [623](#)  
[628](#)

[OTF2\\_GlobalSnapReaderCallback\\_OmpFork](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [MpiIrecvRequest](#), [624](#)  
[629](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[MpiIsend](#), [625](#)

[OTF2\\_GlobalSnapReaderCallback\\_OmpTaskCreate](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [MpiIsendComplete](#), [626](#)  
[629](#)

[OTF2\\_GlobalSnapReaderCallback\\_OmpTaskSwitch](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [MpiRecv](#), [626](#)  
[630](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[MpiSend](#), [627](#)

[OTF2\\_GlobalSnapReaderCallback\\_ParameterInt](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [OmpAcquireLock](#), [628](#)  
[631](#)

[OTF2\\_GlobalSnapReaderCallback\\_ParameterString](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [OmpFork](#), [629](#)  
[632](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OmpTaskCreate](#), [629](#)

[OTF2\\_GlobalSnapReaderCallback\\_ParameterUnsignedInt](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [OmpTaskSwitch](#), [630](#)  
[632](#)

[OTF2\\_GlobalSnapReaderCallback\\_SnapshotEnd](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [ParameterInt](#), [631](#)  
[633](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[ParameterString](#), [632](#)

[OTF2\\_GlobalSnapReaderCallback\\_SnapshotStart](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [ParameterUnsignedInt](#), [632](#)  
[634](#)

[OTF2\\_GlobalSnapReaderCallback\\_Unknown](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [SnapshotEnd](#), [633](#)  
[635](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[SnapshotStart](#), [634](#)

[OTF2\\_GlobalSnapReaderCallbacks](#) [OTF2\\_GlobalSnapReaderCallback\\_-](#)  
[OTF2\\_GlobalSnapReaderCallbacks.h](#), [Unknown](#), [635](#)  
[635](#)

---

## INDEX

---

- OTF2\_GlobalSnapReaderCallbacks, 635
- OTF2\_GlobalSnapReaderCallbacks\_- Clear, 636
- OTF2\_GlobalSnapReaderCallbacks\_- Delete, 636
- OTF2\_GlobalSnapReaderCallbacks\_- New, 636
- OTF2\_GlobalSnapReaderCallbacks\_- SetEnterCallback, 637
- OTF2\_GlobalSnapReaderCallbacks\_- SetMeasurementOnOffCallback, 637
- OTF2\_GlobalSnapReaderCallbacks\_- SetMetricCallback, 638
- OTF2\_GlobalSnapReaderCallbacks\_- SetMpiCollectiveBeginCallback, 638
- OTF2\_GlobalSnapReaderCallbacks\_- SetMpiCollectiveEndCallback, 639
- OTF2\_GlobalSnapReaderCallbacks\_- SetMpiIrecvCallback, 639
- OTF2\_GlobalSnapReaderCallbacks\_- SetMpiIrecvRequestCallback, 640
- OTF2\_GlobalSnapReaderCallbacks\_- SetMpiIsendCallback, 641
- OTF2\_GlobalSnapReaderCallbacks\_- SetMpiIsendCompleteCallback, 641
- OTF2\_GlobalSnapReaderCallbacks\_- SetMpiRecvCallback, 642
- OTF2\_GlobalSnapReaderCallbacks\_- SetMpiSendCallback, 642
- OTF2\_GlobalSnapReaderCallbacks\_- SetOmpAcquireLockCallback, 643
- OTF2\_GlobalSnapReaderCallbacks\_- SetOmpForkCallback, 643
- OTF2\_GlobalSnapReaderCallbacks\_- SetOmpTaskCreateCallback, 644
- OTF2\_GlobalSnapReaderCallbacks\_- SetOmpTaskSwitchCallback, 645
- OTF2\_GlobalSnapReaderCallbacks\_- SetParameterIntCallback, 645
- OTF2\_GlobalSnapReaderCallbacks\_- SetParameterStringCallback, 646
- OTF2\_GlobalSnapReaderCallbacks\_- SetParameterUnsignedIntCallback, 646
- OTF2\_GlobalSnapReaderCallbacks\_- SetSnapshotEndCallback, 647
- OTF2\_GlobalSnapReaderCallbacks\_- SetSnapshotStartCallback, 648
- OTF2\_GlobalSnapReaderCallbacks\_- SetUnknownCallback, 648
- OTF2\_GlobalSnapReaderCallbacks\_Clear
- OTF2\_GlobalSnapReaderCallbacks.h, 636
- OTF2\_GlobalSnapReaderCallbacks\_Delete
- OTF2\_GlobalSnapReaderCallbacks.h, 636
- OTF2\_GlobalSnapReaderCallbacks\_New
- OTF2\_GlobalSnapReaderCallbacks.h, 636
- OTF2\_GlobalSnapReaderCallbacks\_SetEnterCallback
- OTF2\_GlobalSnapReaderCallbacks.h, 637
- OTF2\_GlobalSnapReaderCallbacks\_SetMeasurementOnOffCallback
- OTF2\_GlobalSnapReaderCallbacks.h, 637
- OTF2\_GlobalSnapReaderCallbacks\_SetMetricCallback
- OTF2\_GlobalSnapReaderCallbacks.h, 638
- OTF2\_GlobalSnapReaderCallbacks\_SetMpiCollectiveBeginCallback
- OTF2\_GlobalSnapReaderCallbacks.h, 638
- OTF2\_GlobalSnapReaderCallbacks\_SetMpiCollectiveEndCallback
- OTF2\_GlobalSnapReaderCallbacks.h, 639
- OTF2\_GlobalSnapReaderCallbacks\_SetMpiIrecvCallback
- OTF2\_GlobalSnapReaderCallbacks.h, 639
- OTF2\_GlobalSnapReaderCallbacks\_SetMpiIrecvRequestCallback
- OTF2\_GlobalSnapReaderCallbacks.h, 640
- OTF2\_GlobalSnapReaderCallbacks\_SetMpiIsendCallback

OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_Definitions.h, 201  
 641 OTF2\_IdMap  
 OTF2\_GlobalSnapReaderCallbacks\_SetMpiReaderCallback, 650  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h  
 641 OTF2\_IdMap, 650  
 OTF2\_GlobalSnapReaderCallbacks\_SetMpiReaderCallback, 651  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_Clear, 651  
 642 OTF2\_IdMap\_Create, 652  
 OTF2\_GlobalSnapReaderCallbacks\_SetMpiReaderCallback, 652  
 OTF2\_GlobalSnapReaderCallbacks.h, 652  
 642 OTF2\_IdMap\_CreateFromUint32Array,  
 OTF2\_GlobalSnapReaderCallbacks\_SetOmpAcquireLockCallback, 652  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_Free, 653  
 643 OTF2\_IdMap\_GetGlobalId, 653  
 OTF2\_GlobalSnapReaderCallbacks\_SetOmpForAllCallback, 654  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_GetGlobalIdSave, 654  
 643 OTF2\_IdMap\_GetMode, 654  
 OTF2\_IdMap\_GetSize, 654  
 OTF2\_GlobalSnapReaderCallbacks\_SetOmpTaskCallback, 655  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMapMode, 650  
 644 OTF2\_IdMapMode\_enum, 651  
 OTF2\_GlobalSnapReaderCallbacks\_SetOmpTaskCallback, 651  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h, 651  
 645 OTF2\_IdMap\_Clear  
 OTF2\_GlobalSnapReaderCallbacks\_SetParameterCallback, 651  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_Create  
 645 OTF2\_IdMap.h, 652  
 OTF2\_GlobalSnapReaderCallbacks\_SetParameterCallback, 652  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h, 652  
 646 OTF2\_IdMap\_CreateFromUint32Array  
 OTF2\_GlobalSnapReaderCallbacks\_SetParameterCallback, 652  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h, 652  
 646 OTF2\_IdMap\_CreateFromUint64Array  
 OTF2\_GlobalSnapReaderCallbacks\_SetParameterCallback, 653  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_Free  
 646 OTF2\_IdMap.h, 653  
 OTF2\_GlobalSnapReaderCallbacks\_SetSnapshotsCallback, 653  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h, 653  
 647 OTF2\_IdMap\_GetGlobalId  
 OTF2\_GlobalSnapReaderCallbacks\_SetSnapshotsCallback, 654  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_GetMode  
 648 OTF2\_IdMap.h, 654  
 OTF2\_GlobalSnapReaderCallbacks\_SetUnpackCallback, 654  
 OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h, 654  
 648 OTF2\_IdMap\_Traverse  
 OTF2\_GroupFlag\_enum, OTF2\_IdMap.h, 655  
 OTF2\_Definitions.h, 200 OTF2\_IdMapMode  
 OTF2\_GroupType\_enum, OTF2\_IdMap.h, 650

---



## INDEX

---

- OTF2\_IdMapMode\_enum
  - OTF2\_IdMap.h, [651](#)
- OTF2\_LocationGroupType\_enum
  - OTF2\_Definitions.h, [202](#)
- OTF2\_LocationType\_enum
  - OTF2\_Definitions.h, [202](#)
- OTF2\_LockType\_enum
  - OTF2\_Events.h, [273](#)
- OTF2\_MappingType\_enum
  - OTF2\_GeneralDefinitions.h, [458](#)
- OTF2\_Marker.h
  - OTF2\_MarkerScope\_enum, [656](#)
  - OTF2\_MarkerSeverity\_enum, [657](#)
- OTF2\_MarkerReader.h
  - OTF2\_MarkerReader\_ReadMarkers, [658](#)
  - OTF2\_MarkerReader\_SetCallbacks, [658](#)
- OTF2\_MarkerReader\_ReadMarkers
  - OTF2\_MarkerReader.h, [658](#)
- OTF2\_MarkerReader\_SetCallbacks
  - OTF2\_MarkerReader.h, [658](#)
- OTF2\_MarkerReaderCallback\_DefMarker
  - OTF2\_MarkerReaderCallbacks.h, [661](#)
- OTF2\_MarkerReaderCallback\_Marker
  - OTF2\_MarkerReaderCallbacks.h, [661](#)
- OTF2\_MarkerReaderCallback\_Unknown
  - OTF2\_MarkerReaderCallbacks.h, [662](#)
- OTF2\_MarkerReaderCallbacks.h
  - OTF2\_MarkerReaderCallback\_DefMarker, [661](#)
  - OTF2\_MarkerReaderCallback\_Marker, [661](#)
  - OTF2\_MarkerReaderCallback\_Unknown, [662](#)
  - OTF2\_MarkerReaderCallbacks\_Clear, [662](#)
  - OTF2\_MarkerReaderCallbacks\_Delete, [662](#)
  - OTF2\_MarkerReaderCallbacks\_New, [663](#)
  - OTF2\_MarkerReaderCallbacks\_SetDefMarkerCallback, [663](#)
- OTF2\_MarkerReaderCallbacks\_SetMarkerCallback,
  - [664](#)
- OTF2\_MarkerReaderCallbacks\_SetUnknownCallback,
  - [664](#)
- OTF2\_MarkerReaderCallbacks\_Clear
  - OTF2\_MarkerReaderCallbacks.h, [662](#)
- OTF2\_MarkerReaderCallbacks\_Delete
  - OTF2\_MarkerReaderCallbacks.h, [662](#)
- OTF2\_MarkerReaderCallbacks\_New
  - OTF2\_MarkerReaderCallbacks.h, [663](#)
- OTF2\_MarkerReaderCallbacks\_SetDefMarkerCallback
  - OTF2\_MarkerReaderCallbacks.h, [663](#)
- OTF2\_MarkerReaderCallbacks\_SetMarkerCallback
  - OTF2\_MarkerReaderCallbacks.h, [664](#)
- OTF2\_MarkerReaderCallbacks\_SetUnknownCallback
  - OTF2\_MarkerReaderCallbacks.h, [664](#)
- OTF2\_MarkerScope\_enum
  - OTF2\_Marker.h, [656](#)
- OTF2\_MarkerSeverity\_enum
  - OTF2\_Marker.h, [657](#)
- OTF2\_MarkerWriter.h
  - OTF2\_MarkerWriter\_WriteDefMarker, [666](#)
  - OTF2\_MarkerWriter\_WriteMarker, [666](#)
- OTF2\_MarkerWriter\_WriteDefMarker
  - OTF2\_MarkerWriter.h, [666](#)
- OTF2\_MarkerWriter\_WriteMarker
  - OTF2\_MarkerWriter.h, [666](#)
- OTF2\_MeasurementMode\_enum
  - OTF2\_Events.h, [273](#)
- OTF2\_MemoryAllocate
  - Memory pooling for OTF2, [94](#)
- OTF2\_MemoryCallbacks,
  - [124](#)
- OTF2\_MemoryFreeAll
  - Memory pooling for OTF2, [94](#)
- OTF2\_MetricBase\_enum
  - OTF2\_Definitions.h, [202](#)
- OTF2\_MetricMode\_enum
  - OTF2\_Definitions.h, [203](#)
- OTF2\_MetricOccurrence\_enum
  - OTF2\_Definitions.h, [203](#)
- OTF2\_MarkerMode\_enum
  - OTF2\_Definitions.h, [204](#)

- OTF2\_MetricTiming\_enum
  - OTF2\_Definitions.h, [204](#)
- OTF2\_MetricType\_enum
  - OTF2\_Definitions.h, [205](#)
- OTF2\_MetricValue\_union, [125](#)
- OTF2\_MetricValueProperty\_enum
  - OTF2\_Definitions.h, [205](#)
- OTF2\_MPI\_Archive\_SetCollectiveCallbacks
  - OTF2\_MPI\_Collectives.h, [669](#)
- OTF2\_MPI\_Archive\_SetCollectiveCallbacksSplit
  - OTF2\_MPI\_Collectives.h, [669](#)
- OTF2\_MPI\_Collectives.h
  - OTF2\_MPI\_Archive\_SetCollectiveCallbacks, [669](#)
  - OTF2\_MPI\_Archive\_SetCollectiveCallbacksSplit, [669](#)
  - OTF2\_MPI\_Reader\_SetCollectiveCallbacks, [670](#)
- OTF2\_MPI\_Reader\_SetCollectiveCallbacks
  - OTF2\_MPI\_Collectives.h, [670](#)
- OTF2\_MPI\_UserData, [125](#)
- OTF2\_Paradigm\_enum
  - OTF2\_GeneralDefinitions.h, [458](#)
- OTF2\_ParameterType\_enum
  - OTF2\_Definitions.h, [206](#)
- OTF2\_PostFlushCallback
  - Controlling OTF2 flush behavior in writing mode, [92](#)
- OTF2\_PreFlushCallback
  - Controlling OTF2 flush behavior in writing mode, [93](#)
- OTF2\_Reader.h
  - OTF2\_Reader\_Close, [676](#)
  - OTF2\_Reader\_CloseDefFiles, [676](#)
  - OTF2\_Reader\_CloseDefReader, [677](#)
  - OTF2\_Reader\_CloseEvtFiles, [677](#)
  - OTF2\_Reader\_CloseEvtReader, [677](#)
  - OTF2\_Reader\_CloseGlobalDefReader, [678](#)
  - OTF2\_Reader\_CloseGlobalEvtReader, [678](#)
  - OTF2\_Reader\_CloseGlobalSnapReader, [679](#)
  - OTF2\_Reader\_CloseMarkerReader, [679](#)
  - OTF2\_Reader\_CloseMarkerWriter, [679](#)
  - OTF2\_Reader\_CloseSnapFiles, [680](#)
  - OTF2\_Reader\_CloseSnapReader, [680](#)
  - OTF2\_Reader\_CloseThumbReader, [681](#)
  - OTF2\_Reader\_GetBoolProperty, [681](#)
  - OTF2\_Reader\_GetChunkSize, [682](#)
  - OTF2\_Reader\_GetCompression, [682](#)
  - OTF2\_Reader\_GetCreator, [682](#)
  - OTF2\_Reader\_GetDefReader, [683](#)
  - OTF2\_Reader\_GetDescription, [683](#)
  - OTF2\_Reader\_GetEvtReader, [683](#)
  - OTF2\_Reader\_GetFileSubstrate, [684](#)
  - OTF2\_Reader\_GetGlobalDefReader, [684](#)
  - OTF2\_Reader\_GetGlobalEvtReader, [684](#)
  - OTF2\_Reader\_GetGlobalSnapReader, [685](#)
  - OTF2\_Reader\_GetMachineName, [685](#)
  - OTF2\_Reader\_GetMarkerReader, [685](#)
  - OTF2\_Reader\_GetMarkerWriter, [686](#)
  - OTF2\_Reader\_GetNumberOfGlobalDefinitions, [686](#)
  - OTF2\_Reader\_GetNumberOfLocations, [687](#)
  - OTF2\_Reader\_GetNumberOfSnapshots, [687](#)
  - OTF2\_Reader\_GetNumberOfThumbnails, [687](#)
  - OTF2\_Reader\_GetProperty, [688](#)
  - OTF2\_Reader\_GetPropertyNames, [688](#)
  - OTF2\_Reader\_GetSnapReader, [689](#)
  - OTF2\_Reader\_GetThumbReader, [689](#)
  - OTF2\_Reader\_GetTraceId, [689](#)
  - OTF2\_Reader\_GetVersion, [690](#)
  - OTF2\_Reader\_HasGlobalEvent, [690](#)
  - OTF2\_Reader\_Open, [691](#)
  - OTF2\_Reader\_OpenDefFiles, [691](#)
  - OTF2\_Reader\_OpenEvtFiles, [691](#)
  - OTF2\_Reader\_OpenSnapFiles, [692](#)

## INDEX

---

OTF2\_Reader\_ReadAllGlobalDefinitions, OTF2\_Reader\_SetSerialCollectiveCallbacks,  
692 704  
OTF2\_Reader\_ReadAllGlobalEvents, OTF2\_Reader\_Close  
693 OTF2\_Reader.h, 676  
OTF2\_Reader\_ReadAllGlobalSnapshots, OTF2\_Reader\_CloseDefFiles  
693 OTF2\_Reader.h, 676  
OTF2\_Reader\_ReadAllLocalDefinitions, OTF2\_Reader\_CloseDefReader  
693 OTF2\_Reader.h, 677  
OTF2\_Reader\_ReadAllLocalEvents, OTF2\_Reader\_CloseEvtFiles  
694 OTF2\_Reader.h, 677  
OTF2\_Reader\_ReadAllLocalSnapshots, OTF2\_Reader\_CloseEvtReader  
694 OTF2\_Reader.h, 677  
OTF2\_Reader\_ReadAllMarkers, 695 OTF2\_Reader\_CloseGlobalDefReader  
OTF2\_Reader\_ReadGlobalDefinitions, OTF2\_Reader.h, 678  
695 OTF2\_Reader\_CloseGlobalEvtReader  
OTF2\_Reader\_ReadGlobalEvent, 696 OTF2\_Reader.h, 678  
OTF2\_Reader\_ReadGlobalEvents, 696 OTF2\_Reader\_CloseGlobalSnapReader  
OTF2\_Reader\_ReadGlobalSnapshots, OTF2\_Reader.h, 679  
696 OTF2\_Reader\_CloseMarkerReader  
OTF2\_Reader\_ReadLocalDefinitions, OTF2\_Reader.h, 679  
697 OTF2\_Reader\_CloseMarkerWriter  
OTF2\_Reader\_ReadLocalEvents, 697 OTF2\_Reader.h, 679  
OTF2\_Reader\_ReadLocalEventsBackward, OTF2\_Reader\_CloseSnapFiles  
698 OTF2\_Reader.h, 680  
OTF2\_Reader\_ReadLocalSnapshots, OTF2\_Reader\_CloseSnapReader  
698 OTF2\_Reader.h, 680  
OTF2\_Reader\_ReadMarkers, 699 OTF2\_Reader\_CloseThumbReader  
OTF2\_Reader\_RegisterDefCallbacks, OTF2\_Reader.h, 681  
699 OTF2\_Reader\_GetBoolProperty  
OTF2\_Reader\_RegisterEvtCallbacks, OTF2\_Reader.h, 681  
700 OTF2\_Reader\_GetChunkSize  
OTF2\_Reader\_RegisterGlobalDefCallbacks, OTF2\_Reader.h, 682  
700 OTF2\_Reader\_GetCompression  
OTF2\_Reader\_RegisterGlobalEvtCallbacks, OTF2\_Reader.h, 682  
701 OTF2\_Reader\_GetCreator  
OTF2\_Reader\_RegisterGlobalSnapCallbacks, OTF2\_Reader.h, 682  
701 OTF2\_Reader\_GetDefReader  
OTF2\_Reader\_RegisterMarkerCallbacks, OTF2\_Reader.h, 683  
702 OTF2\_Reader\_GetDescription  
OTF2\_Reader\_RegisterSnapCallbacks, OTF2\_Reader.h, 683  
702 OTF2\_Reader\_GetEvtReader  
OTF2\_Reader\_SelectLocation, 703 OTF2\_Reader.h, 683  
OTF2\_Reader\_SetCollectiveCallbacks, OTF2\_Reader\_GetFileSubstrate  
703 OTF2\_Reader.h, 684

OTF2_Reader_GetGlobalDefReader	OTF2_Reader_ReadAllGlobalEvents
OTF2_Reader.h, 684	OTF2_Reader.h, 693
OTF2_Reader_GetGlobalEvtReader	OTF2_Reader_ReadAllGlobalSnapshots
OTF2_Reader.h, 684	OTF2_Reader.h, 693
OTF2_Reader_GetGlobalSnapReader	OTF2_Reader_ReadAllLocalDefinitions
OTF2_Reader.h, 685	OTF2_Reader.h, 693
OTF2_Reader_GetMachineName	OTF2_Reader_ReadAllLocalEvents
OTF2_Reader.h, 685	OTF2_Reader.h, 694
OTF2_Reader_GetMarkerReader	OTF2_Reader_ReadAllLocalSnapshots
OTF2_Reader.h, 685	OTF2_Reader.h, 694
OTF2_Reader_GetMarkerWriter	OTF2_Reader_ReadAllMarkers
OTF2_Reader.h, 686	OTF2_Reader.h, 695
OTF2_Reader_GetNumberOfGlobalDefinitions	OTF2_Reader_ReadGlobalDefinitions
OTF2_Reader.h, 686	OTF2_Reader.h, 695
OTF2_Reader_GetNumberOfLocations	OTF2_Reader_ReadGlobalEvent
OTF2_Reader.h, 687	OTF2_Reader.h, 696
OTF2_Reader_GetNumberOfSnapshots	OTF2_Reader_ReadGlobalEvents
OTF2_Reader.h, 687	OTF2_Reader.h, 696
OTF2_Reader_GetNumberOfThumbnails	OTF2_Reader_ReadGlobalSnapshots
OTF2_Reader.h, 687	OTF2_Reader.h, 696
OTF2_Reader_GetProperty	OTF2_Reader_ReadLocalDefinitions
OTF2_Reader.h, 688	OTF2_Reader.h, 697
OTF2_Reader_GetPropertyNames	OTF2_Reader_ReadLocalEvents
OTF2_Reader.h, 688	OTF2_Reader.h, 697
OTF2_Reader_GetSnapReader	OTF2_Reader_ReadLocalEventsBackward
OTF2_Reader.h, 689	OTF2_Reader.h, 698
OTF2_Reader_GetThumbReader	OTF2_Reader_ReadLocalSnapshots
OTF2_Reader.h, 689	OTF2_Reader.h, 698
OTF2_Reader_GetTraceId	OTF2_Reader_ReadMarkers
OTF2_Reader.h, 689	OTF2_Reader.h, 699
OTF2_Reader_GetVersion	OTF2_Reader_RegisterDefCallbacks
OTF2_Reader.h, 690	OTF2_Reader.h, 699
OTF2_Reader_HasGlobalEvent	OTF2_Reader_RegisterEvtCallbacks
OTF2_Reader.h, 690	OTF2_Reader.h, 700
OTF2_Reader_Open	OTF2_Reader_RegisterGlobalDefCallbacks
OTF2_Reader.h, 691	OTF2_Reader.h, 700
OTF2_Reader_OpenDefFiles	OTF2_Reader_RegisterGlobalEvtCallbacks
OTF2_Reader.h, 691	OTF2_Reader.h, 701
OTF2_Reader_OpenEvtFiles	OTF2_Reader_RegisterGlobalSnapCallbacks
OTF2_Reader.h, 691	OTF2_Reader.h, 701
OTF2_Reader_OpenSnapFiles	OTF2_Reader_RegisterMarkerCallbacks
OTF2_Reader.h, 692	OTF2_Reader.h, 702
OTF2_Reader_ReadAllGlobalDefinitions	OTF2_Reader_RegisterSnapCallbacks
OTF2_Reader.h, 692	OTF2_Reader.h, 702

## INDEX

---

OTF2\_Reader\_SelectLocation  
    OTF2\_Reader.h, [703](#)  
OTF2\_Reader\_SetCollectiveCallbacks  
    OTF2\_Reader.h, [703](#)  
OTF2\_Reader\_SetSerialCollectiveCallbacks  
    OTF2\_Reader.h, [704](#)  
OTF2\_RecorderKind\_enum  
    OTF2\_Definitions.h, [206](#)  
OTF2\_RegionFlag\_enum  
    OTF2\_Definitions.h, [206](#)  
OTF2\_RegionRole\_enum  
    OTF2\_Definitions.h, [207](#)  
OTF2\_RmaAtomicType\_enum  
    OTF2\_Events.h, [274](#)  
OTF2\_RmaSyncLevel\_enum  
    OTF2\_Events.h, [275](#)  
OTF2\_RmaSyncType\_enum  
    OTF2\_Events.h, [275](#)  
OTF2\_SnapReader.h  
    OTF2\_SnapReader\_GetLocationID,  
        [705](#)  
    OTF2\_SnapReader\_ReadSnapshots,  
        [705](#)  
    OTF2\_SnapReader\_Seek, [706](#)  
    OTF2\_SnapReader\_SetCallbacks, [706](#)  
OTF2\_SnapReader\_GetLocationID  
    OTF2\_SnapReader.h, [705](#)  
OTF2\_SnapReader\_ReadSnapshots  
    OTF2\_SnapReader.h, [705](#)  
OTF2\_SnapReader\_Seek  
    OTF2\_SnapReader.h, [706](#)  
OTF2\_SnapReader\_SetCallbacks  
    OTF2\_SnapReader.h, [706](#)  
OTF2\_SnapReaderCallback\_Enter  
    OTF2\_SnapReaderCallbacks.h, [713](#)  
OTF2\_SnapReaderCallback\_MeasurementOnOff  
    OTF2\_SnapReaderCallbacks.h, [713](#)  
OTF2\_SnapReaderCallback\_Metric  
    OTF2\_SnapReaderCallbacks.h, [714](#)  
OTF2\_SnapReaderCallback\_MpiCollectiveBegin  
    OTF2\_SnapReaderCallbacks.h, [715](#)  
OTF2\_SnapReaderCallback\_MpiCollectiveEnd  
    OTF2\_SnapReaderCallbacks.h, [715](#)  
OTF2\_SnapReaderCallback\_MpiIrecv  
    OTF2\_SnapReaderCallbacks.h, [716](#)  
OTF2\_SnapReaderCallback\_MpiIrecvRequest  
    OTF2\_SnapReaderCallbacks.h, [717](#)  
OTF2\_SnapReaderCallback\_MpiIsend  
    OTF2\_SnapReaderCallbacks.h, [718](#)  
OTF2\_SnapReaderCallback\_MpiIsendComplete  
    OTF2\_SnapReaderCallbacks.h, [719](#)  
OTF2\_SnapReaderCallback\_MpiRecv  
    OTF2\_SnapReaderCallbacks.h, [720](#)  
OTF2\_SnapReaderCallback\_MpiSend  
    OTF2\_SnapReaderCallbacks.h, [721](#)  
OTF2\_SnapReaderCallback\_OmpAcquireLock  
    OTF2\_SnapReaderCallbacks.h, [721](#)  
OTF2\_SnapReaderCallback\_OmpFork  
    OTF2\_SnapReaderCallbacks.h, [722](#)  
OTF2\_SnapReaderCallback\_OmpTaskCreate  
    OTF2\_SnapReaderCallbacks.h, [723](#)  
OTF2\_SnapReaderCallback\_OmpTaskSwitch  
    OTF2\_SnapReaderCallbacks.h, [724](#)  
OTF2\_SnapReaderCallback\_ParameterInt  
    OTF2\_SnapReaderCallbacks.h, [724](#)  
OTF2\_SnapReaderCallback\_ParameterString  
    OTF2\_SnapReaderCallbacks.h, [725](#)  
OTF2\_SnapReaderCallback\_ParameterUnsignedInt  
    OTF2\_SnapReaderCallbacks.h, [726](#)  
OTF2\_SnapReaderCallback\_SnapshotEnd  
    OTF2\_SnapReaderCallbacks.h, [727](#)  
OTF2\_SnapReaderCallback\_SnapshotStart  
    OTF2\_SnapReaderCallbacks.h, [727](#)  
OTF2\_SnapReaderCallback\_Unknown  
    OTF2\_SnapReaderCallbacks.h, [728](#)  
OTF2\_SnapReaderCallbacks  
    OTF2\_SnapReaderCallbacks.h, [728](#)  
OTF2\_SnapReaderCallbacks.h  
    OTF2\_SnapReaderCallback\_Enter, [713](#)  
    OTF2\_SnapReaderCallback\_MeasurementOnOff,  
        [713](#)  
    OTF2\_SnapReaderCallback\_Metric,  
        [714](#)  
    OTF2\_SnapReaderCallback\_MpiCollectiveBegin, [714](#)  
    OTF2\_SnapReaderCallback\_MpiCollectiveEnd, [715](#)  
    OTF2\_SnapReaderCallback\_MpiIrecv, [715](#)

---

OTF2\_SnapReaderCallback\_MpiIrecv, 716  
 OTF2\_SnapReaderCallback\_MpiIrecvRequest, 717  
 OTF2\_SnapReaderCallback\_MpiIsend, 718  
 OTF2\_SnapReaderCallback\_MpiIsendComplete, 719  
 OTF2\_SnapReaderCallback\_MpiRecv, 720  
 OTF2\_SnapReaderCallback\_MpiSend, 721  
 OTF2\_SnapReaderCallback\_OmpAcquireLock, 721  
 OTF2\_SnapReaderCallback\_OmpFork, 722  
 OTF2\_SnapReaderCallback\_OmpTaskCreate, 723  
 OTF2\_SnapReaderCallback\_OmpTaskSwitch, 724  
 OTF2\_SnapReaderCallback\_ParameterInt, 724  
 OTF2\_SnapReaderCallback\_ParameterString, 725  
 OTF2\_SnapReaderCallback\_ParameterUnsignedInt, 726  
 OTF2\_SnapReaderCallback\_SnapshotEnd, 727  
 OTF2\_SnapReaderCallback\_SnapshotStart, 727  
 OTF2\_SnapReaderCallback\_Unknown, 728  
 OTF2\_SnapReaderCallbacks, 728  
 OTF2\_SnapReaderCallbacks\_Clear, 729  
 OTF2\_SnapReaderCallbacks\_Delete, 729  
 OTF2\_SnapReaderCallbacks\_New, 729  
 OTF2\_SnapReaderCallbacks\_SetEntryPointCallback, 729  
 OTF2\_SnapReaderCallbacks\_SetMetricCallback, 731  
 OTF2\_SnapReaderCallbacks\_SetMpiCollectiveBeginCallback, 731  
 OTF2\_SnapReaderCallbacks\_SetMpiCollectiveEndCallback, 732  
 OTF2\_SnapReaderCallbacks\_SetMpiIrecvCallback, 732  
 OTF2\_SnapReaderCallbacks\_SetMpiIrecvRequestCallback, 733  
 OTF2\_SnapReaderCallbacks\_SetMpiIsendCallback, 733  
 OTF2\_SnapReaderCallbacks\_SetMpiIsendCompleteCallback, 734  
 OTF2\_SnapReaderCallbacks\_SetMpiRecvCallback, 735  
 OTF2\_SnapReaderCallbacks\_SetMpiSendCallback, 735  
 OTF2\_SnapReaderCallbacks\_SetOmpAcquireLockCallback, 736  
 OTF2\_SnapReaderCallbacks\_SetOmpForkCallback, 736  
 OTF2\_SnapReaderCallbacks\_SetOmpTaskCreateCallback, 737  
 OTF2\_SnapReaderCallbacks\_SetOmpTaskSwitchCallback, 737  
 OTF2\_SnapReaderCallbacks\_SetParameterIntCallback, 738  
 OTF2\_SnapReaderCallbacks\_SetParameterStringCallback, 739  
 OTF2\_SnapReaderCallbacks\_SetParameterUnsignedIntCallback, 739  
 OTF2\_SnapReaderCallbacks\_SetSnapshotEndCallback, 740  
 OTF2\_SnapReaderCallbacks\_SetSnapshotStartCallback, 740  
 OTF2\_SnapReaderCallbacks\_SetUnknownCallback, 741  
 OTF2\_SnapReaderCallbacks\_Clear, 729  
 OTF2\_SnapReaderCallbacks\_Delete, 729  
 OTF2\_SnapReaderCallbacks\_New, 729  
 OTF2\_SnapReaderCallbacks\_SetEntryPointCallback, 729  
 OTF2\_SnapReaderCallbacks\_SetMetricCallback, 730  
 OTF2\_SnapReaderCallbacks\_SetMpiCollectiveBeginCallback, 731  
 OTF2\_SnapReaderCallbacks\_SetMpiCollectiveEndCallback, 732  
 OTF2\_SnapReaderCallbacks\_SetMpiIrecvCallback, 732  
 OTF2\_SnapReaderCallbacks\_SetMpiIrecvRequestCallback, 733  
 OTF2\_SnapReaderCallbacks\_SetMpiIsendCallback, 733  
 OTF2\_SnapReaderCallbacks\_SetMpiIsendCompleteCallback, 734  
 OTF2\_SnapReaderCallbacks\_SetMpiRecvCallback, 735  
 OTF2\_SnapReaderCallbacks\_SetMpiSendCallback, 735  
 OTF2\_SnapReaderCallbacks\_SetOmpAcquireLockCallback, 736  
 OTF2\_SnapReaderCallbacks\_SetOmpForkCallback, 736  
 OTF2\_SnapReaderCallbacks\_SetOmpTaskCreateCallback, 737  
 OTF2\_SnapReaderCallbacks\_SetOmpTaskSwitchCallback, 737  
 OTF2\_SnapReaderCallbacks\_SetParameterIntCallback, 738  
 OTF2\_SnapReaderCallbacks\_SetParameterStringCallback, 739  
 OTF2\_SnapReaderCallbacks\_SetParameterUnsignedIntCallback, 739  
 OTF2\_SnapReaderCallbacks\_SetSnapshotEndCallback, 740  
 OTF2\_SnapReaderCallbacks\_SetSnapshotStartCallback, 740  
 OTF2\_SnapReaderCallbacks\_SetUnknownCallback, 741  
 OTF2\_SnapReaderCallbacks.h, 729



## INDEX

---

OTF2\_SnapReaderCallbacks\_SetEnterCallback, 729  
OTF2\_SnapReaderCallbacks\_SetMeasurementCallback, 730  
OTF2\_SnapReaderCallbacks\_SetMetricCallback, 731  
OTF2\_SnapReaderCallbacks\_SetMpiCollectiveBeginCallback, 731  
OTF2\_SnapReaderCallbacks\_SetMpiCollectiveEndCallback, 732  
OTF2\_SnapReaderCallbacks\_SetMpiIrecvRequestCallback, 733  
OTF2\_SnapReaderCallbacks\_SetMpiIsendCallback, 733  
OTF2\_SnapReaderCallbacks\_SetMpiIsendCompleteCallback, 734  
OTF2\_SnapReaderCallbacks\_SetMpiRecvCallback, 735  
OTF2\_SnapReaderCallbacks\_SetMpiSendCallback, 735  
OTF2\_SnapReaderCallbacks\_SetOmpAcquireLockCallback, 736  
OTF2\_SnapReaderCallbacks\_SetOmpForkCallback, 736  
OTF2\_SnapReaderCallbacks\_SetOmpTaskCreateCallback, 737  
OTF2\_SnapReaderCallbacks\_SetOmpTaskSwitchCallback, 737  
OTF2\_SnapReaderCallbacks\_SetParameterIntCallback, 738  
OTF2\_SnapReaderCallbacks\_SetParameterStringCallback, 739  
OTF2\_SnapReaderCallbacks\_SetParameterUnsignedIntCallback, 739  
OTF2\_SnapReaderCallbacks\_SetSnapshotStartCallback, 740  
OTF2\_SnapReaderCallbacks\_SetSnapshotEndCallback, 740  
OTF2\_SnapReaderCallbacks\_SetUnknownCallback, 741  
OTF2\_SnapWriter, 744  
OTF2\_SnapWriter.h, 744  
OTF2\_SnapWriter\_Enter, 745  
OTF2\_SnapWriter\_GetLocationID, 746  
OTF2\_SnapWriter\_MeasurementOnOff, 746  
OTF2\_SnapWriter\_Metric, 746  
OTF2\_SnapWriter\_MpiCollectiveBegin, 747  
OTF2\_SnapWriter\_MpiCollectiveEnd, 748  
OTF2\_SnapWriter\_MpiIrecv, 748  
OTF2\_SnapWriter\_MpiIrecvRequest, 749  
OTF2\_SnapWriter\_MpiIsend, 750  
OTF2\_SnapWriter\_MpiIsendComplete, 751  
OTF2\_SnapWriter\_MpiRecv, 751  
OTF2\_SnapWriter\_MpiSend, 752  
OTF2\_SnapWriter\_OmpAcquireLock, 754  
OTF2\_SnapWriter\_OmpFork, 754  
OTF2\_SnapWriter\_OmpTaskCreate, 755  
OTF2\_SnapWriter\_OmpTaskSwitch, 756  
OTF2\_SnapWriter\_ParameterInt, 756  
OTF2\_SnapWriter\_ParameterString, 756  
OTF2\_SnapWriter\_ParameterUnsignedInt, 757  
OTF2\_SnapWriter\_SnapshotEnd, 758  
OTF2\_SnapWriter\_SnapshotStart, 758  
OTF2\_SnapWriter.h, 745  
OTF2\_SnapWriter\_MeasurementOnOff, 746  
OTF2\_SnapWriter\_Metric, 746  
OTF2\_SnapWriter\_MpiCollectiveBegin, 747  
OTF2\_SnapWriter.h, 747

---

- OTF2\_SnapWriter\_MpiCollectiveEnd  
OTF2\_SnapWriter.h, [748](#)
- OTF2\_SnapWriter\_MpiIrecv  
OTF2\_SnapWriter.h, [748](#)
- OTF2\_SnapWriter\_MpiIrecvRequest  
OTF2\_SnapWriter.h, [749](#)
- OTF2\_SnapWriter\_MpiIsend  
OTF2\_SnapWriter.h, [750](#)
- OTF2\_SnapWriter\_MpiIsendComplete  
OTF2\_SnapWriter.h, [751](#)
- OTF2\_SnapWriter\_MpiRecv  
OTF2\_SnapWriter.h, [751](#)
- OTF2\_SnapWriter\_MpiSend  
OTF2\_SnapWriter.h, [752](#)
- OTF2\_SnapWriter\_OmpAcquireLock  
OTF2\_SnapWriter.h, [753](#)
- OTF2\_SnapWriter\_OmpFork  
OTF2\_SnapWriter.h, [754](#)
- OTF2\_SnapWriter\_OmpTaskCreate  
OTF2\_SnapWriter.h, [754](#)
- OTF2\_SnapWriter\_OmpTaskSwitch  
OTF2\_SnapWriter.h, [755](#)
- OTF2\_SnapWriter\_ParameterInt  
OTF2\_SnapWriter.h, [756](#)
- OTF2\_SnapWriter\_ParameterString  
OTF2\_SnapWriter.h, [756](#)
- OTF2\_SnapWriter\_ParameterUnsignedInt  
OTF2\_SnapWriter.h, [757](#)
- OTF2\_SnapWriter\_SnapshotEnd  
OTF2\_SnapWriter.h, [758](#)
- OTF2\_SnapWriter\_SnapshotStart  
OTF2\_SnapWriter.h, [758](#)
- OTF2\_SystemTreeDomain\_enum  
OTF2\_Definitions.h, [209](#)
- OTF2\_Thumbnail.h
  - OTF2\_ThumbReader\_GetHeader, [760](#)
  - OTF2\_ThumbReader\_ReadSample,  
[761](#)
  - OTF2\_ThumbWriter\_WriteSample,  
[761](#)
- OTF2\_ThumbnailType\_enum  
OTF2\_GeneralDefinitions.h, [459](#)
- OTF2\_ThumbReader\_GetHeader  
OTF2\_Thumbnail.h, [760](#)
- OTF2\_ThumbReader\_ReadSample  
OTF2\_Thumbnail.h, [761](#)
- OTF2\_ThumbWriter\_WriteSample  
OTF2\_Thumbnail.h, [761](#)
- OTF2\_Type\_enum  
OTF2\_GeneralDefinitions.h, [460](#)
- Usage in reading mode - a simple exam-  
ple, [115](#)
- Usage in reading mode - MPI example,  
[101](#)
- Usage in writing mode - a simple exam-  
ple, [86](#)
- Usage in writing mode - MPI example,  
[108](#)
- Usage of OTF2 tools, [17](#)