

# **OPEN TRACE FORMAT 2**

## **USER MANUAL**

1.2.1 (revision 3238)

Wed Sep 18 2013 11:37:18

---

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

---

## CONTENTS

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

## CONTENTS

---

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

---

## CONTENTS

---

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

## CONTENTS

---

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

---

## CONTENTS

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

# **Chapter 1**

## **Open Trace Format 2**

### **1.1 Introduction**

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

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

### **1.2 Get started**

[Usage in writing mode](#)

[Usage in reading mode](#)

[Definition records](#)

[Event records](#)

[Snapshot records](#)

[Usage of OTF2 tools](#)

## **CHAPTER 1. OPEN TRACE FORMAT 2**

---

# **Appendices**



## Appendix A

# OTF2 Tools

### A.1 Usage of OTF2 tools

#### A.1.1 OTF2 config tool

A call to otf2-config has the following syntax:

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

Commands:
  --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
  --help        prints this usage information

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

Options:
  --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
  --cuda        specifies that the required flags are for the CUDA compiler
```

## APPENDIX A. OTF2 TOOLS

---

### A.1.2 OTF2 print tool

A call to oft2-print has the following syntax:

```
Usage: oft2-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
-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

### A.1.3 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.

### A.1.4 OTF2 marker tool

A call to oft2-marker has the following syntax:

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

## A.1 Usage of OTF2 tools

---

Read or edit a marker file.

Options:

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

Argument descriptions:

```
<GROUP>, <CATEGORY>, <TEXT>  
          Arbitrary strings.  
<SEVERITY>          One of:  
          * 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>
```

## **APPENDIX A. OTF2 TOOLS**

---

<SCOPE-REF> must be a valid definition reference of the specified scope. Use 'otf2-print -G' for a list of defined references.  
There is no <SCOPE-REF> for <SCOPE> 'GLOBAL'.  
For a scope 'GROUP' the type of the referenced group must be 'OTF2\_GROUP\_TYPE\_LOCATIONS' or 'OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS'.

## Appendix B

# OTF2 INSTALL

For generic installation instructions see below.

Configuration of OTF2

\*\*\*\*\*

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

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

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

Defaults for the options are specified in brackets.

Configuration:

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

Installation directories:

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

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

## **APPENDIX B. OTF2 INSTALL**

---

For better control, use the options below.

Fine tuning of the installation directories:

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

Program names:

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

System types:

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

Optional Features:

```
--disable-option-checking ignore unrecognized --enable/--with options
--disable-FEATURE      do not include FEATURE (same as --enable-FEATURE=no)
--enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
--enable-silent-rules   less verbose build output (undo: 'make V=1')
--disable-silent-rules  verbose build output (undo: 'make V=0')
--with-platform=(auto,disabled,<platform>)
                           autodetect platform [auto], disabled or select one
                           from: altix, aix, arm, bgl, bgp, bgq, crayxt, linux,
                           solaris, mac, necsx.
--disable-dependency-tracking speeds up one-time build
--enable-dependency-tracking do not reject slow dependency extractors
--enable-debug          activate internal debug output [no]
--enable-backend-test-runs
                           Run tests at make check [no]. If disabled, tests are
                           still build at make check. Additionally, scripts
                           (scorep_*tests.sh) containing the tests are
                           generated in <builddir>/build-backend.
--enable-shared[=PKGS]   build shared libraries [default=no]
--enable-static[=PKGS]   build static libraries [default=yes]
--enable-fast-install[=PKGS]
                           optimize for fast installation [default=yes]
```

---

```

--disable-libtool-lock    avoid locking (might break parallel builds)

Optional Packages:
  --with-PACKAGE [=ARG]      use PACKAGE [ARG=yes]
  --without-PACKAGE        do not use PACKAGE (same as --with-PACKAGE=no)
  --with-sionconfig=(yes|no|<path-to-sionconfig>)
                            Whether to use sionconfig and where to find it.
                            "yes" assumes it is in PATH [no].
  --with-otf-prefix=PREFIX
                            Prefix where otf is installed (optional)
  --with-otf-exec-prefix=PREFIX
                            Exec prefix where otf is installed (optional)
  --with-pic                try to use only PIC/non-PIC objects [default=use
                            both]
  --with-gnu-ld              assume the C compiler uses GNU ld [default=no]
  --with-sysroot=DIR Search for dependent libraries within DIR
                            (or the compiler's sysroot if not specified).

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

```

---

## **APPENDIX B. OTF2 INSTALL**

---

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

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

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

Installation Instructions  
\*\*\*\*\*

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

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

Basic Installation  
=====

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

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

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

If you need to do unusual things to compile the package, please try to figure out how 'configure' could check whether to do them, and mail diffs or instructions to the address given in the 'README' so they can be considered for the next release. If you are using the cache, and at

---

some point ‘config.cache’ contains results you don’t want to keep, you may remove or edit it.

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

The simplest way to compile this package is:

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

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

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

## **APPENDIX B. OTF2 INSTALL**

---

=====

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

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

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

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

### Compiling For Multiple Architectures

=====

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

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

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

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

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

### Installation Names

=====

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

You can specify separate installation prefixes for

---

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

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

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

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

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

#### Optional Features

---

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

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

## **APPENDIX B. OTF2 INSTALL**

---

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

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

### Particular systems

---

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

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

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

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

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

and if that doesn't work, try

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

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

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

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

### Specifying the System Type

---

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

---

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

CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

OS  
KERNEL-OS

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

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

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

#### Sharing Defaults

---

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

#### Defining Variables

---

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

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

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

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

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

#### 'configure' Invocation

---

## **APPENDIX B. OTF2 INSTALL**

---

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

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

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

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

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

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

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

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

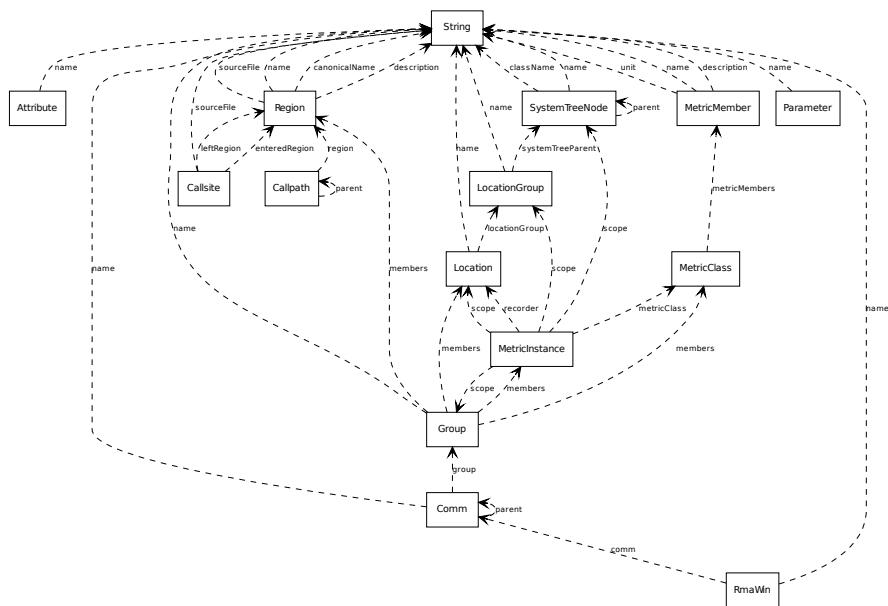
'--prefix=DIR'
    Use DIR as the installation prefix. *note Installation Names:::
    for more details, including other options available for fine-tuning
    the installation locations.

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

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

## Appendix C

### List of all definition records



#### C.1 ClockProperties

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

This definition is only valid as a global definition.

---

## APPENDIX C. LIST OF ALL DEFINITION RECORDS

---

### Attributes

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

### See also

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

### Since

Version 1.0

## C.2 MappingTable

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

This definition is only valid as a local definition.

### Attributes

OTF2_MappingType	mappingType	Says to what type of ID the mapping table has to be applied.
const OTF2_IdMap*	idMap	Mapping table.

### See also

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

### Since

Version 1.0

## C.3 ClockOffset

Clock offsets are used for clock corrections.

## C.5 Attribute

---

This definition is only valid as a local definition.

### Attributes

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

### See also

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

### Since

Version 1.0

## C.4 OTF2\_StringRef String

### Attributes

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

### See also

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

### Since

Version 1.0

## C.5 OTF2\_AttributeRef Attribute

### Attributes

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

---

## APPENDIX C. LIST OF ALL DEFINITION RECORDS

### See also

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

### Since

Version 1.0

## C.6 OTF2\_SystemTreeNodeRef SystemTreeNode

### Attributes

<a href="#">OTF2StringRef</a>	name	Free form instance name of this node. References a <a href="#">String</a> definition.
<a href="#">OTF2StringRef</a>	className	Free form class name of this node References a <a href="#">String</a> definition.
<a href="#">OTF2_-SystemTreeNodeRef</a>	parent	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.

### See also

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

### Since

Version 1.0

## C.7 OTF2\_LocationGroupRef LocationGroup

### Attributes

<a href="#">OTF2StringRef</a>	name	Name of the group. References a <a href="#">String</a> definition.
<a href="#">OTF2_-LocationGroupType</a>	location-GroupType	Type of this group.
<a href="#">OTF2_-SystemTreeNodeRef</a>	systemTreeParent	Parent of this location group in the system tree. References a <a href="#">SystemTreeNode</a> definition.

## C.9 Region

---

See also

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

Since

Version 1.0

## C.8 OTF2\_LocationRef Location

Attributes

<a href="#">OTF2StringRef</a>	name	Name of the location References a <a href="#">String</a> definition.
<a href="#">OTF2_LocationType</a>	location-Type	Location type.
uint64_t	numberOfEvents	Number of events this location has recorded.
<a href="#">OTF2_LocationGroupRef</a>	location-Group	Location group which includes this location. References a <a href="#">LocationGroup</a> definition.

See also

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

Since

Version 1.0

## C.9 OTF2\_RegionRef Region

Attributes

<a href="#">OTF2StringRef</a>	name	Name of the region (demangled name if available). References a <a href="#">String</a> definition.
<a href="#">OTF2StringRef</a>	canonical-Name	Alternative name of the region (e.g. mangled name). References a <a href="#">String</a> definition. Since version 1.1.
<a href="#">OTF2StringRef</a>	description	A more detailed description of this region. References a <a href="#">String</a> definition.
<a href="#">OTF2_RegionRole</a>	regionRole	Region role. Since version 1.1.

## APPENDIX C. LIST OF ALL DEFINITION RECORDS

---

<a href="#">OTF2_Paradigm</a>	paradigm	Paradigm. Since version 1.1.
<a href="#">OTF2_RegionFlag</a>	regionFlags	Region flags. Since version 1.1.
<a href="#">OTF2_StringRef</a>	sourceFile	The source file where this region was declared. References a <a href="#">String</a> definition.
uint32_t	beginLineNumber	Starting line number of this region in the source file.
uint32_t	endLineNumber	Ending line number of this region in the source file.

See also

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

Since

Version 1.0

### C.10 OTF2\_CallsiteRef Callsite

Attributes

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

See also

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

Since

Version 1.0

## C.12 Group

---

### C.11 OTF2\_CallpathRef Callpath

#### Attributes

OTF2_CallpathRef	parent	References a <a href="#">Callpath</a> definition.
OTF2_RegionRef	region	References a <a href="#">Region</a> definition.

#### See also

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

#### Since

Version 1.0

## C.12 OTF2\_GroupRef Group

#### Attributes

OTF2StringRef	name	Name of this group References a <a href="#">String</a> definition.
OTF2_GroupType	groupType	The type of this group. Since version 1.2.
OTF2_Paradigm	paradigm	The paradigm of this communication group. Since version 1.2.
OTF2_GroupFlag	groupFlags	Flags for this group. Since version 1.2.
uint32_t	num- berOfMem- bers	The number of members in this group.
uint64_t	members [ num- berOfMem- bers ]	The identifiers of the group members.

#### See also

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

#### Since

Version 1.0

---

## APPENDIX C. LIST OF ALL DEFINITION RECORDS

---

### C.13 OTF2\_MetricMemberRef MetricMember

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

#### Attributes

<a href="#">OTF2StringRef</a>	name	Name of the metric. References a <a href="#">String</a> definition.
<a href="#">OTF2StringRef</a>	description	Description of the metric. References a <a href="#">String</a> definition.
<a href="#">OTF2_MetricType</a>	metricType	Metric type: PAPI, etc.
<a href="#">OTF2_MetricMode</a>	metricMode	Metric mode: accumulative, fix, relative, etc.
<a href="#">OTF2_Type</a>	valueType	Type of the value: int64_t, uint64_t, or double.
<a href="#">OTF2_MetricBase</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 base <sup>exponent</sup> , 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.
<a href="#">OTF2StringRef</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="#">String</a> definition.

#### See also

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

---

## C.15 MetricInstance

---

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

### Since

Version 1.0

## C.14 OTF2\_MetricRef MetricClass

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

### Attributes

uint8_t	numberOfMetrics	Number of metrics within the set.
OTF2_MetricMemberRef	metricMembers [ numberOfMetrics ]	List of metric members. References a <a href="#">MetricMember</a> definition.
OTF2_MetricOccurrence	metricOccurrence	Defines occurrence of a metric set.
OTF2RecorderKind	recorderKind	What kind of locations will record this metric class, or will this metric class only be recorded by metric instances. Since version 1.2.

### See also

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

### Since

Version 1.0

## C.15 OTF2\_MetricRef MetricInstance

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

---

## APPENDIX C. LIST OF ALL DEFINITION RECORDS

### Attributes

<a href="#">OTF2_MetricRef</a>	metricClass	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.
<a href="#">OTF2_LocationRef</a>	recorder	Recorder of the metric: location ID. References a <a href="#">Location</a> definition.
<a href="#">OTF2_MetricScope</a>	metricScope	Defines type of scope: location, location group, system tree node, or a generic group of locations.
uint64_t	scope	Scope of metric: ID of a location, location group, system tree node, or a generic group of locations.

### See also

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

### Since

Version 1.0

## C.16 OTF2\_CommRef Comm

### Attributes

<a href="#">OTF2StringRef</a>	name	The name given by calling <code>MPI_Comm_set_name</code> on this communicator. Or the empty name to indicate that no name was given. References a <a href="#">String</a> definition.
<a href="#">OTF2GroupRef</a>	group	The describing MPI group of this MPI communicator The group needs to be of type <a href="#">OTF2_GROUP_TYPE_MPI_GROUP</a> or <a href="#">OTF2_GROUP_TYPE_MPI_COMM_SELF</a> . References a <a href="#">Group</a> definition.
<a href="#">OTF2CommRef</a>	parent	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.

## C.18 RmaWin

---

### See also

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

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

### Since

Version 1.0

## C.17 OTF2\_ParameterRef Parameter

### Attributes

<a href="#">OTF2StringRef</a>	name	Name of the parameter (variable name etc.) References a <a href="#">String</a> definition.
<a href="#">OTF2_ParameterType</a>	parameter-Type	Type of the parameter, <a href="#">OTF2_ParameterType</a> for possible types.

### See also

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

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

### Since

Version 1.0

## C.18 OTF2\_RmaWinRef RmaWin

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

### Attributes

<a href="#">OTF2StringRef</a>	name	Name, e.g. 'GASPI Queue 1', 'NVidia Card 2', etc.. References a <a href="#">String</a> definition.
<a href="#">OTF2_CommRef</a>	comm	Communicator object used to create the window. References a <a href="#">Comm</a> definition.

### See also

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

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

---

## APPENDIX C. LIST OF ALL DEFINITION RECORDS

**Since**

Version 1.2

### C.19 MetricClassRecorder

**Attributes**

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

**See also**

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

**Since**

Version 1.2

### C.20 SystemTreeNodeProperty

**Attributes**

<a href="#">OTF2_-SystemTreeNodeRef</a>	systemTreeNode	Parent <a href="#">SystemTreeNode</a> definition to which this one is a supplementary definition. References a <a href="#">SystemTreeNode</a> definition.
<a href="#">OTF2StringRef</a>	name	Name of the property. References a <a href="#">String</a> definition.
<a href="#">OTF2StringRef</a>	value	Property value. References a <a href="#">String</a> definition.

**See also**

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

**Since**

Version 1.2

## C.21 SystemTreeNodeDomain

---

### C.21 SystemTreeNodeDomain

#### Attributes

OTF2_-SystemTreeNodeRef	sys-temTreeNode	Parent <a href="#">SystemTreeNode</a> definition to which this one is a supplementary definition. References a <a href="#">SystemTreeNode</a> definition.
OTF2_-SystemTreeDomain	sys-temTreeDo-main	

#### See also

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

#### Since

Version 1.2

**APPENDIX C. LIST OF ALL DEFINITION RECORDS**

---

## Appendix D

# List of all event records

### D.1 BufferFlush

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

#### Attributes

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

### D.2 MeasurementOnOff

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

#### 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_-MeasurementMode</a>	measure- mentMode	Is the measurement turned on ( <a href="#">OTF2_-MEASUREMENT_ON</a> ) or off ( <a href="#">OTF2_-MEASUREMENT_OFF</a> )?

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

### See also

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

### Since

Version 1.0

## D.3 Enter

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

### Attributes

<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

## D.4 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.

## D.6 MpiIsend

---

### See also

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

### Since

Version 1.0

## D.5 MpiSend

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

### Attributes

<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 <i>communicator</i> .
<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

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

### Since

Version 1.0

## D.6 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).

## APPENDIX D. LIST OF ALL EVENT RECORDS

---

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

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

### Since

Version 1.0

## D.7 MpilsendComplete

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\\_MpiIsendComplete\(\)](#)

### Since

Version 1.0

## D.9 MpiRecv

---

### D.8 MpilrecvRequest

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

#### 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\\_MpiIrecvRequest\(\)](#)

#### Since

Version 1.0

## D.9 MpiRecv

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

#### Attributes

<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 <i>communicator</i> .
<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

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

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

### Since

Version 1.0

### D.10 MpiRecv

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

#### Attributes

<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 <i>communicator</i> .
<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

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

### Since

Version 1.0

### D.11 MpiRequestTest

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

#### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location where this event happened.
<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
38	uint64_t	requestID

38 ID of the related request

## D.13 MpiCollectiveBegin

---

### See also

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

### Since

Version 1.0

## D.12 MpiRequestCancelled

This events appears if the program canceled a request.

### Attributes

<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\\_MpiRequestCancelled\(\)](#)

### Since

Version 1.0

## D.13 MpiCollectiveBegin

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

### Attributes

<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\\_MpiCollectiveBegin\(\)](#)

### Since

Version 1.0

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

### D.14 MpiCollectiveEnd

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

#### Attributes

<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 <i>communicator</i> .
uint64_t	sizeSent	Size of the sent message.
uint64_t	sizeReceived	Size of the received message.

#### See also

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

#### Since

Version 1.0

### D.15 OmpFork

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

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

#### Attributes

<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.

## D.17 OmpAcquireLock

---

### See also

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

### Since

Version 1.0

### Deprecated

In version 1.2

## D.16 OmpJoin

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

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

### Attributes

<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

## D.17 OmpAcquireLock

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

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

## APPENDIX D. LIST OF ALL EVENT RECORDS

---

### Attributes

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

### See also

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

### Since

Version 1.0

### Deprecated

In version 1.2

## D.18 OmpReleaseLock

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

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

### Attributes

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

## D.20 OmpTaskSwitch

---

### See also

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

### Since

Version 1.0

### Deprecated

In version 1.2

## D.19 OmpTaskCreate

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

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

### Attributes

<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

## D.20 OmpTaskSwitch

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

## APPENDIX D. LIST OF ALL EVENT RECORDS

---

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

### Attributes

<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

## D.21 OmpTaskComplete

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

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

### Attributes

<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

## D.23 ParameterString

---

### D.22 Metric

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

#### Attributes

<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.
<a href="#">OTF2_MetricValue</a>	metricValues [ numberOfMetrics ]	List of metric values.

#### See also

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

#### Since

Version 1.0

## D.23 ParameterString

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

## APPENDIX D. LIST OF ALL EVENT RECORDS

---

### Attributes

<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.
<a href="#">OTF2StringRef</a>	string	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.

### See also

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

### Since

Version 1.0

## D.24 ParameterInt

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

### Attributes

<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.
int64_t	value	Value of the recorded parameter.

### See also

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

## D.26 RmaWinCreate

---

### Since

Version 1.0

## D.25 ParameterUnsignedInt

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

### Attributes

<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

## D.26 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.

## **APPENDIX D. LIST OF ALL EVENT RECORDS**

---

### **See also**

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

### **Since**

Version 1.2

## **D.27 RmaWinDestroy**

An RmaWinDestroy record denotes the destruction of an RMA window.

### **Attributes**

<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 <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\\_RmaWinDestroy\(\)](#)

### **Since**

Version 1.2

## **D.28 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\(\)](#)

## D.30 RmaGroupSync

---

### Since

Version 1.2

## D.29 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

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

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

---

<a href="#">OTF2_TimeStamp</a>	timestamp	The time when this event happened.
<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.
<a href="#">OTF2_GroupRef</a>	group	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.

### See also

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

### Since

Version 1.2

## D.31 RmaRequestLock

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

### Attributes

<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.

### D.33 RmaTryLock

---

#### See also

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

#### Since

Version 1.2

## D.32 RmaAcquireLock

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

#### Attributes

<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

## D.33 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 D. LIST OF ALL EVENT RECORDS

<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\\_RmaTryLock\(\)](#)

Since

Version 1.2

### D.34 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

## D.36 RmaWaitChange

---

### D.35 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

## D.36 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\(\)](#)

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

**Since**

Version 1.2

### D.37 RmaPut

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

**Attributes**

<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.
uint64_t	bytes	Bytes sent to target.
uint64_t	matchingId	ID used for matching the appropriate completion record.

**See also**

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

**Since**

Version 1.2

### D.38 RmaGet

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

**Attributes**

<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.

## D.39 RmaAtomic

---

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

### See also

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

### Since

Version 1.2

## D.39 RmaAtomic

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

### Attributes

<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

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

### D.40 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

### D.41 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.

## D.43 RmaOpCompleteRemote

---

### See also

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

### Since

Version 1.2

## D.42 RmaOpTest

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

### Attributes

<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

## D.43 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.

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

<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\\_RmaOpCompleteRemote\(\)](#)

Since

Version 1.2

### D.44 ThreadFork

An ThreadFork record marks that an thread forks a 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_Paradigm</a>	model	
uint32_t	numberOfRequestedThreads	Requested size of the team.

See also

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

Since

Version 1.2

### D.45 ThreadJoin

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

## D.47 ThreadTeamEnd

---

### Attributes

<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	

### See also

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

### Since

Version 1.2

## D.46 ThreadTeamBegin

### 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

## D.47 ThreadTeamEnd

### 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.

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

### See also

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

### Since

Version 1.2

## D.48 ThreadAcquireLock

An ThreadAcquireLock record marks that a thread acquires an lock.

### Attributes

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

### See also

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

### Since

Version 1.2

## D.49 ThreadReleaseLock

An ThreadReleaseLock record marks that a thread releases an lock.

### Attributes

<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	

## D.50 ThreadTaskCreate

---

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

### See also

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

### Since

Version 1.2

## D.50 ThreadTaskCreate

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

### Attributes

<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. (This is redundant, remove?)
uint32_t	generationNumber	Thread-private generation number of task's creating thread.

### See also

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

### Since

Version 1.2

---

## APPENDIX D. LIST OF ALL EVENT RECORDS

### D.51 ThreadTaskSwitch

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

#### Attributes

<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

### D.52 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.

## **D.52 ThreadTaskComplete**

---

### **See also**

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

### **Since**

Version 1.2

**APPENDIX D. LIST OF ALL EVENT RECORDS**

---

## Appendix E

# List of all snapshot records

### E.1 SnapshotStart

This record marks the start of a snapshot.

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

#### Attributes

<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 <i>SnapshotEnd</i> record.

#### See also

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

#### Since

Version 1.2

---

## APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

### E.2 SnapshotEnd

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

#### Attributes

<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	contReadPos	Position to continue reading in the event trace.

#### See also

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

#### Since

Version 1.2

### E.3 MeasurementOnOffSnap

The last occurrence of an [\*MeasurementOnOff\*](#) event of this location, if any.

#### Attributes

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

## E.5 MpiSendSnap

---

### E.4 EnterSnap

This record exists for each [Enter](#) event where the corresponding [Leave](#) event did not occur before the snapshot.

#### Attributes

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

## E.5 MpiSendSnap

This record exists for each [MpiSend](#) event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an [MpiRecv](#) or an [MpiIrecv](#) event. Note that it may so, that a previous [MpiIsend](#) with the same envelope than this one is neither completed nor 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.
<a href="#">OTF2_TimeStamp</a>	origEvent-Time	The original time this event happened.
uint32_t	receiver	MPI rank of receiver in <i>communicator</i> .

## APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

---

<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

### E.6 MpilsendSnap

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 [Mpirecv](#) 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.
<a href="#">OTF2_TimeStamp</a>	origEventTime	The original time this event happened.
uint32_t	receiver	MPI rank of receiver in <i>communicator</i> .
<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

## E.8 MpiRecvSnap

---

### See also

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

### Since

Version 1.2

## E.7 MpilsendCompleteSnap

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

### Attributes

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

### See also

[\*MpiIsendComplete\*](#) event  
[\*OTF2\\_SnapWriter\\_MpiIsendComplete\(\)\*](#)

### Since

Version 1.2

## E.8 MpiRecvSnap

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

## APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

---

### Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEvent-Time	The original time this event happened.
uint32_t	sender	MPI rank of sender in <i>communicator</i> .
OTF2_CommRef	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

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

### Since

Version 1.2

## E.9 MpiRecvRequestSnap

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

### Attributes

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

## E.10 MpiIrecvSnap

---

### See also

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

### Since

Version 1.2

## E.10 MpilrecvSnap

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
<a href="#">OTF2_TimeStamp</a>	origEvent-Time	The original time this event happened.
uint32_t	sender	MPI rank of sender in <i>communicator</i> .
<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

---

## APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

### E.11 MpiCollectiveBeginSnap

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

#### Attributes

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

#### See also

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

#### Since

Version 1.2

### E.12 MpiCollectiveEndSnap

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

#### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
<a href="#">OTF2_TimeStamp</a>	origEventTime	The original time 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 <i>communicator</i> .
uint64_t	sizeSent	Size of the sent message.
uint64_t	sizeReceived	Size of the received message.

## E.14 OmpAcquireLockSnap

---

### See also

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

### Since

Version 1.2

## E.13 OmpForkSnap

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

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
<a href="#">OTF2_TimeStamp</a>	origEvent-Time	The original time this event happened.
uint32_t	numberOfRequestedThreads	Requested size of the team.

### See also

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

### Since

Version 1.2

## E.14 OmpAcquireLockSnap

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

### Attributes

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

## APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

---

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

### See also

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

### Since

Version 1.2

## E.15 OmpTaskCreateSnap

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

### Attributes

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

### See also

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

### Since

Version 1.2

## E.17 MetricSnap

---

### E.16 OmpTaskSwitchSnap

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

#### Attributes

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

#### See also

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

#### Since

Version 1.2

## E.17 MetricSnap

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

As an exception for metric classes where the metric mode denotes 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.
<a href="#">OTF2_TimeStamp</a>	origEventTime	The original time 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.

---

## APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

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

### See also

[Metric](#) event  
[OTF2\\_SnapWriter\\_Metric\(\)](#)

### Since

Version 1.2

## E.18 ParameterStringSnap

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

### Attributes

OTF2_LocationRef	location	The location of the snapshot.
OTF2_TimeStamp	timestamp	The snapshot time of this record.
OTF2_TimeStamp	origEventTime	The original time this event happened.
OTF2_ParameterRef	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.
OTF2StringRef	string	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.

## E.20 ParameterUnsignedIntSnap

---

See also

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

Since

Version 1.2

## E.19 ParameterIntSnap

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

Attributes

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

## E.20 ParameterUnsignedIntSnap

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

## APPENDIX E. LIST OF ALL SNAPSHOT RECORDS

### Attributes

<a href="#">OTF2_LocationRef</a>	location	The location of the snapshot.
<a href="#">OTF2_TimeStamp</a>	timestamp	The snapshot time of this record.
<a href="#">OTF2_TimeStamp</a>	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.
uint64_t	value	Value of the recorded parameter.

### See also

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

### Since

Version 1.2

## Appendix F

# Usage in writing mode

### F.1 Usage in writing mode - a simple example

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

First include the OTF2 header.

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

For this example an additional include statement is necessary.

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

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

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

Define a pre and post flush callback. If no memory is left in OTF2's internal memory buffer or the writer handle is closed a memory buffer flushing routine is triggered. The pre flush callback is triggered right before a buffer flush. It needs to return either OTF2\_FLUSH to flush the recorded data to a file or OTF2\_NO\_FLUSH to suppress flushing data to a file. The post flush callback is triggered right after a memory buffer flush. It has to return a current timestamp which is recorded to mark the time spend in a buffer flush.

---

## APPENDIX F. USAGE IN WRITING MODE

---

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

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

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

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

Create new archive handle.

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

Set the flush callbacks.

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

Define archive as master.

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

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

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

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

## F.1 Usage in writing mode - a simple example

---

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

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

```
OTF2_GlobalDefWriter_WriteString( global_def_writer, 0, "" );
OTF2_GlobalDefWriter_WriteString( global_def_writer, 1, "Master Process" );

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

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

```
OTF2_GlobalDefWriter_WriteRegion( global_def_writer, 23, 3, 4, 5,
OTF2_REGION_ROLE_FUNCTION, OTF2_PARADIGM_USER, OTF2_REGION_FLAG_NONE, 0, 0, 0 );
```

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

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

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

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

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

```
    OTF2_Archive_Close( archive );

    return EXIT_SUCCESS;
}
```

To compile your program use a command like:

```
gcc `otf2-config --cflags` -c otf2_writer_example.c -o otf2_writer_example.o
```

Now you can link your program with:

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

---

**APPENDIX F. USAGE IN WRITING MODE**

## Appendix G

# Usage in reading mode

### G.1 Usage in reading mode - a simple example

This is a short example of how to use the OTF2 reading interface. It shows hows to define and register callbacks and how to use the reader interface to read all events of a given OTF2 archive.

First include the OTF2 header.

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

For this example two additional include statements are necessary.

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

Define an event callback for entering and leaving a region.

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

    return OTF2_SUCCESS;
}
```

---

## APPENDIX G. USAGE IN READING MODE

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

    return OTF2_SUCCESS;
}
```

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

```
OTF2_CallbackCode
GlobDefLocation_Register( void*                     (userData,
                           OTF2_LocationRef        location,
                           OTF2StringRef           name,
                           OTF2_LocationType       locationType,
                           uint64_t                 numberOfEvents,
                           OTF2_LocationGroupRef   locationGroup )
{
    OTF2_Reader* reader = ( OTF2_Reader* )userData;
    OTF2_EvtReader* evt_reader = OTF2_Reader_GetEvtReader( reader, location );

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

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

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

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

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

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

---

## G.1 Usage in reading mode - a simple example

---

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

```
OTF2_GlobalDefReaderCallbacks* global_def_callbacks =
OTF2_GlobalDefReaderCallbacks_New();
OTF2_GlobalDefReaderCallbacks_SetLocationCallback( global_def_callbacks, &G
lobDefLocation_Register );
OTF2_Reader_RegisterGlobalDefCallbacks( reader, global_def_reader, global_d
ef_callbacks, reader );
OTF2_GlobalDefReaderCallbacks_Delete( global_def_callbacks );
```

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

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

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

```
OTF2_GlobalEvtReader* global_evt_reader = OTF2_Reader_GetGlobalEvtReader( r
eader );
```

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

```
OTF2_GlobalEvtReaderCallbacks* event_callbacks =
OTF2_GlobalEvtReaderCallbacks_New();
OTF2_GlobalEvtReaderCallbacks_SetEnterCallback( event_callbacks, &Enter_pri
nt );
OTF2_GlobalEvtReaderCallbacks_SetLeaveCallback( event_callbacks, &Leave_pri
nt );

OTF2_Reader_RegisterGlobalEvtCallbacks( reader, global_evt_reader, event_ca
llbacks, NULL );
OTF2_GlobalEvtReaderCallbacks_Delete( event_callbacks );
```

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

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

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

## **APPENDIX G. USAGE IN READING MODE**

---

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

To compile your program use a command like:

```
gcc `otf2-config --cflags` -c otf2_reader_example.c -o otf2_reader_example.o
```

Now you can link your program with:

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

## Appendix H

### Deprecated List

**Page List of all event records** In version 1.2

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

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

**Global OTF2\_EvtWriter\_OmpAcquireLock**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList,  
In version 1.2

**Global OTF2\_EvtWriter\_OmpFork**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_T  
In version 1.2

**Global OTF2\_EvtWriter\_OmpJoin**(OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_T  
In version 1.2

## **APPENDIX H. DEPRECATED LIST**

---

**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

## Appendix I

# Data Structure Documentation

### I.1 OTF2\_AttributeValue Union Reference

Value container for an attributes.

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

#### Data Fields

- **OTF2\_AttributeRef attributeRef**

*References a [Attribute](#) definition and will be mapped to the global definition if a mapping table of type [OTF2\\_MAPPING\\_ATTRIBUTE](#) is available.*

- **OTF2\_CommRef commRef**

*References a [Comm](#) definition and will be mapped to the global definition if a mapping table of type [OTF2\\_MAPPING\\_COMM](#) is available.*

- float **float32**

*Arbitrary value of type float.*

- double **float64**

*Arbitrary value of type double.*

- **OTF2\_GroupRef groupRef**

*References a [Group](#) definition and will be mapped to the global definition if a mapping table of type [OTF2\\_MAPPING\\_GROUP](#) is available.*

- int16\_t **int16**

*Arbitrary value of type int16\_t.*

- int32\_t **int32**

*Arbitrary value of type int32\_t.*

- int64\_t **int64**

*Arbitrary value of type int64\_t.*

## **APPENDIX I. DATA STRUCTURE DOCUMENTATION**

---

- **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.*
- **OTF2StringRef stringRef**  
*References a [String](#) definition and will be mapped to the global definition if a mapping table of type [OTF2\\_MAPPING\\_STRING](#) is available.*
- **uint16\_t uint16**  
*Arbitrary value of type uint16\_t.*
- **uint32\_t uint32**  
*Arbitrary value of type uint32\_t.*
- **uint64\_t uint64**  
*Arbitrary value of type uint64\_t.*
- **uint8\_t uint8**  
*Arbitrary value of type uint8\_t.*

### **I.1.1 Detailed Description**

Value container for an attributes.

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

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

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

## I.2 OTF2\_FileSionCallbacks Struct Reference

---

### I.2 OTF2\_FileSionCallbacks Struct Reference

Structure holding the SION callbacks.

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

#### Data Fields

- [OTF2\\_FileSionClose otf2\\_file\\_sion\\_close](#)  
*Callback which is called to close a SION file.*
- [OTF2\\_FileSionGetRank otf2\\_file\\_sion\\_get\\_rank](#)  
*Callback which is called to get the MPI rank in read mode.*
- [OTF2\\_FileSionOpen otf2\\_file\\_sion\\_open](#)  
*Callback which is called to open a SION file.*

#### I.2.1 Detailed Description

Structure holding the SION callbacks.

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

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

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

## I.3 OTF2\_FlushCallbacks Struct Reference

Structure holding the flush callbacks.

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

#### Data Fields

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

---

## **APPENDIX I. DATA STRUCTURE DOCUMENTATION**

### **I.3.1 Detailed Description**

Structure holding the flush callbacks.

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

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

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

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

## **I.4 OTF2\_MemoryCallbacks Struct Reference**

Structure holding the memory callbacks.

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

### **Data Fields**

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

### **I.4.1 Detailed Description**

Structure holding the memory callbacks.

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

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

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

## **I.5 OTF2\_MetricValue Union Reference**

Metric value.

### **I.5.1 Detailed Description**

Metric value.

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

## **I.5 OTF2\_MetricValue Union Reference**

---

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

## **APPENDIX I. DATA STRUCTURE DOCUMENTATION**

---

## Appendix J

# File Documentation

### J.1 otf2.h File Reference

Main include file for applications using OTF2.

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

#### J.1.1 Detailed Description

Main include file for applications using OTF2.

#### Maintainer:

Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

#### Authors

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### J.2 OTF2\_Archive.h File Reference

Writing interface for OTF2 archives.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Callbacks.h>
#include <otf2/OTF2_DefWriter.h>
#include <otf2/OTF2_DefReader.h>
```

## **APPENDIX J. FILE DOCUMENTATION**

---

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

### **Defines**

- #define **OTF2\_CHUNK\_SIZE\_DEFINITIONS\_DEFAULT** ( 4 \* 1024 \* 1024 )  
*Default size for OTF2's internal event chunk memory handling.*
- #define **OTF2\_CHUNK\_SIZE\_EVENTS\_DEFAULT** ( 1024 \* 1024 )  
*Default size for OTF2's internal event chunk memory handling.*

### **TypeDefs**

- typedef struct OTF2\_Archive\_struct **OTF2\_Archive**  
*Keeps all meta-data for an OTF2 archive.*
- typedef uint8\_t **OTF2\_MasterSlaveMode**  
*Defines whether a location is master or slave.*

### **Enumerations**

- enum **OTF2\_MasterSlaveMode\_enum** {  
    **OTF2\_SLAVE** = 0,  
    **OTF2\_MASTER** = 1 }  
*Defines whether a location is master or slave.*

## J.2 OTF2\_Archive.h File Reference

---

### Functions

- **OTF2\_ErrorCode OTF2\_Archive\_Close (OTF2\_Archive \*archive)**  
*Close an opened archive.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseDefReader (OTF2\_Archive \*archive, OTF2\_DefReader \*reader)**  
*Close an opened local definition reader.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseDefWriter (OTF2\_Archive \*archive, OTF2\_DefWriter \*writer)**  
*Close an opened local definition writer.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseEvtReader (OTF2\_Archive \*archive, OTF2\_EvtReader \*reader)**  
*Close an opened local event reader.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseEvtWriter (OTF2\_Archive \*archive, OTF2\_EvtWriter \*writer)**  
*Close an opened local event writer.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseGlobalDefReader (OTF2\_Archive \*archive, OTF2\_GlobalDefReader \*globalDefReader)**  
*Closes the global definition reader.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseGlobalEvtReader (OTF2\_Archive \*archive, OTF2\_GlobalEvtReader \*globalEvtReader)**  
*Closes the global event reader.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseGlobalSnapReader (OTF2\_Archive \*archive, OTF2\_GlobalSnapReader \*globalSnapReader)**  
*Close the opened global snapshot reader.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseMarkerReader (OTF2\_Archive \*archive, OTF2\_MarkerReader \*markerReader)**  
*Closes the marker reader.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseMarkerWriter (OTF2\_Archive \*archive, OTF2\_MarkerWriter \*writer)**  
*Close an opened marker writer.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseSnapReader (OTF2\_Archive \*archive, OTF2\_SnapReader \*reader)**  
*Close an opened local snap reader.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseSnapWriter (OTF2\_Archive \*archive, OTF2\_SnapWriter \*writer)**  
*Close an opened local snap writer.*
- **OTF2\_ErrorCode OTF2\_Archive\_CloseThumbReader (OTF2\_Archive \*archive, OTF2\_ThumbReader \*reader)**  
*Close an opened thumbnail reader.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_Archive\_GetChunkSize** (**OTF2\_Archive** \*archive, **uint64\_t** \*chunkSizeEvents, **uint64\_t** \*chunkSizeDefs)  
*Get the chunksize.*
- **OTF2\_ErrorCode OTF2\_Archive\_GetCompression** (**OTF2\_Archive** \*archive, **OTF2\_Compression** \*compression)  
*Get compression mode (none or zlib)*
- **OTF2\_ErrorCode OTF2\_Archive\_GetCreator** (**OTF2\_Archive** \*archive, **char** \*\*creator)  
*Get creator information.*
- **OTF2\_DefReader \* OTF2\_Archive\_GetDefReader** (**OTF2\_Archive** \*archive, **OTF2\_LocationRef** location)  
*Get a local definition reader.*
- **OTF2\_DefWriter \* OTF2\_Archive\_GetDefWriter** (**OTF2\_Archive** \*archive, **OTF2\_LocationRef** location)  
*Get a local definition writer.*
- **OTF2\_ErrorCode OTF2\_Archive\_GetDescription** (**OTF2\_Archive** \*archive, **char** \*\*description)  
*Get description.*
- **OTF2\_EvtReader \* OTF2\_Archive\_GetEvtReader** (**OTF2\_Archive** \*archive, **OTF2\_LocationRef** location)  
*Get a local event reader.*
- **OTF2\_EvtWriter \* OTF2\_Archive\_GetEvtWriter** (**OTF2\_Archive** \*archive, **OTF2\_LocationRef** location)  
*Get a local event writer.*
- **OTF2\_ErrorCode OTF2\_Archive\_GetFileSubstrate** (**OTF2\_Archive** \*archive, **OTF2\_FileSubstrate** \*substrate)  
*Get the file substrate (posix, sion, none)*
- **OTF2\_GlobalDefReader \* OTF2\_Archive\_GetGlobalDefReader** (**OTF2\_Archive** \*archive)  
*Get a global definition reader.*
- **OTF2\_GlobalDefWriter \* OTF2\_Archive\_GetGlobalDefWriter** (**OTF2\_Archive** \*archive)  
*Get a global definition writer.*
- **OTF2\_GlobalEvtReader \* OTF2\_Archive\_GetGlobalEvtReader** (**OTF2\_Archive** \*archive)  
*Get a global event reader.*
- **OTF2\_GlobalSnapReader \* OTF2\_Archive\_GetGlobalSnapReader** (**OTF2\_Archive** \*archive)  
*Get a global snap reader.*

## J.2 OTF2\_Archive.h File Reference

---

- `OTF2_ErrorCode OTF2_Archive_GetMachineName (OTF2_Archive *archive, char **machineName)`  
*Get machine name.*
- `OTF2_MarkerReader * OTF2_Archive_GetMarkerReader (OTF2_Archive *archive)`  
*Get a marker reader.*
- `OTF2_MarkerWriter * OTF2_Archive_GetMarkerWriter (OTF2_Archive *archive)`  
*Get a marker writer.*
- `OTF2_ErrorCode OTF2_Archive_GetMasterSlaveMode (OTF2_Archive *archive, OTF2_MasterSlaveMode *masterOrSlave)`  
*Get master slave mode.*
- `OTF2_ErrorCode OTF2_Archive_GetNumberOfGlobalDefinitions (OTF2_Archive *archive, uint64_t *numberOfDefinitions)`  
*Get the number of global definitions.*
- `OTF2_ErrorCode OTF2_Archive_GetNumberOfLocations (OTF2_Archive *archive, uint64_t *numberOfLocations)`  
*Get the number of locations.*
- `OTF2_ErrorCode OTF2_Archive_GetNumberOfSnapshots (OTF2_Archive *archive, uint32_t *number)`  
*Get the number of snapshots.*
- `OTF2_ErrorCode OTF2_Archive_GetNumberOfThumbnails (OTF2_Archive *archive, uint32_t *number)`  
*Get the number of thumbnails.*
- `OTF2_ErrorCode OTF2_ArchiveGetProperty (OTF2_Archive *archive, const char *name, char **value)`  
*Get the value of the named trace file property.*
- `OTF2_ErrorCode OTF2_ArchiveGetPropertyNames (OTF2_Archive *archive, uint32_t *numberOfProperties, char ***names)`  
*Get the names of all trace file properties.*
- `OTF2_SnapReader * OTF2_Archive_GetSnapReader (OTF2_Archive *archive, OTF2_LocationRef location)`  
*Get a local snap reader.*
- `OTF2_SnapWriter * OTF2_Archive_GetSnapWriter (OTF2_Archive *archive, OTF2_LocationRef location)`  
*Get a local snap writer.*
- `OTF2_ThumbReader * OTF2_Archive_GetThumbReader (OTF2_Archive *archive, uint32_t number)`  
*Get a thumb reader.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ThumbWriter \* OTF2\_Archive\_GetThumbWriter** (**OTF2\_Archive** \*archive, const char \*name, const char \*description, **OTF2\_ThumbnailType** type, uint32\_t numberOfSamples, uint32\_t numberOfMetrics, const uint64\_t \*refsToDefs)

*Get a thumb writer.*

- **OTF2\_ErrorCode OTF2\_Archive\_GetTraceId** (**OTF2\_Archive** \*archive, uint64\_t \*id)

*Get the identifier of the trace file.*

- **OTF2\_ErrorCode OTF2\_Archive\_GetVersion** (**OTF2\_Archive** \*archive, uint8\_t \*major, uint8\_t \*minor, uint8\_t \*bugfix)

*Get format version.*

- **OTF2\_Archive \* OTF2\_Archive\_Open** (const char \*archivePath, const char \*archiveName, const **OTF2\_FileMode** fileMode, const uint64\_t chunkSizeEvents, const uint64\_t chunkSizeDefs, const **OTF2\_FileSubstrate** fileSubstrate, const **OTF2\_Compression** compression)

*Create a new archive.*

- **OTF2\_ErrorCode OTF2\_Archive\_SetBoolProperty** (**OTF2\_Archive** \*archive, const char \*name, bool value, bool overwrite)

*Add or remove a boolean trace file property to this archive.*

- **OTF2\_ErrorCode OTF2\_Archive\_SetCreator** (**OTF2\_Archive** \*archive, const char \*creator)

*Set creator.*

- **OTF2\_ErrorCode OTF2\_Archive\_SetDescription** (**OTF2\_Archive** \*archive, const char \*description)

*Set a description.*

- **OTF2\_ErrorCode OTF2\_Archive\_SetFileSionCallbacks** (**OTF2\_Archive** \*archive, const **OTF2\_FileSionCallbacks** \*fileSionCallbacks, void \*fileSionData)

*Set the SION callbacks for the archive.*

- **OTF2\_ErrorCode OTF2\_Archive\_SetFlushCallbacks** (**OTF2\_Archive** \*archive, const **OTF2\_FlushCallbacks** \*flushCallbacks, void \*flushData)

*Set the flush callbacks for the archive.*

- **OTF2\_ErrorCode OTF2\_Archive\_SetMachineName** (**OTF2\_Archive** \*archive, const char \*machineName)

*Set machine name.*

- **OTF2\_ErrorCode OTF2\_Archive\_SetMasterSlaveMode** (**OTF2\_Archive** \*archive, **OTF2\_MasterSlaveMode** masterOrSlave)

*Set master slave mode.*

- **OTF2\_ErrorCode OTF2\_Archive\_SetMemoryCallbacks** (**OTF2\_Archive** \*archive, const **OTF2\_MemoryCallbacks** \*memoryCallbacks, void \*memoryData)

*Set the memory callbacks for the archive.*

## J.2 OTF2\_Archive.h File Reference

---

- **OTF2\_ErrorCode OTF2\_Archive\_SetNumberOfSnapshots (OTF2\_Archive \*archive, uint32\_t number)**  
*Set the number of snapshots.*
- **OTF2\_ErrorCode OTF2\_Archive\_SetProperty (OTF2\_Archive \*archive, const char \*name, const char \*value, bool overwrite)**  
*Add or remove a trace file property to this archive.*
- **OTF2\_ErrorCode OTF2\_Archive\_Switch FileMode (OTF2\_Archive \*archive, OTF2\_FileMode new FileMode)**  
*Switch file mode of the archive.*

### J.2.1 Detailed Description

Writing interface for OTF2 archives.

#### Maintainer:

Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

#### Authors

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### J.2.2 Define Documentation

#### J.2.2.1 #define OTF2\_CHUNK\_SIZE\_DEFINITIONS\_DEFAULT ( 4 \* 1024 \* 1024 )

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

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

#### J.2.2.2 #define OTF2\_CHUNK\_SIZE\_EVENTS\_DEFAULT ( 1024 \* 1024 )

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

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

### J.2.3 Typedef Documentation

#### J.2.3.1 typedef struct OTF2\_Archive struct OTF2\_Archive

Keeps all meta-data for an OTF2 archive.

## **APPENDIX J. FILE DOCUMENTATION**

---

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

### **J.2.3.2 `typedef uint8_t OTF2_MasterSlaveMode`**

Defines whether a location is master or slave.

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

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

### **J.2.4 Enumeration Type Documentation**

#### **J.2.4.1 `enum OTF2_MasterSlaveMode_enum`**

Defines whether a location is master or slave.

**Enumerator:**

***OTF2\_SLAVE*** Location is slave.

***OTF2\_MASTER*** Location is master.

### **J.2.5 Function Documentation**

#### **J.2.5.1 `OTF2_ErrorCode OTF2_Archive_Close( OTF2_Archive *archive )`**

Close an opened archive.

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

**Parameters**

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

**Returns**

***OTF2\_SUCCESS*** if successful, an error code if an error occurs.

## J.2 OTF2\_Archive.h File Reference

---

### J.2.5.2 OTF2\_ErrorCode OTF2\_Archive\_CloseDefReader ( OTF2\_Archive \* archive, OTF2\_DefReader \* reader )

Close an opened local definition reader.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.2.5.3 OTF2\_ErrorCode OTF2\_Archive\_CloseDefWriter ( OTF2\_Archive \* archive, OTF2\_DefWriter \* writer )

Close an opened local definition writer.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.2.5.4 OTF2\_ErrorCode OTF2\_Archive\_CloseEvtReader ( OTF2\_Archive \* archive, OTF2\_EvtReader \* reader )

Close an opened local event reader.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **J.2.5.5 OTF2\_ErrorCode OTF2\_Archive\_CloseEvtWriter ( OTF2\_Archive \* archive, OTF2\_EvtWriter \* writer )**

Close an opened local event writer.

#### **Parameters**

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

#### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### **J.2.5.6 OTF2\_ErrorCode OTF2\_Archive\_CloseGlobalDefReader ( OTF2\_Archive \* archive, OTF2\_GlobalDefReader \* globalDefReader )**

Closes the global definition reader.

#### **Parameters**

<i>archive</i>	Archive handle.
<i>globalDef- Reader</i>	The global definition reader.

#### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### **J.2.5.7 OTF2\_ErrorCode OTF2\_Archive\_CloseGlobalEvtReader ( OTF2\_Archive \* archive, OTF2\_GlobalEvtReader \* globalEvtReader )**

Closes the global event reader.

This closes also all local event readers.

#### **Parameters**

<i>archive</i>	Archive handle.
<i>glob- alEvtReader</i>	The global event reader.

## J.2 OTF2\_Archive.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.2.5.8 OTF2\_ErrorCode OTF2\_Archive\_CloseGlobalSnapReader ( OTF2\_Archive \* archive, OTF2\_GlobalSnapReader \* globalSnapReader )

Close the opened global snapshot reader.

### Parameters

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

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.2.5.9 OTF2\_ErrorCode OTF2\_Archive\_CloseMarkerReader ( OTF2\_Archive \* archive, OTF2\_MarkerReader \* markerReader )

Closes the marker reader.

### Parameters

<i>archive</i>	Archive handle.
<i>markerReader</i>	The marker reader.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## **APPENDIX J. FILE DOCUMENTATION**

---

**J.2.5.10 OTF2\_ErrorCode OTF2\_Archive\_CloseMarkerWriter ( OTF2\_Archive \*  
archive, OTF2\_MarkerWriter \* writer )**

Close an opened marker writer.

### **Parameters**

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

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.11 OTF2\_ErrorCode OTF2\_Archive\_CloseSnapReader ( OTF2\_Archive \*  
archive, OTF2\_SnapReader \* reader )**

Close an opened local snap reader.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### **Since**

Version 1.2

**J.2.5.12 OTF2\_ErrorCode OTF2\_Archive\_CloseSnapWriter ( OTF2\_Archive \*  
archive, OTF2\_SnapWriter \* writer )**

Close an opened local snap writer.

### **Parameters**

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

## J.2 OTF2\_Archive.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.2.5.13 OTF2\_ErrorCode OTF2\_Archive\_CloseThumbReader ( OTF2\_Archive \* archive, OTF2\_ThumbReader \* reader )

Close an opened thumbnail reader.

### Parameters

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

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.2.5.14 OTF2\_ErrorCode OTF2\_Archive\_GetChunkSize ( OTF2\_Archive \* archive, uint64\_t \* chunkSizeEvents, uint64\_t \* chunkSizeDefs )

Get the chunksize.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **J.2.5.15 OTF2\_ErrorCode OTF2\_Archive\_GetCompression ( OTF2\_Archive \* archive, OTF2\_Compression \* compression )**

Get compression mode (none or zlib)

#### **Parameters**

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

#### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### **J.2.5.16 OTF2\_ErrorCode OTF2\_Archive\_GetCreator ( OTF2\_Archive \* archive, char \*\* creator )**

Get creator information.

#### **Parameters**

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

#### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### **J.2.5.17 OTF2\_DefReader\* OTF2\_Archive\_GetDefReader ( OTF2\_Archive \* archive, OTF2\_LocationRef location )**

Get a local definition reader.

#### **Parameters**

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested reader handle.

#### **Returns**

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

## J.2 OTF2\_Archive.h File Reference

---

**J.2.5.18 OTF2\_DefWriter\* OTF2\_Archive\_GetDefWriter ( OTF2\_Archive \* *archive*, OTF2\_LocationRef *location* )**

Get a local definition writer.

### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested writer handle.

### Returns

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

**J.2.5.19 OTF2\_ErrorCode OTF2\_Archive\_GetDescription ( OTF2\_Archive \* *archive*, char \*\* *description* )**

Get description.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.20 OTF2\_EvtReader\* OTF2\_Archive\_GetEvtReader ( OTF2\_Archive \* *archive*, OTF2\_LocationRef *location* )**

Get a local event reader.

### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested reader handle.

### Returns

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

## **APPENDIX J. FILE DOCUMENTATION**

---

**J.2.5.21 OTF2\_EvtWriter\* OTF2\_Archive\_GetEvtWriter ( OTF2\_Archive \* *archive*, OTF2\_LocationRef *location* )**

Get a local event writer.

### **Parameters**

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested writer handle.

### **Returns**

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

**J.2.5.22 OTF2\_ErrorCode OTF2\_Archive\_GetFileSubstrate ( OTF2\_Archive \* *archive*, OTF2\_FileSubstrate \* *substrate* )**

Get the file substrate (posix, sion, none)

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.23 OTF2\_GlobalDefReader\* OTF2\_Archive\_GetGlobalDefReader ( OTF2\_Archive \* *archive* )**

Get a global definition reader.

### **Parameters**

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

### **Returns**

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

## J.2 OTF2\_Archive.h File Reference

---

**J.2.5.24 OTF2\_GlobalDefWriter\* OTF2\_Archive\_GetGlobalDefWriter ( OTF2\_Archive \* *archive* )**

Get a global definition writer.

### Parameters

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

### Returns

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

**J.2.5.25 OTF2\_GlobalEvtReader\* OTF2\_Archive\_GetGlobalEvtReader ( OTF2\_Archive \* *archive* )**

Get a global event reader.

### Parameters

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

### Returns

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

**J.2.5.26 OTF2\_GlobalSnapReader\* OTF2\_Archive\_GetGlobalSnapReader ( OTF2\_Archive \* *archive* )**

Get a global snap reader.

### Parameters

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

### Since

Version 1.2

### Returns

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

## **APPENDIX J. FILE DOCUMENTATION**

---

**J.2.5.27 OTF2\_ErrorCode OTF2\_Archive\_GetMachineName ( OTF2\_Archive \*  
archive, char \*\* machineName )**

Get machine name.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.28 OTF2\_MarkerReader\* OTF2\_Archive\_GetMarkerReader ( OTF2\_Archive  
\* archive )**

Get a marker reader.

### **Parameters**

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

### **Since**

Version 1.2

### **Returns**

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

**J.2.5.29 OTF2\_MarkerWriter\* OTF2\_Archive\_GetMarkerWriter ( OTF2\_Archive \*  
archive )**

Get a marker writer.

### **Parameters**

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

### **Since**

Version 1.2

## J.2 OTF2\_Archive.h File Reference

---

### Returns

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

**J.2.5.30 OTF2\_ErrorCode OTF2\_Archive\_GetMasterSlaveMode ( OTF2\_Archive \* archive, OTF2\_MasterSlaveMode \* masterOrSlave )**

Get master slave mode.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.31 OTF2\_ErrorCode OTF2\_Archive\_GetNumberOfGlobalDefinitions ( OTF2\_Archive \* archive, uint64\_t \* numberOfDefinitions )**

Get the number of global definitions.

### Parameters

	<i>archive</i>	Archive handle.
out	<i>num- berOfDefi- nitions</i>	Return pointer to the number of global definitions.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.32 OTF2\_ErrorCode OTF2\_Archive\_GetNumberOfLocations ( OTF2\_Archive \* archive, uint64\_t \* numberOfLocations )**

Get the number of locations.

### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

<code>out</code>	<code>num- berOfLoca- tions</code>	Return pointer to the number of locations.
------------------	--	--

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.33 OTF2\_ErrorCode OTF2\_Archive\_GetNumberOfSnapshots ( OTF2\_Archive \* archive, uint32\_t \* number )**

Get the number of snapshots.

### **Parameters**

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

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.34 OTF2\_ErrorCode OTF2\_Archive\_GetNumberOfThumbnails ( OTF2\_Archive \* archive, uint32\_t \* number )**

Get the number of thumbnails.

### **Parameters**

<code>archive</code>	Archive handle.
<code>number</code>	Thumb number.

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.2 OTF2\_Archive.h File Reference

---

**J.2.5.35 OTF2\_ErrorCode OTF2\_ArchiveGetProperty ( OTF2\_Archive \* archive,  
const char \* name, char \*\* value )**

Get the value of the named trace file property.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_PROPERTY\_NOT\_FOUND* if the named property was not found

**J.2.5.36 OTF2\_ErrorCode OTF2\_ArchiveGetPropertyNames ( OTF2\_Archive \*  
archive, uint32\_t \* numberOfProperties, char \*\*\* names )**

Get the names of all trace file properties.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.37 OTF2\_SnapReader\* OTF2\_Archive\_GetSnapReader ( OTF2\_Archive \*  
archive, OTF2\_LocationRef location )**

Get a local snap reader.

### Parameters

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested snap handle.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.2

### **Returns**

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

**J.2.5.38 OTF2\_SnapWriter\* OTF2\_Archive\_GetSnapWriter ( OTF2\_Archive \*  
archive, OTF2\_LocationRef location )**

Get a local snap writer.

### **Parameters**

<i>archive</i>	Archive handle.
<i>location</i>	Location ID of the requested writer handle.

### **Since**

Version 1.2

### **Returns**

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

**J.2.5.39 OTF2\_ThumbReader\* OTF2\_Archive\_GetThumbReader ( OTF2\_Archive \*  
archive, uint32\_t number )**

Get a thumb reader.

### **Parameters**

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

### **Since**

Version 1.2

### **Returns**

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

## J.2 OTF2\_Archive.h File Reference

---

**J.2.5.40** `OTF2_ThumbWriter* OTF2_Archive_GetThumbWriter ( OTF2_Archive * archive, const char * name, const char * description, OTF2_ThumbnailType type, uint32_t numberOfSamples, uint32_t numberOfMetrics, const uint64_t * refsToDefs )`

Get a thumb writer.

### Parameters

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

### Since

Version 1.2

### Returns

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

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

Get the identifier of the trace file.

### Note

This call is only allowed when the archive was opened with mode OTF2\_FILEMODE\_READ.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## APPENDIX J. FILE DOCUMENTATION

---

**J.2.5.42 OTF2\_ErrorCode OTF2\_Archive\_GetVersion ( OTF2\_Archive \* archive,  
                  uint8\_t \* major, uint8\_t \* minor, uint8\_t \* bugfix )**

Get format version.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.43 OTF2\_Archive\* OTF2\_Archive\_Open ( const char \* archivePath, const  
                  char \* archiveName, const OTF2\_FileMode fileMode, const uint64\_t  
                  chunkSizeEvents, const uint64\_t chunkSizeDefs, const OTF2\_FileSubstrate  
                  fileSubstrate, const OTF2\_Compression compression )**

Create a new archive.

Creates a new archive handle that keeps all meta data about the archive on runtime.

### Parameters

<i>archivePath</i>	Path to the archive i.e. the directory where the anchor file is located.
<i>archive- Name</i>	Name of the archive. It is used to generate sub pathes e.g. 'archive- Name.otf2'.
<i>fileMode</i>	Determines if in reading or writing mode. Available values are <i>OTF2_- FILEMODE_WRITE</i> or <i>OTF2_FILEMODE_READ</i> .
<i>chunk- SizeEvents</i>	Requested size of OTF2's internal event chunks in writing mode. Available values are from 256kB to 16MB. The event chunk size affects performance as well as total memory usage. A value satisfying both is about 1MB. If you are not sure which chunk size is the best to use, use <i>OTF2_CHUNK_SIZE_EVENTS_DEFAULT</i> . In reading mode this value is ignored because the correct chunk size is extracted from the anchor file.

## J.2 OTF2\_Archive.h File Reference

---

<i>chunk-SizeDefs</i>	Requested size of OTF2's internal definition chunks in writing mode. Available values are from 256kB to 16MB. The definition chunk size affects performance as well as total memory usage. In addition, the definition chunk size must be big enough to carry the largest possible definition record. Therefore, the definition chunk size must be at least 10 times the number of locations. A value satisfying these requirements is about 4MB. If you are not sure which chunk size is the best to use, use <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.

### Returns

Returns an archive handle if successful, NULL otherwise.

#### J.2.5.44 OTF2\_ErrorCode OTF2\_Archive\_SetBoolProperty ( OTF2\_Archive \* *archive*, const char \* *name*, bool *value*, bool *overwrite* )

Add or remove a boolean trace file property to this archive.

### Note

This call is only allowed when the archive was opened with mode [OTF2\\_FILEMODE\\_WRITE](#).

### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_PROPERTY\_NAME\_INVALID* if property name does not conform to the naming scheme

*OTF2\_ERROR\_PROPERTY\_NOT\_FOUND* if property was not found, but requested to remove

*OTF2\_ERROR\_PROPERTY\_EXISTS* if property exists but overwrite was not set

#### **J.2.5.45 OTF2\_ErrorCode OTF2\_Archive\_SetCreator ( OTF2\_Archive \* archive, const char \* creator )**

Set creator.

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

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.2.5.46 OTF2\_ErrorCode OTF2\_Archive\_SetDescription ( OTF2\_Archive \* archive, const char \* description )**

Set a description.

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

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.2 OTF2\_Archive.h File Reference

---

**J.2.5.47 OTF2\_ErrorCode OTF2\_Archive\_SetFileSionCallbacks ( OTF2\_Archive \*  
archive, const OTF2\_FileSionCallbacks \* fileSionCallbacks, void \*  
fileSionData )**

Set the SION callbacks for the archive.

### Parameters

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

### Returns

OTF2\_ErrorCode, or error code.

**J.2.5.48 OTF2\_ErrorCode OTF2\_Archive\_SetFlushCallbacks ( OTF2\_Archive \*  
archive, const OTF2\_FlushCallbacks \* flushCallbacks, void \* flushData )**

Set the flush callbacks for the archive.

### Parameters

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

### Returns

OTF2\_ErrorCode, or error code.

**J.2.5.49 OTF2\_ErrorCode OTF2\_Archive\_SetMachineName ( OTF2\_Archive \*  
archive, const char \* machineName )**

Set machine name.

Sets the name for the machine the trace was recorded. This value is optional. It only needs to be set for an archive handle marked as 'master' or does not need to be set at all.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.2.5.50 OTF2\_ErrorCode OTF2\_Archive\_SetMasterSlaveMode ( OTF2\_Archive \* archive, OTF2\_MasterSlaveMode masterOrSlave )**

Set master slave mode.

Sets master slave mode for a location. If OTF2\_MASTER is passed, the location creates the directory structure for the trace files to store. Therefore, exactly one location can be master, all other locations must be slaves.

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

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.2.5.51 OTF2\_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 <i>userData</i> argument.

## J.2 OTF2\_Archive.h File Reference

---

### Returns

OTF2\_ErrorCode, or error code.

**J.2.5.52 OTF2\_ErrorCode OTF2\_Archive\_SetNumberOfSnapshots ( OTF2\_Archive \* archive, uint32\_t number )**

Set the number of snapshots.

### Parameters

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

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.2.5.53 OTF2\_ErrorCode OTF2\_Archive\_SetProperty ( OTF2\_Archive \* archive, const char \* name, const char \* value, bool overwrite )**

Add or remove a trace file property to this archive.

Removing a trace file property is done by passing "" in the *value* parameter. The *overwrite* parameter is ignored than.

### 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 will be overwritten.

**Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_PROPERTY\_NAME\_INVALID* if property name does not conform to the naming scheme

*OTF2\_ERROR\_PROPERTY\_NOT\_FOUND* if property was not found, but requested to remove

*OTF2\_ERROR\_PROPERTY\_EXISTS* if property exists but overwrite was not set

**J.2.5.54 OTF2\_ErrorCode OTF2\_Archive\_Switch FileMode ( OTF2\_Archive \* archive, OTF2\_FileMode new FileMode )**

Switch file mode of the archive.

Currently only a switch from *OTF2\_FILEMODE\_READ* to *OTF2\_FILEMODE\_WRITE* is permitted and in this case, the master/slave mode is reset and must be set again with *OTF2\_Archive\_SetMasterSlaveMode*. Currently it is also only permitted when operating on an OTF2 archive with the *OTF2\_SUBSTRATE\_POSIX* file substrate.

**Parameters**

<i>archive</i>	Archive handle.
<i>new FileMode</i>	New <i>OTF2_FileMode</i> to switch to.

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**Since**

Version 1.2

**J.3 OTF2\_AttributeList.h File Reference**

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

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

## J.3 OTF2\_AttributeList.h File Reference

---

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

### Data Structures

- union [OTF2\\_AttributeValue](#)  
*Value container for an attributes.*

### Typedefs

- typedef struct OTF2\_AttributeList\_struct [OTF2\\_AttributeList](#)  
*Attribute list handle.*

### Functions

- [OTF2\\_ErrorCode OTF2\\_AttributeList\\_AddAttribute](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_Type](#) type, [OTF2\\_AttributeValue](#) attributeValue)  
*Add an attribute to an attribute list.*
- [OTF2\\_ErrorCode OTF2\\_AttributeList\\_AddAttributeRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_AttributeRef](#) attributeRef)  
*Add an OTF2\_TYPE\_ATTRIBUTE attribute to an attribute list.*
- [OTF2\\_ErrorCode OTF2\\_AttributeList\\_AddCommRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_CommRef](#) commRef)  
*Add an OTF2\_TYPE\_COMM attribute to an attribute list.*
- [OTF2\\_ErrorCode OTF2\\_AttributeList\\_AddDouble](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, double float64Value)  
*Add an OTF2\_TYPE\_DOUBLE attribute to an attribute list.*
- [OTF2\\_ErrorCode OTF2\\_AttributeList\\_AddFloat](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, float float32Value)  
*Add an OTF2\_TYPE\_FLOAT attribute to an attribute list.*
- [OTF2\\_ErrorCode OTF2\\_AttributeList\\_AddGroupRef](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, [OTF2\\_GroupRef](#) groupRef)  
*Add an OTF2\_TYPE\_GROUP attribute to an attribute list.*
- [OTF2\\_ErrorCode OTF2\\_AttributeList\\_AddInt16](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, int16\_t int16Value)  
*Add an OTF2\_TYPE\_INT16 attribute to an attribute list.*
- [OTF2\\_ErrorCode OTF2\\_AttributeList\\_AddInt32](#) ([OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_AttributeRef](#) attribute, int32\_t int32Value)  
*Add an OTF2\_TYPE\_INT32 attribute to an attribute list.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- OTF2\_ErrorCode OTF2\_AttributeList\_AddInt64 (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, int64\_t int64Value)

*Add an OTF2\_TYPE\_INT64 attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddInt8 (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, int8\_t int8Value)

*Add an OTF2\_TYPE\_INT8 attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddLocationRef (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, OTF2\_LocationRef locationRef)

*Add an OTF2\_TYPE\_LOCATION attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddMetricRef (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, OTF2\_MetricRef metricRef)

*Add an OTF2\_TYPE\_METRIC attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddParameterRef (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, OTF2\_ParameterRef parameterRef)

*Add an OTF2\_TYPE\_PARAMETER attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddRegionRef (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, OTF2\_RegionRef regionRef)

*Add an OTF2\_TYPE\_REGION attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddRmaWinRef (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, OTF2\_RmaWinRef rmaWinRef)

*Add an OTF2\_TYPE\_RMA\_WIN attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddString (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, OTF2StringRef stringRef)

*Add an OTF2\_STRING attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddStringRef (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, OTF2StringRef stringRef)

*Add an OTF2\_TYPE\_STRING attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddUint16 (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, uint16\_t uint16Value)

*Add an OTF2\_TYPE\_UINT16 attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddUint32 (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, uint32\_t uint32Value)

*Add an OTF2\_TYPE\_UINT32 attribute to an attribute list.*

- OTF2\_ErrorCode OTF2\_AttributeList\_AddUint64 (OTF2\_AttributeList \*attributeList, OTF2\_AttributeRef attribute, uint64\_t uint64Value)

*Add an OTF2\_TYPE\_UINT64 attribute to an attribute list.*

### J.3 OTF2\_AttributeList.h File Reference

---

- `OTF2_ErrorCode OTF2_AttributeList_AddUInt8 (OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, uint8_t uint8Value)`

*Add an OTF2\_TYPE\_UINT8 attribute to an attribute list.*
- `OTF2_ErrorCode OTF2_AttributeList_Delete (OTF2_AttributeList *attributeList)`

*Delete an attribute list handle.*
- `OTF2_ErrorCode OTF2_AttributeList_GetAttributeByID (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, OTF2_Type *type, OTF2_AttributeValue *attributeValue)`

*Get an attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList_GetAttributeByIndex (const OTF2_AttributeList *attributeList, uint32_t index, OTF2_AttributeRef *attribute, OTF2_Type *type, OTF2_AttributeValue *attributeValue)`

*Get an attribute from an attribute list by attribute index.*
- `OTF2_ErrorCode OTF2_AttributeList_GetAttributeRef (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, OTF2_AttributeRef *attributeRef)`

*Get an OTF2\_TYPE\_ATTRIBUTE attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList_GetCommRef (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, OTF2_CommRef *commRef)`

*Get an OTF2\_TYPE\_COMM attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList_GetDouble (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, double *float64Value)`

*Get an OTF2\_TYPE\_DOUBLE attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList_GetFloat (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, float *float32Value)`

*Get an OTF2\_TYPE\_FLOAT attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList_GetGroupRef (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, OTF2_GroupRef *groupRef)`

*Get an OTF2\_TYPE\_GROUP attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList.GetInt16 (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, int16_t *int16Value)`

*Get an OTF2\_TYPE\_INT16 attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList.GetInt32 (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, int32_t *int32Value)`

*Get an OTF2\_TYPE\_INT32 attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList.GetInt64 (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, int64_t *int64Value)`

---

## **APPENDIX J. FILE DOCUMENTATION**

*Get an OTF2\_TYPE\_INT64 attribute from an attribute list by attribute ID.*

- **OTF2\_ErrorCode OTF2\_AttributeList\_GetInt8** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **int8\_t** \*int8Value)

*Get an OTF2\_TYPE\_INT8 attribute from an attribute list by attribute ID.*

- **OTF2\_ErrorCode OTF2\_AttributeList\_GetLocationRef** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **OTF2\_LocationRef** \*locationRef)

*Get an OTF2\_TYPE\_LOCATION attribute from an attribute list by attribute ID.*

- **OTF2\_ErrorCode OTF2\_AttributeList\_GetMetricRef** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **OTF2\_MetricRef** \*metricRef)

*Get an OTF2\_TYPE\_METRIC attribute from an attribute list by attribute ID.*

- **uint32\_t OTF2\_AttributeList\_GetNumberOfElements** (const **OTF2\_AttributeList** \*attributeList)

*Get the number of entries in an attribute list.*

- **OTF2\_ErrorCode OTF2\_AttributeList\_GetParameterRef** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **OTF2\_ParameterRef** \*parameterRef)

*Get an OTF2\_TYPE\_PARAMETER attribute from an attribute list by attribute ID.*

- **OTF2\_ErrorCode OTF2\_AttributeList\_GetRegionRef** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **OTF2\_RegionRef** \*regionRef)

*Get an OTF2\_TYPE\_REGION attribute from an attribute list by attribute ID.*

- **OTF2\_ErrorCode OTF2\_AttributeList\_GetRmaWinRef** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **OTF2\_RmaWinRef** \*rmaWinRef)

*Get an OTF2\_TYPE\_RMA\_WIN attribute from an attribute list by attribute ID.*

- **OTF2\_ErrorCode OTF2\_AttributeListGetString** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **OTF2StringRef** \*stringRef)

*Add an OTF2\_STRING attribute to an attribute list.*

- **OTF2\_ErrorCode OTF2\_AttributeListGetStringRef** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **OTF2StringRef** \*stringRef)

*Get an OTF2\_TYPE\_STRING attribute from an attribute list by attribute ID.*

- **OTF2\_ErrorCode OTF2\_AttributeListGetUInt16** (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\_AttributeListGetUInt32** (const **OTF2\_AttributeList** \*attributeList, **OTF2\_AttributeRef** attribute, **uint32\_t** \*uint32Value)

*Get an OTF2\_TYPE\_UINT32 attribute from an attribute list by attribute ID.*

## J.3 OTF2\_AttributeList.h File Reference

---

- `OTF2_ErrorCode OTF2_AttributeList_GetUInt64 (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, uint64_t *uint64Value)`  
*Get an OTF2\_TYPE\_UINT64 attribute from an attribute list by attribute ID.*
- `OTF2_ErrorCode OTF2_AttributeList_GetUInt8 (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute, uint8_t *uint8Value)`  
*Get an OTF2\_TYPE\_UINT8 attribute from an attribute list by attribute ID.*
- `OTF2_AttributeList * OTF2_AttributeList_New (void)`  
*Create a new attribute list handle.*
- `OTF2_ErrorCode OTF2_AttributeList_PopAttribute (OTF2_AttributeList *attributeList, OTF2_AttributeRef *attribute, OTF2_Type *type, OTF2_AttributeValue *attributeValue)`  
*Get first attribute from an attribute list and remove it.*
- `OTF2_ErrorCode OTF2_AttributeList_RemoveAllAttributes (OTF2_AttributeList *attributeList)`  
*Remove all attributes from an attribute list.*
- `OTF2_ErrorCode OTF2_AttributeList_RemoveAttribute (OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute)`  
*Remove an attribute from an attribute list.*
- `bool OTF2_AttributeList_TestAttributeByID (const OTF2_AttributeList *attributeList, OTF2_AttributeRef attribute)`  
*Test if an attribute is in the attribute list.*

### J.3.1 Detailed Description

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

#### Source Template:

`template/OTF2_AttributeList tmpl.h`

#### Maintainer:

Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

#### Authors

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

## **APPENDIX J. FILE DOCUMENTATION**

---

### **J.3.2 How to use the attribute list for writing**

additional attributes to event records.

First create an attribute list handle.

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

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

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

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

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

### **J.3.3 Function Documentation**

#### **J.3.3.1 OTF2\_ErrorCode OTF2\_AttributeList\_AddAttribute ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, OTF2\_Type type, OTF2\_AttributeValue attributeValue )**

Add an attribute to an attribute list.

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

#### **Parameters**

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

## J.3 OTF2\_AttributeList.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.2 OTF2\_ErrorCode OTF2\_AttributeList\_AddAttributeRef ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, OTF2\_AttributeRef attributeRef )**

Add an OTF2\_TYPE\_ATTRIBUTE attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.3 OTF2\_ErrorCode OTF2\_AttributeList\_AddCommRef ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, OTF2\_CommRef commRef )**

Add an OTF2\_TYPE\_COMM attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.4 OTF2\_ErrorCode OTF2\_AttributeList\_AddDouble ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, double float64Value )**

Add an OTF2\_TYPE\_DOUBLE attribute to an attribute list.

## **APPENDIX J. FILE DOCUMENTATION**

---

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.3.3.5 OTF2\_ErrorCode OTF2\_AttributeList\_AddFloat ( OTF2\_AttributeList \* \* *attributeList*, OTF2\_AttributeRef *attribute*, float *float32Value* )**

Add an OTF2\_TYPE\_FLOAT attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.3.3.6 OTF2\_ErrorCode OTF2\_AttributeList\_AddGroupRef ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_GroupRef *groupRef* )**

Add an OTF2\_TYPE\_GROUP attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### **Parameters**

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

## J.3 OTF2\_AttributeList.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.3.3.7 OTF2\_ErrorCode OTF2\_AttributeList\_AddInt16 ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, int16\_t int16Value )

Add an OTF2\_TYPE\_INT16 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.3.3.8 OTF2\_ErrorCode OTF2\_AttributeList\_AddInt32 ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, int32\_t int32Value )

Add an OTF2\_TYPE\_INT32 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.3.3.9 OTF2\_ErrorCode OTF2\_AttributeList\_AddInt64 ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, int64\_t int64Value )

Add an OTF2\_TYPE\_INT64 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.3.3.10 OTF2\_ErrorCode OTF2\_AttributeList\_AddInt8 ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, int8\_t int8Value )**

Add an OTF2\_TYPE\_INT8 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.3.3.11 OTF2\_ErrorCode OTF2\_AttributeList\_AddLocationRef ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, OTF2\_LocationRef locationRef )**

Add an OTF2\_TYPE\_LOCATION attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.3 OTF2\_AttributeList.h File Reference

---

J.3.3.12 **OTF2\_ErrorCode OTF2\_AttributeList\_AddMetricRef ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_MetricRef *metricRef* )**

Add an OTF2\_TYPE\_METRIC attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

J.3.3.13 **OTF2\_ErrorCode OTF2\_AttributeList\_AddParameterRef ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_ParameterRef *parameterRef* )**

Add an OTF2\_TYPE\_PARAMETER attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

J.3.3.14 **OTF2\_ErrorCode OTF2\_AttributeList\_AddRegionRef ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_RegionRef *regionRef* )**

Add an OTF2\_TYPE\_REGION attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.15 OTF2\_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.

**J.3.3.16 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.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### **Parameters**

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

## J.3 OTF2\_AttributeList.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.3.3.17 **OTF2\_ErrorCode OTF2\_AttributeList\_AddStringRef ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, OTF2StringRef stringRef )**

Add an OTF2\_TYPE\_STRING attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.3.3.18 **OTF2\_ErrorCode OTF2\_AttributeList\_AddUint16 ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, uint16\_t uint16Value )**

Add an OTF2\_TYPE\_UINT16 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.3.3.19 **OTF2\_ErrorCode OTF2\_AttributeList\_AddUint32 ( OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute, uint32\_t uint32Value )**

Add an OTF2\_TYPE\_UINT32 attribute to an attribute list.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.3.3.20 OTF2\_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.

#### **J.3.3.21 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.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.3 OTF2\_AttributeList.h File Reference

---

### J.3.3.22 OTF2\_ErrorCode OTF2\_AttributeList\_Delete ( OTF2\_AttributeList \* *attributeList* )

Delete an attribute list handle.

Deletes an attribute list handle and releases all associated resources.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.3.3.23 OTF2\_ErrorCode OTF2\_AttributeList\_GetAttributeByID ( const OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_Type \* *type*, OTF2\_AttributeValue \* *attributeValue* )

Get an attribute from an attribute list by attribute ID.

#### Parameters

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

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.3.3.24 OTF2\_ErrorCode OTF2\_AttributeList\_GetAttributeByIndex ( const OTF2\_AttributeList \* *attributeList*, uint32\_t *index*, OTF2\_AttributeRef \* *attribute*, OTF2\_Type \* *type*, OTF2\_AttributeValue \* *attributeValue* )

Get an attribute from an attribute list by attribute index.

#### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

out	<i>type</i>	Returned type of the attribute.
out	<i>attribute-Value</i>	Returned value of the attribute.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.25 OTF2\_ErrorCode OTF2\_AttributeList\_GetAttributeRef ( const  
OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*,  
OTF2\_AttributeRef \* *attributeRef* )**

Get an OTF2\_TYPE\_ATTRIBUTE attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.26 OTF2\_ErrorCode OTF2\_AttributeList\_GetCommRef ( const  
OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*,  
OTF2\_CommRef \* *commRef* )**

Get an OTF2\_TYPE\_COMM attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.3 OTF2\_AttributeList.h File Reference

---

J.3.3.27 **OTF2\_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.

J.3.3.28 **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.

J.3.3.29 **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*.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.30 OTF2\_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.

**J.3.3.31 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*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.3 OTF2\_AttributeList.h File Reference

---

**J.3.3.32 OTF2\_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.

**J.3.3.33 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.

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

## **APPENDIX J. FILE DOCUMENTATION**

---

	<i>attribute</i>	Reference to attribute definition.
out	<i>locationRef</i>	Returned location value.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.35   OTF2\_ErrorCode OTF2\_AttributeList\_GetMetricRef ( const  
OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*,  
OTF2\_MetricRef \* *metricRef* )**

Get an OTF2\_TYPE\_METRIC attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.36   uint32\_t OTF2\_AttributeList\_GetNumberOfElements ( const  
OTF2\_AttributeList \* *attributeList* )**

Get the number of entries in an attribute list.

### **Parameters**

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

### **Returns**

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

## J.3 OTF2\_AttributeList.h File Reference

---

J.3.3.37 **OTF2\_ErrorCode OTF2\_AttributeList\_GetParameterRef ( const OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_ParameterRef \* *parameterRef* )**

Get an OTF2\_TYPE\_PARAMETER attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

J.3.3.38 **OTF2\_ErrorCode OTF2\_AttributeList\_GetRegionRef ( const OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_RegionRef \* *regionRef* )**

Get an OTF2\_TYPE\_REGION attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

J.3.3.39 **OTF2\_ErrorCode OTF2\_AttributeList\_GetRmaWinRef ( const OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, OTF2\_RmaWinRef \* *rmaWinRef* )**

Get an OTF2\_TYPE\_RMA\_WIN attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.40 OTF2\_ErrorCode OTF2\_AttributeList\_GetString ( const  
OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*,  
OTF2StringRef \* *stringRef* )**

Add an OTF2\_STRING attribute to an attribute list.

### **Deprecated**

Use *OTF2\_AttributeList\_GetStringRef()* instead.

Convenient function around *OTF2\_AttributeList\_AddAttribute*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.41 OTF2\_ErrorCode OTF2\_AttributeList\_GetStringRef ( const  
OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*,  
OTF2StringRef \* *stringRef* )**

Get an OTF2\_TYPE\_STRING attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### **Parameters**

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

## J.3 OTF2\_AttributeList.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.42 OTF2\_ErrorCode OTF2\_AttributeList\_GetUint16 ( const  
OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute,  
uint16\_t \* uint16Value )**

Get an OTF2\_TYPE\_UINT16 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.43 OTF2\_ErrorCode OTF2\_AttributeList\_GetUint32 ( const  
OTF2\_AttributeList \* attributeList, OTF2\_AttributeRef attribute,  
uint32\_t \* uint32Value )**

Get an OTF2\_TYPE\_UINT32 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## **APPENDIX J. FILE DOCUMENTATION**

---

**J.3.3.44 OTF2\_ErrorCode OTF2\_AttributeList\_GetUint64 ( const  
OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*,  
uint64\_t \* *uint64Value* )**

Get an OTF2\_TYPE\_UINT64 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.45 OTF2\_ErrorCode OTF2\_AttributeList\_GetUint8 ( const  
OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute*, uint8\_t  
\* *uint8Value* )**

Get an OTF2\_TYPE\_UINT8 attribute from an attribute list by attribute ID.

Convenient function around *OTF2\_AttributeList\_GetAttributeByID*.

### **Parameters**

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

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.46 OTF2\_AttributeList\* OTF2\_AttributeList\_New ( void )**

Create a new attribute list handle.

### **Returns**

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

## J.3 OTF2\_AttributeList.h File Reference

---

**J.3.3.47 OTF2\_ErrorCode OTF2\_AttributeList\_PopAttribute ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef \* *attribute*, OTF2\_Type \* *type*, OTF2\_AttributeValue \* *attributeValue* )**

Get first attribute from an attribute list and remove it.

Returns the first entry in the attribute list and removes it from the list.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.48 OTF2\_ErrorCode OTF2\_AttributeList\_RemoveAllAttributes ( OTF2\_AttributeList \* *attributeList* )**

Remove all attributes from an attribute list.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.49 OTF2\_ErrorCode OTF2\_AttributeList\_RemoveAttribute ( OTF2\_AttributeList \* *attributeList*, OTF2\_AttributeRef *attribute* )**

Remove an attribute from an attribute list.

### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.3.3.50    bool OTF2\_AttributeList\_TestAttributeByID ( const OTF2\_AttributeList \*  
attributeList, OTF2\_AttributeRef attribute )**

Test if an attribute is in the attribute list.

### **Parameters**

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

### **Returns**

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

## **J.4    OTF2\_Callbacks.h File Reference**

This header file provides all user callbacks.

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

### **Data Structures**

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

### **Typedefs**

- typedef int(\* [OTF2\\_FileSionClose](#) )(void \*userData, [OTF2\\_FileType](#) fileType, [OTF2\\_LocationRef](#) location, int sid)

## J.4 OTF2\_Callbacks.h File Reference

---

*Callbacks to wrap sion\_parclose\_mpi() for the OTF2 SION substrate.*

- `typedef OTF2_ErrorCode(* OTF2_FileSionGetRank )(void *userData, OTF2_-FileType fileType, OTF2_LocationRef location, int32_t *rank)`

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

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

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

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

*Function pointer for allocating memory for chunks.*

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

*Function pointer to release all allocated chunks.*

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

*Definition for the post flush callback.*

- `typedef OTF2_FlushType(* OTF2_PreFlushCallback )(void *userData, OTF2_-FileType fileType, OTF2_LocationRef location, void *callerData, bool final)`

*Definition for the pre flush callback.*

### J.4.1 Detailed Description

This header file provides all user callbacks.

#### Maintainer:

Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

#### Authors

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

## **APPENDIX J. FILE DOCUMENTATION**

---

### **J.4.2 TYPEDOC Documentation**

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

Callbacks to wrap sion\_parclose\_mpi() for the OTF2 SION substrate.

#### **Parameters**

<i>userData</i>	Data passed to the call <a href="#">OTF2_Archive_SetFileSionCallbacks</a> .
<i>fileType</i>	The file type for which the file close is called.
<i>location</i>	The location ID of the writer for which the flush has happened (for file types without an ID this is <a href="#">OTF2_UNDEFINED_LOCATION</a> ).
<i>sid</i>	Sion file handle.

#### **Returns**

Return value of sion\_parclose\_mpi()

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

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

In case no OTF2\_FileSionOpen and no OTF2\_FileSionClose callback is given, the SION substrate still needs information what rank the current location has.

#### **Parameters**

	<i>userData</i>	Data passed to the call <a href="#">OTF2_Archive_SetFileSionCallbacks</a> .
	<i>fileType</i>	The file type for which the file close is called.
	<i>location</i>	The location ID of the writer for which the flush has happened (for file types without an ID this is <a href="#">OTF2_UNDEFINED_LOCATION</a> ).
<i>out</i>	<i>rank</i>	The associated MPI rank for the <i>location</i> .

#### **Returns**

[OTF2\\_SUCCESS](#), or error code.

## J.4 OTF2\_Callbacks.h File Reference

---

J.4.2.3 **typedef int( \* OTF2\_FileSionOpen)(void \*userData, OTF2\_FileType fileType, OTF2\_LocationRef location, const char \*fname, const char \*fileMode, long long int \*chunkSize, int \*fsblkSize, FILE \*\*filePtr)**

Callbacks to wrap sion\_paropen\_mpi() for the OTF2 SION substrate. Every parameter that can be given by OTF2 is named equally like the the according parameter of sion\_paropen\_mpi(). Therfore, these given parameters MUST be given to SION.

### Parameters

	<i>userData</i>	Data passed to the call <a href="#">OTF2_Archive_SetFileSionCallbacks</a> .
	<i>fileType</i>	The file type for which the file open is called.
	<i>location</i>	The location ID of the writer for which the flush has happened (for file types without an ID this is <a href="#">OTF2_UNDEFINED_LOCATION</a> ).
	<i>fname</i>	Name of file, should equal on all tasks.
	<i>fileMode</i>	Like the type parameter of fopen.
in, out	<i>chunkSize</i>	Requested space for this task.
in, out	<i>fsblkSize</i>	Blocksize of filesystem, must be equal on all processors.
out	<i>filePtr</i>	Filepointer for this task.

### Returns

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

J.4.2.4 **typedef void\*( \* OTF2\_MemoryAllocate)(void \*userData, OTF2\_FileType fileType, OTF2\_LocationRef location, void \*\*perBufferData, uint64\_t chunkSize)**

Function pointer for allocating memory for chunks.

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

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

### Parameters

<i>userData</i>	Data passed to the call <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> ).

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>perBuffer-Data</i>	A writeable pointer to store callee data. For the first call this will be NULL.
<i>chunkSize</i>	The size of the requested chunk.

### **Returns**

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

#### **J.4.2.5 `typedef void( * OTF2_MemoryFreeAll)(void *userData, OTF2_FileType fileType, OTF2_LocationRef location, void **perBufferData, bool final)`**

Function pointer to release all allocated chunks.

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

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

### **Parameters**

<i>userData</i>	Data passed to the call <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>perBuffer-Data</i>	A writeable pointer to store callee data. For the first call this will be NULL.
<i>final</i>	Indicates whether this is the final free when closing the writer objects. perBufferData should be handled than.

#### **J.4.2.6 `typedef OTF2_TimeStamp( * OTF2_PostFlushCallback)(void *userData, OTF2_FileType fileType, OTF2_LocationRef location)`**

Definition for the post flush callback.

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

## J.5 OTF2\_Definitions.h File Reference

---

### Parameters

<i>userData</i>	Data passed to the call <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.

**J.4.2.7 `typedef OTF2_FlushType( * OTF2_PreFlushCallback)(void *userData,  
OTF2_FileType fileType, OTF2_LocationRef location, void *callerData, bool  
final)`**

Definition for the pre flush callback.

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

### Parameters

<i>userData</i>	Data passed to the call <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 <i>fileType</i> , 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](#).

## J.5 OTF2\_Definitions.h File Reference

Data types used in the definition records.

---

## APPENDIX J. FILE DOCUMENTATION

---

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

### TypeDefs

- **typedef uint32\_t OTF2\_GroupFlag**  
*Wrapper for enum OTF2\_GroupFlag\_enum.*
- **typedef uint8\_t OTF2\_GroupType**  
*Wrapper for enum OTF2\_GroupType\_enum.*
- **typedef uint8\_t OTF2\_LocationGroupType**  
*Wrapper for enum OTF2\_LocationGroupType\_enum.*
- **typedef uint8\_t OTF2\_LocationType**  
*Wrapper for enum OTF2\_LocationType\_enum.*
- **typedef uint8\_t OTF2\_MetricBase**  
*Wrapper for enum OTF2\_MetricBase\_enum.*
- **typedef uint8\_t OTF2\_MetricMode**  
*Wrapper for enum OTF2\_MetricMode\_enum.*
- **typedef uint8\_t OTF2\_MetricOccurrence**  
*Wrapper for enum OTF2\_MetricOccurrence\_enum.*
- **typedef uint8\_t OTF2\_MetricScope**  
*Wrapper for enum OTF2\_MetricScope\_enum.*
- **typedef uint8\_t OTF2\_MetricTiming**  
*Wrapper for enum OTF2\_MetricTiming\_enum.*
- **typedef uint8\_t OTF2\_MetricType**  
*Wrapper for enum OTF2\_MetricType\_enum.*
- **typedef uint8\_t OTF2\_MetricValueProperty**  
*Wrapper for enum OTF2\_MetricValueProperty\_enum.*
- **typedef uint8\_t OTF2\_ParameterType**  
*Wrapper for enum OTF2\_ParameterType\_enum.*
- **typedef uint8\_t OTF2\_RecorderKind**  
*Wrapper for enum OTF2\_RecorderKind\_enum.*
- **typedef uint32\_t OTF2\_RegionFlag**  
*Wrapper for enum OTF2\_RegionFlag\_enum.*
- **typedef uint8\_t OTF2\_RegionRole**  
*Wrapper for enum OTF2\_RegionRole\_enum.*
- **typedef uint8\_t OTF2\_SystemTreeDomain**  
*Wrapper for enum OTF2\_SystemTreeDomain\_enum.*

## J.5 OTF2\_Definitions.h File Reference

---

### Enumerations

- enum `OTF2_GroupFlag_enum` {  
    `OTF2_GROUP_FLAG_NONE` = 0,  
    `OTF2_GROUP_FLAG_GLOBAL_MEMBERS` = ( 1 << 0 ) }  
*List of possible flags to specify special characteristics of a Group.*
- enum `OTF2_GroupType_enum` {  
    `OTF2_GROUP_TYPE_UNKNOWN` = 0,  
    `OTF2_GROUP_TYPE_LOCATIONS` = 1,  
    `OTF2_GROUP_TYPE_REGIONS` = 2,  
    `OTF2_GROUP_TYPE_METRIC` = 3,  
    `OTF2_GROUP_TYPE_COMM_LOCATIONS` = 4,  
    `OTF2_GROUP_TYPE_COMM_GROUP` = 5,  
    `OTF2_GROUP_TYPE_COMM_SELF` = 6 }
- enum `OTF2_LocationGroupType_enum` {  
    `OTF2_LOCATION_GROUP_TYPE_UNKNOWN` = 0,  
    `OTF2_LOCATION_GROUP_TYPE_PROCESS` = 1 }  
*List of possible definitions of type LocationGroup.*
- enum `OTF2_LocationType_enum` {  
    `OTF2_LOCATION_TYPE_UNKNOWN` = 0,  
    `OTF2_LOCATION_TYPE_CPU_THREAD` = 1,  
    `OTF2_LOCATION_TYPE_GPU` = 2,  
    `OTF2_LOCATION_TYPE_METRIC` = 3 }  
*List of possible definitions of type Location.*
- enum `OTF2_MetricBase_enum` {  
    `OTF2_BASE_BINARY` = 0,  
    `OTF2_BASE_DECIMAL` = 1 }  
*Metric base types.*
- enum `OTF2_MetricMode_enum` {  
    `OTF2_METRIC_ACCUMULATED_START` = `OTF2_METRIC_VALUE_ACCUMULATED` | `OTF2_METRIC_TIMING_START`,  
    `OTF2_METRIC_ACCUMULATED_POINT` = `OTF2_METRIC_VALUE_ACCUMULATED` | `OTF2_METRIC_TIMING_POINT`,  
    `OTF2_METRIC_ACCUMULATED_LAST` = `OTF2_METRIC_VALUE_ACCUMULATED` | `OTF2_METRIC_TIMING_LAST`,

---

## **APPENDIX J. FILE DOCUMENTATION**

```
OTF2_METRIC_ACCUMULATED_NEXT = OTF2_METRIC_VALUE_-  
ACCUMULATED | OTF2_METRIC_TIMING_NEXT,  
OTF2_METRIC_ABSOLUTE_POINT = OTF2_METRIC_VALUE_ABSOLUTE  
| OTF2_METRIC_TIMING_POINT,  
OTF2_METRIC_ABSOLUTE_LAST = OTF2_METRIC_VALUE_ABSOLUTE  
| OTF2_METRIC_TIMING_LAST,  
OTF2_METRIC_ABSOLUTE_NEXT = OTF2_METRIC_VALUE_ABSOLUTE  
| OTF2_METRIC_TIMING_NEXT,  
OTF2_METRIC_RELATIVE_POINT = OTF2_METRIC_VALUE_RELATIVE  
| OTF2_METRIC_TIMING_POINT,  
OTF2_METRIC_RELATIVE_LAST = OTF2_METRIC_VALUE_RELATIVE  
| OTF2_METRIC_TIMING_LAST,  
OTF2_METRIC_RELATIVE_NEXT = OTF2_METRIC_VALUE_RELATIVE  
| OTF2_METRIC_TIMING_NEXT }
```

*Metric mode is a combination of value property and timing information.*

- enum **OTF2\_MetricOccurrence\_enum** {  
    OTF2\_METRIC\_SYNCHRONOUS\_STRICT = 0,  
    OTF2\_METRIC\_SYNCHRONOUS = 1,  
    OTF2\_METRIC\_ASYNCHRONOUS = 2 }

*Metric occurrence.*

- enum **OTF2\_MetricScope\_enum** {  
    OTF2\_SCOPE\_LOCATION = 0,  
    OTF2\_SCOPE\_LOCATION\_GROUP = 1,  
    OTF2\_SCOPE\_SYSTEM\_TREE\_NODE = 2,  
    OTF2\_SCOPE\_GROUP = 3 }
- enum **OTF2\_MetricTiming\_enum** {  
    OTF2\_METRIC\_TIMING\_START = 0,  
    OTF2\_METRIC\_TIMING\_POINT = 1 << 4,  
    OTF2\_METRIC\_TIMING\_LAST = 2 << 4,  
    OTF2\_METRIC\_TIMING\_NEXT = 3 << 4,  
    OTF2\_METRIC\_TIMING\_MASK = 240 }

*Determines when the values have been collected or for which interval of time they are valid. Used for the upper half-byte of OTF2\_MetricMode.*

- enum **OTF2\_MetricType\_enum** {  
    OTF2\_METRIC\_TYPE\_OTHER = 0,  
    OTF2\_METRIC\_TYPE\_PAPI = 1,

## J.5 OTF2\_Definitions.h File Reference

---

- OTF2\_METRIC\_TYPE\_RUSAGE = 2,  
OTF2\_METRIC\_TYPE\_USER = 3 }
- enum OTF2\_MetricValueProperty\_enum {  
OTF2\_METRIC\_VALUE\_ACCUMULATED = 0,  
OTF2\_METRIC\_VALUE\_ABSOLUTE = 1,  
OTF2\_METRIC\_VALUE\_RELATIVE = 2,  
OTF2\_METRIC\_VALUE\_MASK = 15 }  
*Information about whether the metric value is accumulated, absolute, or relative. Used for the lower half-byte of OTF2\_MetricMode.*
- enum OTF2\_ParameterType\_enum {  
OTF2\_PARAMETER\_TYPE\_STRING = 0,  
OTF2\_PARAMETER\_TYPE\_INT64 = 1,  
OTF2\_PARAMETER\_TYPE\_UINT64 = 2 }  
*List of possible definitions of type Parameter.*
- enum OTF2\_RecorderKind\_enum {  
OTF2\_RECORDER\_KIND\_UNKNOWN = 0,  
OTF2\_RECORDER\_KIND\_ABSTRACT = 1,  
OTF2\_RECORDER\_KIND\_CPU = 2,  
OTF2\_RECORDER\_KIND\_GPU = 3 }  
*List of possible kinds a MetricClass can be recorded by.*
- enum OTF2\_RegionFlag\_enum {  
OTF2\_REGION\_FLAG\_NONE = 0,  
OTF2\_REGION\_FLAG\_DYNAMIC = ( 1 << 0 ),  
OTF2\_REGION\_FLAG\_PHASE = ( 1 << 1 ) }  
*List of possible flags to specify special characteristics of a Region.*
- enum OTF2\_RegionRole\_enum {  
OTF2\_REGION\_ROLE\_UNKNOWN = 0,  
OTF2\_REGION\_ROLE\_FUNCTION = 1,  
OTF2\_REGION\_ROLE\_WRAPPER = 2,  
OTF2\_REGION\_ROLE\_LOOP = 3,  
OTF2\_REGION\_ROLE\_CODE = 4,  
OTF2\_REGION\_ROLE\_PARALLEL = 5,  
OTF2\_REGION\_ROLE\_SECTIONS = 6,  
OTF2\_REGION\_ROLE\_SECTION = 7,  
OTF2\_REGION\_ROLE\_WORKSHARE = 8,

---

## **APPENDIX J. FILE DOCUMENTATION**

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

*List of possible roles of a Region.*

- enum **OTF2\_SystemTreeDomain\_enum** {  
    OTF2\_SYSTEM\_TREE\_DOMAIN\_MACHINE = 0,  
    OTF2\_SYSTEM\_TREE\_DOMAIN\_SHARED\_MEMORY = 1,  
    OTF2\_SYSTEM\_TREE\_DOMAIN\_NUMA = 2,  
    OTF2\_SYSTEM\_TREE\_DOMAIN\_SOCKET = 3,  
    OTF2\_SYSTEM\_TREE\_DOMAIN\_CACHE = 4,  
    OTF2\_SYSTEM\_TREE\_DOMAIN\_CORE = 5,  
    OTF2\_SYSTEM\_TREE\_DOMAIN\_PU = 6 }

## **J.5 OTF2\_Definitions.h File Reference**

---

### **J.5.1 Detailed Description**

Data types used in the definition records.

**Source Template:**

*templates/OTF2\_Definitions.tpl.h*

**Maintainer:**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

**Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### **J.5.2 Enumeration Type Documentation**

#### **J.5.2.1 enum OTF2\_GroupFlag\_enum**

List of possible flags to specify special characteristics of a Group.

**Since**

Version 1.2

**Enumerator:**

***OTF2\_GROUP\_FLAG\_NONE*** A group without special characterization.

***OTF2\_GROUP\_FLAG\_GLOBAL\_MEMBERS*** No translation needs to be done when a group of type ***OTF2\_GROUP\_TYPE\_COMM\_GROUP*** has this flag.

#### **J.5.2.2 enum OTF2\_GroupType\_enum**

**Since**

Version 1.2

**Enumerator:**

***OTF2\_GROUP\_TYPE\_UNKNOWN*** Group of unknown type.

***OTF2\_GROUP\_TYPE\_LOCATIONS*** Group of locations.

***OTF2\_GROUP\_TYPE\_REGIONS*** Group of regions.

***OTF2\_GROUP\_TYPE\_METRIC*** Group of metrics.

***OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS*** List of location IDs, which are MPI ranks. The size of this group should match the size of MPI\_COMM\_WORLD. Each entry in the list is a location ID, where the index of the entry is equal to the rank in MPI\_COMM\_WORLD. (Ie. rank i corresponds to location members[i])

Also, if this definition is present, the location group ids of locations with type OTF2\_LOCATION\_TYPE\_CPU\_THREAD should match The MPI rank.

This group needs to be defined, before any group of type *OTF2\_GROUP\_TYPE\_MPI\_GROUP*.

Note: This does not makes sense in local definitions.

***OTF2\_GROUP\_TYPE\_COMM\_GROUP*** MPI group.

***OTF2\_GROUP\_TYPE\_COMM\_SELF*** Special group type to efficiently handle MPI self-like communicators.

### **J.5.2.3 enum OTF2\_LocationGroupType\_enum**

List of possible definitions of type LocationGroup.

#### **Since**

Version 1.0

#### **Enumerator:**

***OTF2\_LOCATION\_GROUP\_TYPE\_UNKNOWN*** A location group of unknown type.

***OTF2\_LOCATION\_GROUP\_TYPE\_PROCESS*** A process.

### **J.5.2.4 enum OTF2\_LocationType\_enum**

List of possible definitions of type Location.

#### **Since**

Version 1.0

#### **Enumerator:**

***OTF2\_LOCATION\_TYPE\_UNKNOWN*** A location of unknown type.

## J.5 OTF2\_Definitions.h File Reference

---

***OTF2\_LOCATION\_TYPE\_CPU\_THREAD*** A CPU thread.

***OTF2\_LOCATION\_TYPE\_GPU*** A GPU location.

***OTF2\_LOCATION\_TYPE\_METRIC*** A metric only location e.g. an external device.

### J.5.2.5 enum OTF2\_MetricBase\_enum

Metric base types.

#### Since

Version 1.0

#### Enumerator:

***OTF2\_BASE\_BINARY*** Binary base.

***OTF2\_BASE\_DECIMAL*** Decimal base.

### J.5.2.6 enum OTF2\_MetricMode\_enum

Metric mode is a combination of value property and timing information.

#### Since

Version 1.0

#### Enumerator:

***OTF2\_METRIC\_ACCUMULATED\_START*** Accumulated metric, 'START' timing.

***OTF2\_METRIC\_ACCUMULATED\_POINT*** Accumulated metric, 'POINT' timing.

***OTF2\_METRIC\_ACCUMULATED\_LAST*** Accumulated metric, 'LAST' timing.

***OTF2\_METRIC\_ACCUMULATED\_NEXT*** Accumulated metric, 'NEXT' timing.

***OTF2\_METRIC\_ABSOLUTE\_POINT*** Absolute metric, 'POINT' timing.

***OTF2\_METRIC\_ABSOLUTE\_LAST*** Absolute metric, 'LAST' timing.

***OTF2\_METRIC\_ABSOLUTE\_NEXT*** Absolute metric, 'NEXT' timing.

***OTF2\_METRIC\_RELATIVE\_POINT*** Relative metric, 'POINT' timing.

***OTF2\_METRIC\_RELATIVE\_LAST*** Relative metric, 'LAST' timing.

***OTF2\_METRIC\_RELATIVE\_NEXT*** Relative metric, 'NEXT' timing.

### **J.5.2.7 enum OTF2\_MetricOccurrence\_enum**

Metric occurrence.

#### **Since**

Version 1.0

#### **Enumerator:**

***OTF2\_METRIC\_SYNCHRONOUS\_STRICT*** Metric occurs at every region enter and leave.

***OTF2\_METRIC\_SYNCHRONOUS*** Metric occurs only at a region enter and leave, but does not need to occur at every enter/leave.

***OTF2\_METRIC\_ASYNCNHRONOUS*** Metric can occur at any place i.e. it is not related to region enter and leaves.

### **J.5.2.8 enum OTF2\_MetricScope\_enum**

#### **Since**

Version 1.0

#### **Enumerator:**

***OTF2\_SCOPE\_LOCATION*** Scope of a metric is another location.

***OTF2\_SCOPE\_LOCATION\_GROUP*** Scope of a metric is a location group.

***OTF2\_SCOPE\_SYSTEM\_TREE\_NODE*** Scope of a metric is a system tree node.

***OTF2\_SCOPE\_GROUP*** Scope of a metric is a generic group of locations.

### **J.5.2.9 enum OTF2\_MetricTiming\_enum**

Determines when the values have been collected or for which interval of time they are valid. Used for the upper half-byte of OTF2\_MetricMode.

#### **Since**

Version 1.0

#### **Enumerator:**

***OTF2\_METRIC\_TIMING\_START*** Metric value belongs to the time interval since the beginning of the measurement.

## J.5 OTF2\_Definitions.h File Reference

---

***OTF2\_METRIC\_TIMING\_POINT*** Metric value is only valid at a point in time but not necessarily for any interval of time.

***OTF2\_METRIC\_TIMING\_LAST*** Metric value is related to the time interval since the last counter sample of the same metric, i.e. the immediate past.

***OTF2\_METRIC\_TIMING\_NEXT*** Metric value is valid from now until the next counter sample, i.e. the future right ahead.

***OTF2\_METRIC\_TIMING\_MASK*** This mask can be used to get the upper half-byte in OTF2\_MetricMode that is used to indicate metric timing information.

### J.5.2.10 enum OTF2\_MetricType\_enum

**Since**

Version 1.0

**Enumerator:**

***OTF2\_METRIC\_TYPE\_OTHER*** Any metric of a type not explicitly listed below.

***OTF2\_METRIC\_TYPE\_PAPI*** PAPI counter.

***OTF2\_METRIC\_TYPE\_RUSAGE*** Resource usage counter.

***OTF2\_METRIC\_TYPE\_USER*** User metrics.

### J.5.2.11 enum OTF2\_MetricValueProperty\_enum

Information about whether the metric value is accumulated, absolute, or relative.  
Used for the lower half-byte of OTF2\_MetricMode.

**Since**

Version 1.0

**Enumerator:**

***OTF2\_METRIC\_VALUE\_ACCUMULATED*** Accumulated metric is monotonously increasing (i.e., PAPI counter for number of executed floating point operations).

***OTF2\_METRIC\_VALUE\_ABSOLUTE*** Absolute metric (i.e., temperature, rate, mean value, etc.).

***OTF2\_METRIC\_VALUE\_RELATIVE*** Relative metric.

***OTF2\_METRIC\_VALUE\_MASK*** This mask can be used to get lower half-byte in OTF2\_MetricMode that is used to indicate metric value property.

### **J.5.2.12 enum OTF2\_ParameterType\_enum**

List of possible for definitions of type Parameter.

#### **Since**

Version 1.0

#### **Enumerator:**

***OTF2\_PARAMETER\_TYPE\_STRING*** Parameter is of type string.

***OTF2\_PARAMETER\_TYPE\_INT64*** Parameter is of type signed 8-byte integer.

***OTF2\_PARAMETER\_TYPE\_UINT64*** Parameter is of type unsigned 8-byte integer.

### **J.5.2.13 enum OTF2\_RecorderKind\_enum**

List of possible kinds a MetricClass can be recorded by.

#### **Since**

Version 1.2

#### **Enumerator:**

***OTF2\_RECORDER\_KIND\_UNKNOWN*** No specific kind of recorder.

***OTF2\_RECORDER\_KIND\_ABSTRACT*** Only *MetricInstances* will record this metric class.

***OTF2\_RECORDER\_KIND\_CPU*** This metric class will only be recorded by locations of type [\*OTF2\\_LOCATION\\_TYPE\\_CPU\\_THREAD\*](#).

***OTF2\_RECORDER\_KIND\_GPU*** This metric class will only be recorded by locations of type [\*OTF2\\_LOCATION\\_TYPE\\_GPU\*](#).

## **J.5 OTF2\_Definitions.h File Reference**

---

### **J.5.2.14 enum OTF2\_RegionFlag\_enum**

List of possible flags to specify special characteristics of a Region.

#### **Since**

Version 1.1

#### **Enumerator:**

***OTF2\_REGION\_FLAG\_NONE*** A region without special characterization.

***OTF2\_REGION\_FLAG\_DYNAMIC*** Each time this region is entered it will get an individual call path in the profile.

***OTF2\_REGION\_FLAG\_PHASE*** Each time this region is entered it will get an individual root node in the profile.

### **J.5.2.15 enum OTF2\_RegionRole\_enum**

List of possible roles of a Region.

#### **Since**

Version 1.1

#### **Enumerator:**

***OTF2\_REGION\_ROLE\_UNKNOWN*** A region of unknown role.

***OTF2\_REGION\_ROLE\_FUNCTION*** An entire function/subroutine.

***OTF2\_REGION\_ROLE\_WRAPPER*** An API function wrapped by Score-P.

***OTF2\_REGION\_ROLE\_LOOP*** A loop in the code.

***OTF2\_REGION\_ROLE\_CODE*** An arbitrary section of code.

***OTF2\_REGION\_ROLE\_PARALLEL*** E.g. OpenMP "parallel" construct (structured block)

***OTF2\_REGION\_ROLE\_SECTIONS*** E.g. OpenMP "sections" construct.

***OTF2\_REGION\_ROLE\_SECTION*** Individual "section" inside an OpenMP "sections" construct.

***OTF2\_REGION\_ROLE\_WORKSHARE*** E.g. OpenMP "workshare" construct.

***OTF2\_REGION\_ROLE\_SINGLE*** E.g. OpenMP "single" construct.

---

## **APPENDIX J. FILE DOCUMENTATION**

- OTF2\_REGION\_ROLE\_SINGLE\_SBLOCK*** E.g. OpenMP "single" construct (structured block)
- OTF2\_REGION\_ROLE\_MASTER*** E.g. OpenMP "master" construct.
- OTF2\_REGION\_ROLE\_CRITICAL*** E.g. OpenMP "critical" construct.
- OTF2\_REGION\_ROLE\_CRITICAL\_SBLOCK*** E.g. OpenMP "critical" construct (structured block)
- OTF2\_REGION\_ROLE\_ATOMIC*** E.g. OpenMP "atomic" construct.
- OTF2\_REGION\_ROLE\_BARRIER*** Explicit barrier.
- OTF2\_REGION\_ROLE\_IMPLICIT\_BARRIER*** Implicit barrier.
- OTF2\_REGION\_ROLE\_FLUSH*** E.g. OpenMP "flush" construct.
- OTF2\_REGION\_ROLE\_ORDERED*** E.g. OpenMP "ordered" construct.
- OTF2\_REGION\_ROLE\_ORDERED\_SBLOCK*** E.g. OpenMP "ordered" construct (structured block)
- OTF2\_REGION\_ROLE\_TASK*** "task" construct (structured block)
- OTF2\_REGION\_ROLE\_TASK\_CREATE*** "task" construct (creation)
- OTF2\_REGION\_ROLE\_TASK\_WAIT*** "taskwait" construct
- OTF2\_REGION\_ROLE\_COLL\_ONE2ALL*** Collective 1:N communication operation.
- OTF2\_REGION\_ROLE\_COLL\_ALL2ONE*** Collective N:1 communication operation.
- OTF2\_REGION\_ROLE\_COLL\_ALL2ALL*** Collective N:N communication operation.
- OTF2\_REGION\_ROLE\_COLL\_OTHER*** Collective M:N communication operation.
- OTF2\_REGION\_ROLE\_FILE\_IO*** Any file I/O operation.
- OTF2\_REGION\_ROLE\_POINT2POINT*** A point-to-point communication function.
- OTF2\_REGION\_ROLE\_RMA*** A remote memory access communication operation.
- OTF2\_REGION\_ROLE\_DATA\_TRANSFER*** A data transfer operation in memory.
- OTF2\_REGION\_ROLE\_ARTIFICIAL*** An artificial region, mostly used by the monitor software.
- Since**

Version 1.2.

## J.6 OTF2\_DefReader.h File Reference

---

### J.5.2.16 enum OTF2\_SystemTreeDomain\_enum

Since

Version 1.2

**Enumerator:**

***OTF2\_SYSTEM\_TREE\_DOMAIN\_MACHINE*** All nodes below a node with this attribute encompass a tightly coupled HPC system.

***OTF2\_SYSTEM\_TREE\_DOMAIN\_SHARED\_MEMORY*** All nodes below a node with this attribute encompass a system where processes can communicate via hardware shared memory.

***OTF2\_SYSTEM\_TREE\_DOMAIN\_NUMA*** A numa domain. A set of processors around memory which the processors can directly access.

***OTF2\_SYSTEM\_TREE\_DOMAIN\_SOCKET*** Socket, physical package, or chip. In the physical meaning, i.e. that you can add or remove physically.

***OTF2\_SYSTEM\_TREE\_DOMAIN\_CACHE*** Cache. Can be L1i, L1d, L2, L3, ...

***OTF2\_SYSTEM\_TREE\_DOMAIN\_CORE*** Core. A computation unit (may be shared by several logical processors).

***OTF2\_SYSTEM\_TREE\_DOMAIN\_PU*** Processing Unit (An non-shared ALU, FPU, ...)

## J.6 OTF2\_DefReader.h File Reference

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

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

### Functions

- **OTF2\_ErrorCode OTF2\_DefReader\_GetLocationID** (const **OTF2\_DefReader** \*reader, **OTF2\_LocationRef** \*location)

*Get the location ID of this reader object.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_DefReader\_ReadDefinitions (OTF2\_DefReader \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)**  
*Reads the given number of records from the definition reader.*
- **OTF2\_ErrorCode OTF2\_DefReader\_SetCallbacks (OTF2\_DefReader \*reader, const OTF2\_DefReaderCallbacks \*callbacks, void \*userData)**  
*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### **J.6.1 Detailed Description**

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

#### **Maintainer:**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

#### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### **J.6.2 Function Documentation**

#### **J.6.2.1 OTF2\_ErrorCode OTF2\_DefReader\_GetLocationID ( const OTF2\_DefReader \* reader, OTF2\_LocationRef \* location )**

Get the location ID of this reader object.

#### **Parameters**

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

#### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.6 OTF2\_DefReader.h File Reference

---

### J.6.2.2 OTF2\_ErrorCode OTF2\_DefReader\_ReadDefinitions ( OTF2\_DefReader \* reader, uint64\_t recordsToRead, uint64\_t \* recordsRead )

Reads the given number of records from the definition reader.

#### Parameters

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.
<i>out</i>	<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 finnished his job by checking recordsRead < recordsToRead.

#### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK** if an user supplied callback returned OTF2\_CALLBACK\_INTERRUPT

**OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE** if an duplicate mapping table definition was read

*otherwise* the error code

### J.6.2.3 OTF2\_ErrorCode OTF2\_DefReader\_SetCallbacks ( OTF2\_DefReader \* reader, const OTF2\_DefReaderCallbacks \* callbacks, void \* userData )

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

#### Parameters

<i>reader</i>	This given reader object will be setted up with new callback functions.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#">OTF2_DefReaderCallbacks_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.

## J.7 OTF2\_DefReaderCallbacks.h File Reference

This defines the callbacks for the definition reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_IdMap.h>
```

### Typedefs

- `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_Attribute )(void *userData, OTF2_AttributeRef self, OTF2StringRef name, OTF2_Type type)`

*Function pointer definition for the callback which is triggered by a `Attribute` definition record.*

- `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_Callpath )(void *userData, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region)`

*Function pointer definition for the callback which is triggered by a `Callpath` definition record.*

- `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_Callsite )(void *userData, OTF2_CallsiteRef self, OTF2StringRef sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion, OTF2_RegionRef leftRegion)`

*Function pointer definition for the callback which is triggered by a `Callsite` definition record.*

- `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_ClockOffset )(void *userData, OTF2_TimeStamp time, int64_t offset, double standardDeviation)`

*Function pointer definition for the callback which is triggered by a `ClockOffset` definition record.*

- `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_Comm )(void *userData, OTF2_CommRef self, OTF2StringRef name, OTF2_GroupRef group, OTF2_CommRef parent)`

*Function pointer definition for the callback which is triggered by a `Comm` definition record.*

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_Group )(void *userData, OTF2_GroupRef self, OTF2StringRef 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, OTF2StringRef 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, OTF2StringRef name, OTF2_LocationGroupType locationGroupType, OTF2_SystemTreeNodeRef systemTreeParent)`

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

- `typedef OTF2_CallbackCode(* OTF2_DefReaderCallback_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, OTF2RecorderKind 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 [MetricClassRecorder](#) 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, OTF2StringRef name, OTF2StringRef description, OTF2_MetricType metricType, OTF2_MetricMode`

## **APPENDIX J. FILE DOCUMENTATION**

---

metricMode, [OTF2\\_Type](#) valueType, [OTF2\\_MetricBase](#) metricBase, int64\_t exponent, [OTF2StringRef](#) unit)

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

- [typedef OTF2\\_CallbackCode\(\\* OTF2\\_DefReaderCallback\\_Parameter\)\(void \\*userData, OTF2\\_ParameterRef self, OTF2StringRef name, OTF2\\_ParameterType parameterType\)](#)

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

- [typedef OTF2\\_CallbackCode\(\\* OTF2\\_DefReaderCallback\\_Region \)\(void \\*userData, OTF2\\_RegionRef self, OTF2StringRef name, OTF2StringRef canonicalName, OTF2StringRef description, OTF2\\_RegionRole regionRole, OTF2\\_Paradigm paradigm, OTF2\\_RegionFlag regionFlags, OTF2StringRef sourceFile, uint32\\_t beginLineNumber, uint32\\_t endLineNumber\)](#)

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

- [typedef OTF2\\_CallbackCode\(\\* OTF2\\_DefReaderCallback\\_RmaWin \)\(void \\*userData, OTF2\\_RmaWinRef self, OTF2StringRef name, OTF2\\_CommRef comm\)](#)

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

- [typedef OTF2\\_CallbackCode\(\\* OTF2\\_DefReaderCallback\\_String \)\(void \\*userData, OTF2StringRef self, const char \\*string\)](#)

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

- [typedef OTF2\\_CallbackCode\(\\* OTF2\\_DefReaderCallback\\_SystemTreeNode \)\(void \\*userData, OTF2\\_SystemTreeNodeRef self, OTF2StringRef name, OTF2StringRef className, OTF2\\_SystemTreeNodeRef parent\)](#)

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

- [typedef OTF2\\_CallbackCode\(\\* OTF2\\_DefReaderCallback\\_SystemTreeNodeDomain \)\(void \\*userData, OTF2\\_SystemTreeNodeRef systemTreeNode, OTF2\\_SystemDomain systemTreeDomain\)](#)

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

- [typedef OTF2\\_CallbackCode\(\\* OTF2\\_DefReaderCallback\\_SystemTreeNodeProperty \)\(void \\*userData, OTF2\\_SystemTreeNodeRef systemTreeNode, OTF2StringRef name, OTF2StringRef value\)](#)

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

- [typedef OTF2\\_CallbackCode\(\\* OTF2\\_DefReaderCallback\\_Unknown \)\(void \\*userData\)](#)

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

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

- `typedef struct OTF2_DefReaderCallbacks_struct OTF2_DefReaderCallbacks`

*Opaque struct which holds all definition record callbacks.*

### Functions

- `void OTF2_DefReaderCallbacks_Clear (OTF2_DefReaderCallbacks *defReaderCallbacks)`

*Clears a struct for the definition callbacks.*

- `void OTF2_DefReaderCallbacks_Delete (OTF2_DefReaderCallbacks *defReaderCallbacks)`

*Deallocates a struct for the definition callbacks.*

- `OTF2_DefReaderCallbacks * OTF2_DefReaderCallbacks_New (void)`

*Allocates a new struct for the definition callbacks.*

- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetAttributeCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_Attribute attributeCallback)`

*Registers the callback for the `Attribute` definition.*

- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallpathCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_Callpath callpathCallback)`

*Registers the callback for the `Callpath` definition.*

- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCallsiteCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_Callsite callsiteCallback)`

*Registers the callback for the `Callsite` definition.*

- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetClockOffsetCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_ClockOffset clockOffsetCallback)`

*Registers the callback for the `ClockOffset` definition.*

- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetCommCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_Comm commCallback)`

*Registers the callback for the `Comm` definition.*

- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetGroupCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_Group groupCallback)`

*Registers the callback for the `Group` definition.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetLocationCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_Location locationCallback)**

*Registers the callback for the [Location](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetLocationGroupCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_LocationGroup locationGroupCallback)**

*Registers the callback for the [LocationGroup](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetMappingTableCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_MappingTable mappingTableCallback)**

*Registers the callback for the [MappingTable](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetMetricClassCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_MetricClass metricClassCallback)**

*Registers the callback for the [MetricClass](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetMetricClassRecorderCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_MetricClassRecorder metricClassRecorderCallback)**

*Registers the callback for the [MetricClassRecorder](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetMetricInstanceCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_MetricInstance metricInstanceCallback)**

*Registers the callback for the [MetricInstance](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetMetricMemberCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_MetricMember metricMemberCallback)**

*Registers the callback for the [MetricMember](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetParameterCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_Parameter parameterCallback)**

*Registers the callback for the [Parameter](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetRegionCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_Region regionCallback)**

*Registers the callback for the [Region](#) definition.*
- **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetRmaWinCallback (OTF2\_DefReaderCallbacks \*defReaderCallbacks, OTF2\_DefReaderCallback\_RmaWin rmaWinCallback)**

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

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetStringCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_String stringCallback)`

*Registers the callback for the `String` definition.*
- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodeCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_SystemTreeNode systemTreeNodeCallback)`

*Registers the callback for the `SystemTreeNode` definition.*
- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodeDomainCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_SystemTreeNodeDomain systemTreeNodeDomainCallback)`

*Registers the callback for the `SystemTreeNodeDomain` definition.*
- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodePropertyCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_SystemTreeNodeProperty systemTreeNodePropertyCallback)`

*Registers the callback for the `SystemTreeNodeProperty` definition.*
- `OTF2_ErrorCode OTF2_DefReaderCallbacks_SetUnknownCallback (OTF2_DefReaderCallbacks *defReaderCallbacks, OTF2_DefReaderCallback_Unknown unknownCallback)`

*Registers the callback for an unknown definition.*

### J.7.1 Detailed Description

This defines the callbacks for the definition reader.

#### Source Template:

`templates/OTF2_DefReaderCallbacks.tmpl.h`

#### Maintainer:

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

#### Authors

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### J.7.2 Typedef Documentation

**J.7.2.1 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-Attribute)(void *userData, OTF2_AttributeRef self, OTF2StringRef name, OTF2_Type type)`**

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

#### Parameters

<i>userData</i>	User data as set by <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>type</i>	Type of the attribute value.

#### Since

Version 1.0

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.7.2.2 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-Callpath)(void *userData, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region)`**

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

#### Parameters

<i>userData</i>	User data as set by <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>	References a <a href="#">Callpath</a> definition.
<i>region</i>	References a <a href="#">Region</a> definition.

#### Since

Version 1.0

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.7.2.3 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-  
Callsite)(void *userData, OTF2_CallsiteRef self, OTF2StringRef  
sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion,  
OTF2_RegionRef leftRegion)`**

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

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_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*.

**J.7.2.4 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-  
ClockOffset)(void *userData, OTF2_TimeStamp time, int64_t offset, double  
standardDeviation)`**

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

Clock offsets are used for clock corrections.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
-----------------	---

## APPENDIX J. FILE DOCUMENTATION

---

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

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.7.2.5 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_ - Comm)(void *userData, OTF2_CommRef self, OTF2StringRef name, OTF2_GroupRef group, OTF2_CommRef parent)`**

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

### Parameters

<i>userData</i>	User data as set by <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 <code>OTF2_GROUP_TYPE_MPI_GROUP</code> or <code>OTF2_GROUP_TYPE_MPI_COMM_SELF</code> . References a <a href="#">Group</a> definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <code>OTF2_UNDEFINED_COMM</code> to indicate no parent. References a <a href="#">Comm</a> definition.

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

J.7.2.6 **typedef OTF2\_CallbackCode( \* OTF2\_DefReaderCallback\_Group)(void \*userData, OTF2\_GroupRef self, OTF2StringRef name, OTF2\_GroupType groupType, OTF2\_Paradigm paradigm, OTF2\_GroupFlag groupFlags, uint32\_t numberOfMembers, const uint64\_t \*members)**

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

### Parameters

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

J.7.2.7 **typedef OTF2\_CallbackCode( \* OTF2\_DefReaderCallback\_Location)(void \*userData, OTF2\_LocationRef self, OTF2StringRef name, OTF2\_LocationType locationType, uint64\_t numberOfEvents, OTF2\_LocationGroupRef locationGroup)**

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

### Parameters

<i>userData</i>	User data as set by <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.

## APPENDIX J. FILE DOCUMENTATION

---

<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](#).

```
J.7.2.8 typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-  
LocationGroup)(void *userData, OTF2_LocationGroupRef self,  
OTF2StringRef name, OTF2_LocationGroupType locationGroupType,  
OTF2_SystemTreeNodeRef systemTreeParent)
```

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

### Parameters

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

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

J.7.2.9 **typedef OTF2\_CallbackCode( \* OTF2\_DefReaderCallback\_-  
MappingTable)(void \*userData, OTF2\_MappingType mappingType, const  
OTF2\_IdMap \*idMap)**

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

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

### Parameters

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

J.7.2.10 **typedef OTF2\_CallbackCode( \* OTF2\_DefReaderCallback\_-  
MetricClass)(void \*userData, OTF2\_MetricRef self, uint8\_t  
numberOfMetrics, const OTF2\_MetricMemberRef \*metricMembers,  
OTF2\_MetricOccurrence metricOccurrence, OTF2RecorderKind  
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>numberOf- Metrics</i>	Number of metrics within the set.

## **APPENDIX J. FILE DOCUMENTATION**

---

<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*.

**J.7.2.11 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-MetricClassRecorder)(void *userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)`**

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

### **Parameters**

<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*.

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

```
J.7.2.12 typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
MetricInstance)(void *userData, OTF2_MetricRef self,  
OTF2_MetricRef metricClass, OTF2_LocationRef recorder,  
OTF2_MetricScope metricScope, uint64_t scope)
```

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

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

### 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="#"><i>OTF2_RECORDER_KIND_ABSTRACT</i></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\*](#).

```
J.7.2.13 typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_  
MetricMember)(void *userData, OTF2_MetricMemberRef  
self, OTF2StringRef name, OTF2StringRef description,  
OTF2_MetricType metricType, OTF2_MetricMode metricMode,  
OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent,  
OTF2_StringRef unit)
```

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

A metric is defined by a metric member definition. A metric member is always a member of a metric class. Therefore, a single metric is a special case of a metric

## APPENDIX J. FILE DOCUMENTATION

---

class with only one member. It is not allowed to reference a metric member id in a metric event, but only metric class IDs.

### Parameters

<i>userData</i>	User data as set by <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: int64_t, uint64_t, or double.
<i>metricBase</i>	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor base <sup>exponent</sup> , 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](#).

**J.7.2.14 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_Parameter)(void *userData, OTF2_ParameterRef self, OTF2StringRef name, OTF2_ParameterType parameterType)`**

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

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
-----------------	---

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

<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](#).

**J.7.2.15** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_Region)(void *userData, OTF2_RegionRef self, OTF2StringRef name, OTF2StringRef canonicalName, OTF2StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)`

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

### Parameters

<i>userData</i>	User data as set by <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>beginLi- neNumber</i>	Starting line number of this region in the source file.
<i>endLi- neNumber</i>	Ending line number of this region in the source file.

**Since**

Version 1.0

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.7.2.16** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-  
RmaWin)(void *userData, OTF2_RmaWinRef self, OTF2_StringRef  
name, OTF2_CommRef comm)`

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

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

**Parameters**

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

**J.7.2.17** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-  
String)(void *userData, OTF2_StringRef self, const char  
*string)`

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

**Parameters**

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

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

**J.7.2.18** `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-SystemTreeNode)(void *userData, OTF2_SystemTreeNodeRef self, OTF2StringRef name, OTF2StringRef className, OTF2_SystemTreeNodeRef parent)`

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

### Parameters

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

**J.7.2.19 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-SystemTreeNodeDomain)(void *userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_SystemTreeDomain systemTreeDomain)`**

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

#### Parameters

<i>userData</i>	User data as set by <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.

#### Since

Version 1.2

#### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.7.2.20 `typedef OTF2_CallbackCode( * OTF2_DefReaderCallback_-SystemTreeNodeProperty)(void *userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2StringRef name, OTF2StringRef value)`**

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

#### 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>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

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.7.2.21 `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 <a href="#">OTF2_Reader_RegisterDefCallbacks</a> or <a href="#">OTF2_DefReader_SetCallbacks</a> .
-----------------	---

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## J.7.3 Function Documentation

**J.7.3.1 `void OTF2_DefReaderCallbacks_Clear( OTF2_DefReaderCallbacks * defReaderCallbacks )`**

Clears a struct for the definition callbacks.

### Parameters

<i>defReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_DefReaderCallbacks_New</a> .
---------------------------	--

**J.7.3.2 `void OTF2_DefReaderCallbacks_Delete( OTF2_DefReaderCallbacks * defReaderCallbacks )`**

Deallocates a struct for the definition callbacks.

### Parameters

<i>defReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_DefReaderCallbacks_New</a> .
---------------------------	--

## APPENDIX J. FILE DOCUMENTATION

### J.7.3.3 OTF2\_DefReaderCallbacks\* OTF2\_DefReaderCallbacks\_New ( void )

Allocates a new struct for the definition callbacks.

#### Returns

A newly allocated struct of type [OTF2\\_DefReaderCallbacks](#).

### J.7.3.4 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetAttributeCallback ( OTF2\_DefReaderCallbacks \* defReaderCallbacks, OTF2\_DefReaderCallback\_Attribute attributeCallback )

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

#### Parameters

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

#### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid *defReaderCallbacks* argument

### J.7.3.5 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetCallpathCallback ( OTF2\_DefReaderCallbacks \* defReaderCallbacks, OTF2\_DefReaderCallback\_Callpath callpathCallback )

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

#### Parameters

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

#### Returns

[OTF2\\_SUCCESS](#) if successful

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

### J.7.3.6 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetCallsiteCallback ( `OTF2_DefReaderCallbacks * defReaderCallbacks,` `OTF2_DefReaderCallback_Callsite callsiteCallback` )

Registers the callback for the `Callsite` definition.

#### Parameters

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

#### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

### J.7.3.7 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetClockOffsetCallback ( `OTF2_DefReaderCallbacks * defReaderCallbacks,` `OTF2_DefReaderCallback_ClockOffset clockOffsetCallback` )

Registers the callback for the `ClockOffset` definition.

#### Parameters

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

#### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

**J.7.3.8 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetCommCallback**  
( **OTF2\_DefReaderCallbacks \* defReaderCallbacks,**  
**OTF2\_DefReaderCallback\_Comm commCallback** )

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

### Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>commCallback</i>	Function which should be called for all <a href="#">Comm</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

**J.7.3.9 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetGroupCallback**  
( **OTF2\_DefReaderCallbacks \* defReaderCallbacks,**  
**OTF2\_DefReaderCallback\_Group groupCallback** )

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

### Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>groupCallback</i>	Function which should be called for all <a href="#">Group</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

**J.7.3.10 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetLocationCallback**  
( **OTF2\_DefReaderCallbacks \* defReaderCallbacks,**  
**OTF2\_DefReaderCallback\_Location locationCallback** )

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

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

### Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>locationCallback</i>	Function which should be called for all <a href="#">Location</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

### J.7.3.11 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetLocationGroupCallback

( `OTF2_DefReaderCallbacks * defReaderCallbacks,`

`OTF2_DefReaderCallback_LocationGroup locationGroupCallback` )

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

### Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>locationGroupCallback</i>	Function which should be called for all <a href="#">LocationGroup</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

### J.7.3.12 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetMappingTableCallback

( `OTF2_DefReaderCallbacks * defReaderCallbacks,`

`OTF2_DefReaderCallback_MappingTable mappingTableCallback` )

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

### Parameters

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

## APPENDIX J. FILE DOCUMENTATION

<i>mappingTable-Callback</i>	Function which should be called for all <a href="#">MappingTable</a> definitions.
------------------------------	---

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.7.3.13 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetMetricClassCallback**  
( `OTF2_DefReaderCallbacks * defReaderCallbacks,`  
`OTF2_DefReaderCallback_MetricClass metricClassCallback` )

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

### Parameters

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

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.7.3.14 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetMetricClassRecorderCallback**  
( `OTF2_DefReaderCallbacks * defReaderCallbacks,`  
`OTF2_DefReaderCallback_MetricClassRecorder metricClassRecorderCallback` )

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

### Parameters

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

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

<i>metric- Class- Recorder- Callback</i>	Function which should be called for all <a href="#">MetricClassRecorder</a> definitions.
--	--

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

### J.7.3.15 [\*\*OTF2\\_ErrorCode OTF2\\_DefReaderCallbacks\\_SetMetricInstanceCallback\*\*](#)

( `OTF2_DefReaderCallbacks * defReaderCallbacks,`  
`OTF2_DefReaderCallback_MetricInstance metricInstanceCallback` )

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

### Parameters

<i>defReader- Callbacks</i>	Struct for all callbacks.
<i>metricIn- stanceCall- back</i>	Function which should be called for all <a href="#">MetricInstance</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

### J.7.3.16 [\*\*OTF2\\_ErrorCode OTF2\\_DefReaderCallbacks\\_SetMetricMemberCallback\*\*](#)

( `OTF2_DefReaderCallbacks * defReaderCallbacks,`  
`OTF2_DefReaderCallback_MetricMember metricMemberCallback` )

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

### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>metricMemberCallback</i>	Function which should be called for all <a href="#">MetricMember</a> definitions.
-----------------------------	---

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

### **J.7.3.17 OTF2\_ErrorCode 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 <a href="#">Parameter</a> definitions.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

### **J.7.3.18 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetRegionCallback**

**( `OTF2_DefReaderCallbacks * defReaderCallbacks,`  
`OTF2_DefReaderCallback_Region regionCallback` )**

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

### **Parameters**

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>regionCallback</i>	Function which should be called for all <a href="#">Region</a> definitions.

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

#### J.7.3.19 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetRmaWinCallback

( `OTF2_DefReaderCallbacks * defReaderCallbacks,`  
`OTF2_DefReaderCallback_RmaWin rmaWinCallback` )

Registers the callback for the `RmaWin` definition.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

#### J.7.3.20 OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetStringCallback

( `OTF2_DefReaderCallbacks * defReaderCallbacks,`  
`OTF2_DefReaderCallback_String stringCallback` )

Registers the callback for the `String` definition.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.7.3.21 **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_SetSystemTreeNodeCallback**  
( **OTF2\_DefReaderCallbacks \* defReaderCallbacks,**  
**OTF2\_DefReaderCallback\_SystemTreeNode systemTreeNodeCallback** )

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

### Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>systemTreeNodeCallback</i>	Function which should be called for all <a href="#">SystemTreeNode</a> definitions.

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

J.7.3.22 **OTF2\_ErrorCode OTF2\_DefReaderCallbacks\_-**  
**SetSystemTreeNodeDomainCallback** ( **OTF2\_DefReaderCallbacks**  
\* *defReaderCallbacks*, **OTF2\_DefReaderCallback\_-**  
**SystemTreeNodeDomain** *systemTreeNodeDomainCallback*  
)

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

### Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>systemTreeNodeDomainCallback</i>	Function which should be called for all <a href="#">SystemTreeNodeDomain</a> definitions.

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

## J.7 OTF2\_DefReaderCallbacks.h File Reference

---

```
J.7.3.23 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetSystemTreeNodePropertyCallback ( OTF2_DefReaderCallbacks
* defReaderCallbacks, OTF2_DefReaderCallback_
SystemTreeNodeProperty systemTreeNodePropertyCallback
)
```

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

### Parameters

<i>defReaderCallbacks</i>	Struct for all callbacks.
<i>systemTreeNodePropertyCallback</i>	Function which should be called for all <a href="#">SystemTreeNodeProperty</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful  
[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid *defReaderCallbacks* argument

```
J.7.3.24 OTF2_ErrorCode OTF2_DefReaderCallbacks_SetUnknownCallback
( OTF2_DefReaderCallbacks * defReaderCallbacks,
  OTF2_DefReaderCallback_Unknown unknownCallback )
```

Registers the callback for an unknown definition.

### Parameters

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

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful  
[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid *defReaderCallbacks* argument

### J.8 OTF2\_DefWriter.h File Reference

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

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

#### TypeDefs

- **typedef struct OTF2\_DefWriter\_struct OTF2\_DefWriter**

*Handle definition for the external definition writer.*

#### Functions

- **OTF2\_ErrorCode OTF2\_DefWriter\_GetLocationID (const OTF2\_DefWriter \*writer, OTF2\_LocationRef \*location)**

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

- **OTF2\_ErrorCode OTF2\_DefWriter\_WriteAttribute (OTF2\_DefWriter \*writer, OTF2\_AttributeRef self, OTF2StringRef name, OTF2\_Type type)**

*Writes a Attribute definition record into the DefWriter.*

- **OTF2\_ErrorCode OTF2\_DefWriter\_WriteCallpath (OTF2\_DefWriter \*writer, OTF2\_CallpathRef self, OTF2\_CallpathRef parent, OTF2\_RegionRef region)**

*Writes a Callpath definition record into the DefWriter.*

- **OTF2\_ErrorCode OTF2\_DefWriter\_WriteCallsite (OTF2\_DefWriter \*writer, OTF2\_CallsiteRef self, OTF2StringRef sourceFile, uint32\_t lineNumber, OTF2\_RegionRef enteredRegion, OTF2\_RegionRef leftRegion)**

*Writes a Callsite definition record into the DefWriter.*

- **OTF2\_ErrorCode OTF2\_DefWriter\_WriteClockOffset (OTF2\_DefWriter \*writer, OTF2\_TimeStamp time, int64\_t offset, double standardDeviation)**

*Writes a ClockOffset definition record into the DefWriter.*

- **OTF2\_ErrorCode OTF2\_DefWriter\_WriteComm (OTF2\_DefWriter \*writer, OTF2\_CommRef self, OTF2StringRef 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, OTF2StringRef name, OTF2\_GroupType groupType,**

## J.8 OTF2\_DefWriter.h File Reference

---

`OTF2_Paradigm paradigm, OTF2_GroupFlag groupFlags, uint32_t numberOfMembers, const uint64_t *members)`

*Writes a Group definition record into the DefWriter.*

- `OTF2_ErrorCode OTF2_DefWriter_WriteLocation (OTF2_DefWriter *writer, OTF2_LocationRef self, OTF2StringRef 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, OTF2StringRef name, OTF2_LocationGroupType locationGroupType, OTF2_SystemTreeNodeRef systemTreeParent)`  
*Writes a LocationGroup definition record into the DefWriter.*

*Writes a MappingTable 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, OTF2RecorderKind recorderKind)`  
*Writes a MetricClass definition record into the DefWriter.*

*Writes a MetricClassRecorder 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.*

*Writes a MetricMember definition record into the DefWriter.*

- `OTF2_ErrorCode OTF2_DefWriter_WriteMetricMember (OTF2_DefWriter *writer, OTF2_MetricMemberRef self, OTF2StringRef name, OTF2StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2StringRef unit)`  
*Writes a MetricMember definition record into the DefWriter.*

*Writes a Parameter definition record into the DefWriter.*

- `OTF2_ErrorCode OTF2_DefWriter_WriteParameter (OTF2_DefWriter *writer, OTF2_ParameterRef self, OTF2StringRef name, OTF2_ParameterType parameterType)`  
*Writes a Parameter definition record into the DefWriter.*

*Writes a Region definition record into the DefWriter.*

- `OTF2_ErrorCode OTF2_DefWriter_WriteRegion (OTF2_DefWriter *writer, OTF2_RegionRef self, OTF2StringRef name, OTF2StringRef canonicalName, OTF2StringRef description, OTF2RegionRole regionRole, OTF2_Paradigm paradigm, OTF2RegionFlag regionFlags, OTF2StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)`  
*Writes a Region definition record into the DefWriter.*

## **APPENDIX J. FILE DOCUMENTATION**

---

*Writes a Region definition record into the DefWriter.*

- **OTF2\_ErrorCode OTF2\_DefWriter\_WriteRmaWin (OTF2\_DefWriter \*writer, OTF2\_RmaWinRef self, OTF2StringRef 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, OTF2StringRef name, OTF2StringRef 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, OTF2StringRef name, OTF2StringRef value)**

*Writes a SystemTreeNodeProperty definition record into the DefWriter.*

### **J.8.1 Detailed Description**

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

#### **Source Template:**

*templates/OTF2\_DefWriter tmpl.h*

### **J.8.2 Function Documentation**

#### **J.8.2.1 OTF2\_ErrorCode OTF2\_DefWriter\_GetLocationID ( const OTF2\_DefWriter \* writer, OTF2\_LocationRef \* location )**

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

#### **Parameters**

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

## J.8 OTF2\_DefWriter.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.8.2.2 OTF2\_ErrorCode OTF2\_DefWriter\_WriteAttribute ( OTF2\_DefWriter \*  
writer, OTF2\_AttributeRef self, OTF2StringRef name, OTF2\_Type  
type )**

Writes a Attribute definition record into the DefWriter.

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.8.2.3 OTF2\_ErrorCode OTF2\_DefWriter\_WriteCallpath ( OTF2\_DefWriter  
\* writer, OTF2\_CallpathRef self, OTF2\_CallpathRef parent,  
OTF2\_RegionRef region )**

Writes a Callpath definition record into the DefWriter.

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## APPENDIX J. FILE DOCUMENTATION

**J.8.2.4 OTF2\_ErrorCode OTF2\_DefWriter\_WriteCallsite ( OTF2\_DefWriter \* writer,  
OTF2\_CallsiteRef self, OTF2\_StringRef sourceFile, uint32\_t lineNumber,  
OTF2\_RegionRef enteredRegion, OTF2\_RegionRef leftRegion )**

Writes a Callsite definition record into the DefWriter.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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.

**J.8.2.5 OTF2\_ErrorCode OTF2\_DefWriter\_WriteClockOffset ( OTF2\_DefWriter \* writer, OTF2\_TimeStamp time, int64\_t offset, double standardDeviation )**

Writes a ClockOffset definition record into the DefWriter.

Clock offsets are used for clock corrections.

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## J.8 OTF2\_DefWriter.h File Reference

---

**J.8.2.6 OTF2\_ErrorCode OTF2\_DefWriter\_WriteComm ( OTF2\_DefWriter \* writer,  
OTF2\_CommRef self, OTF2StringRef name, OTF2\_GroupRef group,  
OTF2\_CommRef parent )**

Writes a Comm definition record into the DefWriter.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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 <i>OTF2_GROUP_TYPE_MPI_GROUP</i> or <i>OTF2_GROUP_TYPE_MPI_COMM_SELF</i> . References a <a href="#">Group</a> definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <i>OTF2_UNDEFINED_COMM</i> to indicate no parent. References a <a href="#">Comm</a> definition.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.8.2.7 OTF2\_ErrorCode OTF2\_DefWriter\_WriteGroup ( OTF2\_DefWriter \* writer,  
OTF2\_GroupRef self, OTF2StringRef name, OTF2\_GroupType  
groupType, OTF2\_Paradigm paradigm, OTF2\_GroupFlag groupFlags,  
uint32\_t numberofMembers, const uint64\_t \* members )**

Writes a Group definition record into the DefWriter.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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.

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>numberOfMembers</i>	The number of members in this group.
<i>members</i>	The identifiers of the group members.

### **Since**

Version 1.0

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.8.2.8 OTF2\_ErrorCode OTF2\_DefWriter\_WriteLocation ( OTF2\_DefWriter \* *writer*, OTF2\_LocationRef *self*, OTF2StringRef *name*, OTF2\_LocationType *locationType*, uint64\_t *numberOfEvents*, OTF2\_LocationGroupRef *locationGroup* )**

Writes a Location definition record into the DefWriter.

### **Parameters**

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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\_SUCCESS* if successful, an error code if an error occurs.

## J.8 OTF2\_DefWriter.h File Reference

---

**J.8.2.9 OTF2\_ErrorCode OTF2\_DefWriter\_WriteLocationGroup ( OTF2\_DefWriter \* writer, OTF2\_LocationGroupRef self, OTF2StringRef name, OTF2\_LocationGroupType locationGroupType, OTF2\_SystemTreeNodeRef systemTreeParent )**

Writes a LocationGroup definition record into the DefWriter.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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.

**J.8.2.10 OTF2\_ErrorCode OTF2\_DefWriter\_WriteMappingTable ( OTF2\_DefWriter \* writer, OTF2\_MappingType mappingType, const OTF2\_IdMap \* idMap )**

Writes a MappingTable definition record into the DefWriter.

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

### Parameters

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

### Since

Version 1.0

## APPENDIX J. FILE DOCUMENTATION

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.8.2.11 OTF2\_ErrorCode OTF2\_DefWriter\_WriteMetricClass ( OTF2\_DefWriter \* writer, OTF2\_MetricRef self, uint8\_t numberOfMetrics, const OTF2\_MetricMemberRef \* metricMembers, OTF2\_MetricOccurrence metricOccurrence, OTF2RecorderKind 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="#">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.

**J.8.2.12 OTF2\_ErrorCode OTF2\_DefWriter\_WriteMetricClassRecorder ( OTF2\_DefWriter \* writer, OTF2\_MetricRef metricClass, OTF2\_LocationRef recorder )**

Writes a MetricClassRecorder definition record into the DefWriter.

### Parameters

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

## J.8 OTF2\_DefWriter.h File Reference

---

<i>metricClass</i>	Parent <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.

**J.8.2.13 `OTF2_ErrorCode OTF2_DefWriter_WriteMetricInstance ( OTF2_DefWriter * writer, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope )`**

Writes a MetricInstance definition record into the DefWriter.

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

### Parameters

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

**J.8.2.14 `OTF2_ErrorCode OTF2_DefWriter.WriteMetricMember( OTF2_DefWriter * writer, OTF2_MetricMemberRef self, OTF2_StringRef name, OTF2_StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2_StringRef unit )`**

Writes a MetricMember definition record into the DefWriter.

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

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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: int64_t, uint64_t, or double.
<i>metricBase</i>	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor $\text{base}^{\text{exponent}}$ , to get the value in its base unit. For example, if the metric values come in as KiBi, than the base should be <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.

## J.8 OTF2\_DefWriter.h File Reference

---

**J.8.2.15 OTF2\_ErrorCode OTF2\_DefWriter\_WriteParameter ( OTF2\_DefWriter \* *writer*, OTF2\_ParameterRef *self*, OTF2StringRef *name*, OTF2\_ParameterType *parameterType* )**

Writes a Parameter definition record into the DefWriter.

### Parameters

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

**J.8.2.16 OTF2\_ErrorCode OTF2\_DefWriter\_WriteRegion ( OTF2\_DefWriter \* *writer*, OTF2\_RegionRef *self*, OTF2StringRef *name*, OTF2StringRef *canonicalName*, OTF2StringRef *description*, OTF2\_RegionRole *regionRole*, OTF2\_Paradigm *paradigm*, OTF2\_RegionFlag *regionFlags*, OTF2StringRef *sourceFile*, uint32\_t *beginLineNumber*, uint32\_t *endLineNumber* )**

Writes a Region definition record into the DefWriter.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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.

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>regionFlags</i>	Region flags. Since version 1.1.
<i>sourceFile</i>	The source file where this region was declared. References a <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.

**J.8.2.17 OTF2\_ErrorCode OTF2\_DefWriter\_WriteRmaWin ( OTF2\_DefWriter \* *writer*, OTF2\_RmaWinRef *self*, OTF2StringRef *name*, OTF2\_CommRef *comm* )**

Writes a RmaWin definition record into the DefWriter.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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.

## J.8 OTF2\_DefWriter.h File Reference

---

**J.8.2.18 OTF2\_ErrorCode OTF2\_DefWriter.WriteString ( OTF2\_DefWriter \* writer,  
OTF2\_StringRef self, const char \* string )**

Writes a String definition record into the DefWriter.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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.

**J.8.2.19 OTF2\_ErrorCode OTF2\_DefWriter.WriteSystemTreeNode ( OTF2\_DefWriter \* writer, OTF2\_SystemTreeNodeRef self, OTF2\_StringRef name, OTF2\_StringRef className, OTF2\_SystemTreeNodeRef parent )**

Writes a SystemTreeNode definition record into the DefWriter.

### Parameters

<i>writer</i>	Writer object.
<i>self</i>	The unique identifier for this <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\\_SUCCESS](#) if successful, an error code if an error occurs.

## **APPENDIX J. FILE DOCUMENTATION**

---

**J.8.2.20 OTF2\_ErrorCode OTF2\_DefWriter\_WriteSystemTreeNodeDomain (**  
**OTF2\_DefWriter \* writer, OTF2\_SystemTreeNodeRef systemTreeNode,**  
**OTF2\_SystemTreeDomain systemTreeDomain )**

Writes a SystemTreeNodeDomain definition record into the DefWriter.

### **Parameters**

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

### **Since**

Version 1.2

### **Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.8.2.21 OTF2\_ErrorCode OTF2\_DefWriter\_WriteSystemTreeNodeProperty (**  
**OTF2\_DefWriter \* writer, OTF2\_SystemTreeNodeRef systemTreeNode,**  
**OTF2StringRef name, OTF2StringRef value )**

Writes a SystemTreeNodeProperty definition record into the DefWriter.

### **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.

## **J.9 OTF2\_ErrorCodes.h File Reference**

---

### **J.9 OTF2\_ErrorCodes.h File Reference**

Error codes and error handling.

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

#### **Typedefs**

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

#### **Enumerations**

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

---

## **APPENDIX J. FILE DOCUMENTATION**

OTF2\_ERROR\_EDESTADDRREQ,  
OTF2\_ERROR\_EDOM,  
OTF2\_ERROR\_EDQUOT,  
OTF2\_ERROR\_EEXIST,  
OTF2\_ERROR\_EFAULT,  
OTF2\_ERROR\_EFBIG,  
OTF2\_ERROR\_EINPROGRESS,  
OTF2\_ERROR\_EINTR,  
OTF2\_ERROR\_EINVAL,  
OTF2\_ERROR\_EIO,  
OTF2\_ERROR\_EISCONN,  
OTF2\_ERROR\_EISDIR,  
OTF2\_ERROR\_ELOOP,  
OTF2\_ERROR\_EMFILE,  
OTF2\_ERROR\_EMLINK,  
OTF2\_ERROR\_EMSGSIZE,  
OTF2\_ERROR\_EMULTIHOP,  
OTF2\_ERROR\_ENAMETOOLONG,  
OTF2\_ERROR\_ENETDOWN,  
OTF2\_ERROR\_ENETRESET,  
OTF2\_ERROR\_ENETUNREACH,  
OTF2\_ERROR\_ENFILE,  
OTF2\_ERROR\_ENOBUFS,  
OTF2\_ERROR\_ENODATA,  
OTF2\_ERROR\_ENODEV,  
OTF2\_ERROR\_ENOENT,  
OTF2\_ERROR\_ENOEXEC,  
OTF2\_ERROR\_ENOLCK,  
OTF2\_ERROR\_ENOLINK,  
OTF2\_ERROR\_ENOMEM,  
OTF2\_ERROR\_ENOMSG,  
OTF2\_ERROR\_ENOPROTOOPT,  
OTF2\_ERROR\_ENOSPC,

---

## **J.9 OTF2\_ErrorCodes.h File Reference**

---

```
OTF2_ERROR_ENOSR,  
OTF2_ERROR_ENOSTR,  
OTF2_ERROR_ENOSYS,  
OTF2_ERROR_ENOTCONN,  
OTF2_ERROR_ENOTDIR,  
OTF2_ERROR_ENOTEMPTY,  
OTF2_ERROR_ENOTSOCK,  
OTF2_ERROR_ENOTSUP,  
OTF2_ERROR_ENOTTY,  
OTF2_ERROR_ENXIO,  
OTF2_ERROR_EOPNOTSUPP,  
OTF2_ERROR_EOVERFLOW,  
OTF2_ERROR_EPERM,  
OTF2_ERROR_EPIPE,  
OTF2_ERROR_EPROTO,  
OTF2_ERROR_EPROTONOSUPPORT,  
OTF2_ERROR_EPROTOTYPE,  
OTF2_ERROR_ERANGE,  
OTF2_ERROR_EROFS,  
OTF2_ERROR_ESPIPE,  
OTF2_ERROR_ESRCH,  
OTF2_ERROR_ESTALE,  
OTF2_ERRORETIME,  
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_MEMFAULT,  
OTF2_ERROR_MEM_ALLOC_FAILED,  
OTF2_ERROR_PROCESSED_WITH_FAULTS,  
OTF2_ERROR_INDEX_OUT_OF_BOUNDS,  
OTF2_ERROR_INVALID_FILENO,  
OTF2_ERROR_END_OF_BUFFER,  
OTF2_ERROR_FILE_INTERACTION,  
OTF2_ERROR_FILE_CAN_NOT_OPEN,  
OTF2_ERROR_INTERRUPTED_BY_CALLBACK,  
OTF2_ERROR_PROPERTY_NAME_INVALID,  
OTF2_ERROR_PROPERTY_EXISTS,  
OTF2_ERROR_PROPERTY_NOT_FOUND,  
OTF2_ERROR_PROPERTY_VALUE_INVALID,  
OTF2_ERROR_FILE_COMPRESSION_NOT_SUPPORTED,  
OTF2_ERROR_DUPLICATE_MAPPING_TABLE,  
OTF2_ERROR_INVALID_FILE_MODE_TRANSITION }
```

### **Functions**

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

#### **J.9.1 Detailed Description**

Error codes and error handling.

#### **Maintainer:**

Daniel Lorenz <[d.lorenz@fz-juelich.de](mailto:d.lorenz@fz-juelich.de)>

#### **File Status:**

ALPHA

#### **Author**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

## J.9 OTF2\_ErrorCodes.h File Reference

---

### J.9.2 Typedef Documentation

J.9.2.1 **typedef OTF2\_ErrorCode( \* OTF2\_ErrorCallback)(void \*userData, const char \*file, uint64\_t line, const char \*function, OTF2\_ErrorCode errorCode, const char \*msgFormatString, va\_list va)**

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

#### Parameters

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

#### Returns

Should return the errorCode

### J.9.3 Enumeration Type Documentation

#### J.9.3.1 enum OTF2\_ErrorCode

This is the list of error codes for OTF2.

##### Enumerator:

**OTF2\_DEPRECATED** Special marker for error messages which indicates an deprecation.

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

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

**OTF2\_SUCCESS** Operation successful

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

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

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

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

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

---

## **APPENDIX J. FILE DOCUMENTATION**

***OTF2\_ERROR\_EAFNOSUPPORT*** Address family is not supported  
***OTF2\_ERROR\_EAGAIN*** Resource temporaly not available  
***OTF2\_ERROR\_EALREADY*** Connection is already processed  
***OTF2\_ERROR\_EBADF*** Invalid file pointer  
***OTF2\_ERROR\_EBADMSG*** Invalid message  
***OTF2\_ERROR\_EBUSY*** Resource or device is busy  
***OTF2\_ERROR\_ECANCELED*** Operation was aborted  
***OTF2\_ERROR\_ECHILD*** No child process available  
***OTF2\_ERROR\_ECONNREFUSED*** Connection was refused  
***OTF2\_ERROR\_ECONNRESET*** Connection was reset  
***OTF2\_ERROR\_EDEADLK*** Resolved deadlock  
***OTF2\_ERROR\_EDESTADDRREQ*** Destination address was expected  
***OTF2\_ERROR\_EDOM*** Domain error  
***OTF2\_ERROR\_EDQUOT*** Reserved  
***OTF2\_ERROR\_EEXIST*** File does already exist  
***OTF2\_ERROR\_EFAULT*** Invalid Address  
***OTF2\_ERROR\_EFBIG*** File is to big  
***OTF2\_ERROR\_EINPROGRESS*** Operation is work in progress  
***OTF2\_ERROR\_EINTR*** 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*** To many layers of symbolic links  
***OTF2\_ERROR\_EMFILE*** To many opened files  
***OTF2\_ERROR\_EMLINK*** To many links  
***OTF2\_ERROR\_EMSGSIZE*** Message buffer is to small  
***OTF2\_ERROR\_EMULTIHOP*** Reserved  
***OTF2\_ERROR\_ENAMETOOLONG*** Filename is to long  
***OTF2\_ERROR\_ENETDOWN*** Network is down  
***OTF2\_ERROR\_ENETRESET*** Connection was reset from the network  
***OTF2\_ERROR\_ENETUNREACH*** Network is not reachable  
***OTF2\_ERROR\_ENFILE*** To much opened files  
***OTF2\_ERROR\_ENOBUFS*** No buffer space available

---

## J.9 OTF2\_ErrorCodes.h File Reference

---

***OTF2\_ERROR\_ENODATA*** No more data left in the queue  
***OTF2\_ERROR\_ENODEV*** This device does not support this function  
***OTF2\_ERROR\_ENOENT*** File or Directory does not exist  
***OTF2\_ERROR\_ENOEXEC*** Can not execute binary  
***OTF2\_ERROR\_ENOLCK*** Locking failed  
***OTF2\_ERROR\_ENOLINK*** Reserved  
***OTF2\_ERROR\_ENOMEM*** Not enough main memory available  
***OTF2\_ERROR\_ENOMSG*** Message has not the expected type  
***OTF2\_ERROR\_ENOPROTOOPT*** This protocol is not available  
***OTF2\_ERROR\_ENOSPC*** No space left on device  
***OTF2\_ERROR\_ENOSR*** No stream available  
***OTF2\_ERROR\_ENOSTR*** This is not a stream  
***OTF2\_ERROR\_ENOSYS*** Requested function is not implemented  
***OTF2\_ERROR\_ENOTCONN*** Socket is not connected  
***OTF2\_ERROR\_ENOTDIR*** This is not an directory  
***OTF2\_ERROR\_ENOTEMPTY*** This directory is not empty  
***OTF2\_ERROR\_ENOTSOCK*** No socket  
***OTF2\_ERROR\_ENOTSUP*** This operation is not supported  
***OTF2\_ERROR\_ENOTTY*** This IOCTL is not supported by the device  
***OTF2\_ERROR\_ENXIO*** Device is not yet configured  
***OTF2\_ERROR\_EOPNOTSUPP*** Operation is not supported by this socket  
***OTF2\_ERROR\_EOVERFLOW*** Value is to long for the datatype  
***OTF2\_ERROR\_EPERM*** Operation is not permitted  
***OTF2\_ERROR\_EPIPE*** Broken pipe  
***OTF2\_ERROR\_EPROTO*** Protocoll error  
***OTF2\_ERROR\_EPROTONOSUPPORT*** Protocoll is not supported  
***OTF2\_ERROR\_EPROTOTYPE*** Wrong protocoll type for this socket  
***OTF2\_ERROR\_ERANGE*** Value is out of range  
***OTF2\_ERROR\_EROFS*** Filesystem is read only  
***OTF2\_ERROR\_ESPIPE*** This seek is not allowed  
***OTF2\_ERROR\_ESRCH*** No matching process found  
***OTF2\_ERROR\_ESTALE*** Reserved  
***OTF2\_ERRORETIME*** Timeout in file stream or IOCTL  
***OTF2\_ERROR\_ETIMEDOUT*** Connection timed out

## **APPENDIX J. FILE DOCUMENTATION**

---

***OTF2\_ERROR\_ETXTBSY*** File couldn't be executed while it is opened  
***OTF2\_ERROR\_EWOULDBLOCK*** Operation would be blocking  
***OTF2\_ERROR\_EXDEV*** Invalid link between devices  
***OTF2\_ERROR\_END\_OF\_FUNCTION*** Unintentional reached end of function  
***OTF2\_ERROR\_INVALID\_CALL*** Function call not allowed in current state

***OTF2\_ERROR\_INVALID\_ARGUMENT*** Parameter value out of range  
***OTF2\_ERROR\_INVALID\_RECORD*** Invalid definition or event record  
***OTF2\_ERROR\_INVALID\_DATA*** Invalid or inconsistent record data  
***OTF2\_ERROR\_INVALID\_SIZE\_GIVEN*** The given size can not be used  
***OTF2\_ERROR\_UNKNOWN\_TYPE*** The given type is not known  
***OTF2\_ERROR\_INTEGRITY\_FAULT*** The structural integrity is not given  
***OTF2\_ERROR\_MEM\_FAULT*** This could not be done with the given memory  
***OTF2\_ERROR\_MEM\_ALLOC\_FAILED*** Memory allocation failed  
***OTF2\_ERROR\_PROCESSED\_WITHFAULTS*** An error appeared when data was processed

***OTF2\_ERROR\_INDEX\_OUT\_OF\_BOUNDS*** Index out of bounds  
***OTF2\_ERROR\_INVALID\_LINENO*** Invalid source code line number  
***OTF2\_ERROR\_END\_OF\_BUFFER*** End of buffer/file reached  
***OTF2\_ERROR\_FILE\_INTERACTION*** Invalid file operation  
***OTF2\_ERROR\_FILE\_CAN\_NOT\_OPEN*** Unable to open file  
***OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK*** Record reading interrupted by reader callback

***OTF2\_ERROR\_PROPERTY\_NAME\_INVALID*** Property name does not conform to the naming scheme  
***OTF2\_ERROR\_PROPERTY\_EXISTS*** Property already exists  
***OTF2\_ERROR\_PROPERTY\_NOT\_FOUND*** Property not found found in this archive

***OTF2\_ERROR\_PROPERTY\_VALUE\_INVALID*** Property value does not have the expected value

***OTF2\_ERROR\_FILE\_COMPRESSION\_NOT\_SUPPORTED*** Missing library support for requested compression mode

***OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE*** Multiple definitions for the same mapping type

***OTF2\_ERROR\_INVALID\_FILE\_MODE\_TRANSITION*** File mode transition not permitted

## J.9 OTF2\_ErrorCodes.h File Reference

---

### J.9.4 Function Documentation

#### J.9.4.1 const char\* OTF2\_Error\_GetDescription ( OTF2\_ErrorCode errorCode )

Returns the description of an error code.

##### Parameters

<i>errorCode</i>	: Error Code
------------------	--------------

##### Returns

Returns the description of a known error code.

#### J.9.4.2 const char\* OTF2\_Error\_GetName ( OTF2\_ErrorCode errorCode )

Returns the name of an error code.

##### Parameters

<i>errorCode</i>	: Error Code
------------------	--------------

##### Returns

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

#### J.9.4.3 OTF2\_ErrorCallback OTF2\_Error\_RegisterCallback ( OTF2\_ErrorCallback errorCallbackIn, void \* userData )

Register a programmers callback function for error handling.

##### Parameters

<i>errorCallbackIn</i>	: Function will be called instead of printing a default message to standard error.
<i>userData</i>	: Data pointer passed to the callback.

##### Returns

Function pointer to the former error handling function.

### J.10 OTF2\_Events.h File Reference

Enums and types used in event records.

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

#### Data Structures

- union [OTF2\\_MetricValue](#)

*Metric value.*

#### Typedefs

- [typedef uint8\\_t OTF2\\_CollectiveOp](#)  
*Wrapper for enum OTF2\_CollectiveOp\_enum.*
- [typedef uint8\\_t OTF2\\_LockType](#)  
*Wrapper for enum OTF2\_LockType\_enum.*
- [typedef uint8\\_t OTF2\\_MeasurementMode](#)  
*Wrapper for enum OTF2\_MeasurementMode\_enum.*
- [typedef uint8\\_t OTF2\\_RmaAtomicType](#)  
*Wrapper for enum OTF2\_RmaAtomicType\_enum.*
- [typedef uint32\\_t OTF2\\_RmaSyncLevel](#)  
*Wrapper for enum OTF2\_RmaSyncLevel\_enum.*
- [typedef uint8\\_t OTF2\\_RmaSyncType](#)  
*Wrapper for enum OTF2\_RmaSyncType\_enum.*

#### Enumerations

- enum [OTF2\\_CollectiveOp\\_enum](#) { ,  
    [OTF2\\_COLLECTIVE\\_OP\\_CREATE\\_HANDLE](#) = 17,  
    [OTF2\\_COLLECTIVE\\_OP\\_DESTROY\\_HANDLE](#) = 18,  
    [OTF2\\_COLLECTIVE\\_OP\\_ALLOCATE](#) = 19,  
    [OTF2\\_COLLECTIVE\\_OP\\_DEALLOCATE](#) = 20,  
    [OTF2\\_COLLECTIVE\\_OP\\_CREATE\\_HANDLE\\_AND\\_ALLOCATE](#) = 21,  
    [OTF2\\_COLLECTIVE\\_OP\\_DESTROY\\_HANDLE\\_AND\\_DEALLOCATE](#) =  
    22 }

*Types of collective operations.*

## J.10 OTF2\_Events.h File Reference

---

- enum `OTF2_LockType_enum` {  
    `OTF2_LOCK_EXCLUSIVE` = 0,  
    `OTF2_LOCK_SHARED` = 1 }  
  
    *General Lock Type.*
- enum `OTF2_MeasurementMode_enum` {  
    `OTF2_MEASUREMENT_ON` = 1,  
    `OTF2_MEASUREMENT_OFF` = 2 }  
  
    *Types for use in the MeasurementOnOff event.*
- enum `OTF2_RmaAtomicType_enum`  
  
    *RMA Atomic Operation Type.*
- enum `OTF2_RmaSyncLevel_enum` {  
    `OTF2_RMA_SYNC_LEVEL_NONE` = 0,  
    `OTF2_RMA_SYNC_LEVEL_PROCESS` = ( 1 << 0 ),  
    `OTF2_RMA_SYNC_LEVEL_MEMORY` = ( 1 << 1 ) }  
  
    *Synchronization level used in RMA synchronization records.*
- enum `OTF2_RmaSyncType_enum` {  
    `OTF2_RMA_SYNC_TYPE_MEMORY` = 0,  
    `OTF2_RMA_SYNC_TYPE_NOTIFY_IN` = 1,  
    `OTF2_RMA_SYNC_TYPE_NOTIFY_OUT` = 2 }  
  
    *Type of direct RMA synchronization call.*

### J.10.1 Detailed Description

Enums and types used in event records.

#### Source Template:

*templates/OTF2\_Events.tmpl.h*

#### Maintainer:

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

#### Authors

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### **J.10.2 Enumeration Type Documentation**

#### **J.10.2.1 enum OTF2\_CollectiveOp\_enum**

Types of collective operations.

##### **Since**

Version 1.0

##### **Enumerator:**

***OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE*** Collectively create a handle (ie. MPI\_Win, MPI\_Comm, MPI\_File).

***OTF2\_COLLECTIVE\_OP\_DESTROY\_HANDLE*** Collectively destroy a handle.

***OTF2\_COLLECTIVE\_OP\_ALLOCATE*** Collectively allocate memory.

***OTF2\_COLLECTIVE\_OP\_DEALLOCATE*** Collectively deallocate memory.

***OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE\_AND\_ALLOCATE*** Collectively create a handle and allocate memory.

***OTF2\_COLLECTIVE\_OP\_DESTROY\_HANDLE\_AND\_DEALLOCATE*** Collectively destroy a handle and deallocate memory.

#### **J.10.2.2 enum OTF2\_LockType\_enum**

General Lock Type.

##### **Since**

Version 1.2

##### **Enumerator:**

***OTF2\_LOCK\_EXCLUSIVE*** Exclusive lock. No other lock will be granted.

***OTF2\_LOCK\_SHARED*** Shared lock. Other shared locks will be granted, but no exclusive locks.

## **J.10 OTF2\_Events.h File Reference**

---

### **J.10.2.3 enum OTF2\_MeasurementMode\_enum**

Types for use in the MeasurementOnOff event.

#### **Since**

Version 1.0

#### **Enumerator:**

***OTF2\_MEASUREMENT\_ON*** The measurement resumed with event recording.

***OTF2\_MEASUREMENT\_OFF*** The measurement suspended with event recording.

### **J.10.2.4 enum OTF2\_RmaAtomicType\_enum**

RMA Atomic Operation Type.

#### **Since**

Version 1.2

### **J.10.2.5 enum OTF2\_RmaSyncLevel\_enum**

Synchronization level used in RMA synchronization records.

#### **Since**

Version 1.2

#### **Enumerator:**

***OTF2\_RMA\_SYNC\_LEVEL\_NONE*** No process synchronization or access completion (e.g., MPI\_Win\_post, MPI\_Win\_start).

***OTF2\_RMA\_SYNC\_LEVEL\_PROCESS*** Synchronize processes (e.g., MPI\_Win\_create/free).

***OTF2\_RMA\_SYNC\_LEVEL\_MEMORY*** Complete memory accesses (e.g., MPI\_Win\_complete, MPI\_Win\_wait).

### **J.10.2.6 enum OTF2\_RmaSyncType\_enum**

Type of direct RMA synchronization call.

#### **Since**

Version 1.2

#### **Enumerator:**

***OTF2\_RMA\_SYNC\_TYPE\_MEMORY*** Synchronize memory copy.

***OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_IN*** Incoming remote notification.

***OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_OUT*** Outgoing remote notification.

## **J.11 OTF2\_EvtReader.h File Reference**

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

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_EvtReaderCallbacks.h>
```

#### **Functions**

- **OTF2\_ErrorCode OTF2\_EvtReader\_ApplyClockOffsets (OTF2\_EvtReader \*reader, bool action)**  
*Enable or disable applying of the clock offset to event timestamps read from this event reader.*
- **OTF2\_ErrorCode OTF2\_EvtReader\_ApplyMappingTables (OTF2\_EvtReader \*reader, bool action)**  
*Enable or disable applying of the mapping tables to events read from this event reader.*
- **OTF2\_ErrorCode OTF2\_EvtReader\_GetLocationID (const OTF2\_EvtReader \*reader, OTF2\_LocationRef \*location)**  
*Return the location ID of the reading related location.*
- **OTF2\_ErrorCode OTF2\_EvtReader\_GetPos (OTF2\_EvtReader \*reader, uint64\_t \*position)**

## J.11 OTF2\_EvtReader.h File Reference

---

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

- **OTF2\_ErrorCode OTF2\_EvtReader\_ReadEvents (OTF2\_EvtReader \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)**

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

- **OTF2\_ErrorCode OTF2\_EvtReader\_ReadEventsBackward (OTF2\_EvtReader \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)**

This functions reads recordsRead events backwards from the current position.

- **OTF2\_ErrorCode OTF2\_EvtReader\_Seek (OTF2\_EvtReader \*reader, uint64\_t position)**

Seek jumps to an event position.

- **OTF2\_ErrorCode OTF2\_EvtReader\_SetCallbacks (OTF2\_EvtReader \*reader, const OTF2\_EvtReaderCallbacks \*callbacks, void \*userData)**

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

- **OTF2\_ErrorCode OTF2\_EvtReader\_TimeStampRewrite (OTF2\_EvtReader \*reader, OTF2\_TimeStamp time)**

The following function rewrites the timestamp from the event on the actual reading position if the buffer is in OTF2\_BUFFER MODIFY mode. It also modifies the timestamp for all other events in the same timestamp bundle. This function also has to keep track that not only the last timestamp, but all records equal to the last timestamp has to be modified. This is done by a position list, if there has no seek appeared before. In this case a position list can be easily generated because of that the reader has seen all related timestamps before. This not the case if there has a seek appeared before. In this case the related timestamp positions are generated by a linear search.

### J.11.1 Detailed Description

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

#### Maintainer:

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

#### Authors

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

## **APPENDIX J. FILE DOCUMENTATION**

---

### **J.11.2 Function Documentation**

#### **J.11.2.1 OTF2\_ErrorCode OTF2\_EvtReader\_ApplyClockOffsets ( OTF2\_EvtReader \* reader, bool action )**

Enable or disable applying of the clock offset to event timestamps read from this event reader.

This setting has no effect if the 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.

#### **J.11.2.2 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.

#### **J.11.2.3 OTF2\_ErrorCode OTF2\_EvtReader\_GetLocationID ( const OTF2\_EvtReader \* reader, OTF2\_LocationRef \* location )**

Return the location ID of the reading related location.

##### **Parameters**

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

## J.11 OTF2\_EvtReader.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.11.2.4 OTF2\_ErrorCode OTF2\_EvtReader\_GetPos ( OTF2\_EvtReader \* *reader*, uint64\_t \* *position* )

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

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.11.2.5 OTF2\_ErrorCode OTF2\_EvtReader\_ReadEvents ( OTF2\_EvtReader \* *reader*, uint64\_t *recordsToRead*, uint64\_t \* *recordsRead* )

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

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.11.2.6 OTF2\_ErrorCode OTF2\_EvtReader\_ReadEventsBackward ( OTF2\_EvtReader \* *reader*, uint64\_t *recordsToRead*, uint64\_t \* *recordsRead* )

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

## APPENDIX J. FILE DOCUMENTATION

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.11.2.7 OTF2\_ErrorCode OTF2\_EvtReader\_Seek ( OTF2\_EvtReader \* *reader*, uint64\_t *position* )

Seek jumps to an event position.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.11.2.8 OTF2\_ErrorCode OTF2\_EvtReader\_SetCallbacks ( OTF2\_EvtReader \* *reader*, const OTF2\_EvtReaderCallbacks \* *callbacks*, void \* *userData* )

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

These callbacks are ignored, if the events are read by a 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="#">OTF2_EvtReaderCallbacks_New</a> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.11.2.9 OTF2\_ErrorCode OTF2\_EvtReader\_TimeStampRewrite ( OTF2\_EvtReader \* reader, OTF2\_TimeStamp time )

The following function rewrites the timestamp from the event on the actual reading position if the buffer is in OTF2\_BUFFER MODIFY mode. It also modifies the timestamp for all other events in the same timestamp bundle. This function also has to keep track that not only the last timestamp, but all records equal to the last timestamp has to be modified. This is done by a position list, if there has no seek appeared before. In this case a position list can be easily generated because of that the reader has seen all related timestamps before. This not the case if there has a seek appeared before. In this case the related timestamp positions are generated by a linear search.

### Parameters

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

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

This defines the callbacks for the event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

### Typedefs

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_BufferFlush )(OTF2_-LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp stopTime)`

---

## **APPENDIX J. FILE DOCUMENTATION**

*Callback for the BufferFlush event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_Enter )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

*Callback for the Enter event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_Leave )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

*Callback for the Leave event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_MeasurementOnOff )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_MeasurementMode measurementMode)`

*Callback for the MeasurementOnOff event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_Metric )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)`

*Callback for the Metric event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_MpiCollectiveBegin )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)`

*Callback for the MpiCollectiveBegin event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_MpiCollectiveEnd )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)`

*Callback for the MpiCollectiveEnd event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_MpiIrecv )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`

*Callback for the MpiIrecv event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_MpiIrecvRequest )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

*Callback for the MpiIrecvRequest event record.*

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

*Callback for the MpiIsend event record.*

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

*Callback for the MpiIsendComplete event record.*

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

*Callback for the MpiRecv event record.*

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

*Callback for the MpiRequestCancelled event record.*

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

*Callback for the MpiRequestTest event record.*

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

*Callback for the MpiSend event record.*

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

*Callback for the OmpAcquireLock event record.*

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

*Callback for the OmpFork event record.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **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, OTF2StringRef string)**  
*Callback for the ParameterString event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_ParameterUnsignedInt )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, uint64\_t value)**  
*Callback for the ParameterUnsignedInt event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaAcquireLock )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType)**  
*Callback for the RmaAcquireLock event record.*

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaAtomic )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)`

*Callback for the RmaAtomic event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaCollectiveBegin )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList)`

*Callback for the RmaCollectiveBegin event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaCollectiveEnd )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)`

*Callback for the RmaCollectiveEnd event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaGet )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)`

*Callback for the RmaGet event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaGroupSync )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)`

*Callback for the RmaGroupSync event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaOpCompleteBlocking )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

*Callback for the RmaOpCompleteBlocking event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

*Callback for the RmaOpCompleteNonBlocking event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaOpCompleteRemote )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

*Callback for the RmaOpCompleteRemote event record.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaOpTest )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)**

*Callback for the RmaOpTest event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaPut )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t bytes, uint64\_t matchingId)**

*Callback for the RmaPut event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaReleaseLock )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId)**

*Callback for the RmaReleaseLock event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaRequestLock )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType)**

*Callback for the RmaRequestLock event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaSync )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, OTF2\_RmaSyncType syncType)**

*Callback for the RmaSync event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaTryLock )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType)**

*Callback for the RmaTryLock event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaWaitChange )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)**

*Callback for the RmaWaitChange event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_RmaWinCreate )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)**

*Callback for the RmaWinCreate event record.*

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_RmaWinDestroy )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win)`

*Callback for the RmaWinDestroy event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadAcquireLock )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

*Callback for the ThreadAcquireLock event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadFork )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)`

*Callback for the ThreadFork event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadJoin )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model)`

*Callback for the ThreadJoin event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadReleaseLock )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

*Callback for the ThreadReleaseLock event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadTaskComplete )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskComplete event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadTaskCreate )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskCreate event record.*

- `typedef OTF2_CallbackCode(* OTF2_EvtReaderCallback_ThreadTaskSwitch )(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Callback for the ThreadTaskSwitch event record.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_ThreadTeamBegin)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam)**  
*Callback for the ThreadTeamBegin event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_ThreadTeamEnd)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam)**  
*Callback for the ThreadTeamEnd event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_EvtReaderCallback\_Unknown )(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList)**  
*Callback for an unknown event record.*
- **typedef struct OTF2\_EvtReaderCallbacks\_struct OTF2\_EvtReaderCallbacks**

*Opaque struct which holds all event record callbacks.*

### **Functions**

- **void OTF2\_EvtReaderCallbacks\_Clear (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks)**  
*Clears a struct for the event callbacks.*
- **void OTF2\_EvtReaderCallbacks\_Delete (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks)**  
*Deallocates a struct for the event callbacks.*
- **OTF2\_EvtReaderCallbacks \* OTF2\_EvtReaderCallbacks\_New (void)**  
*Allocates a new struct for the event callbacks.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetBufferFlushCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_BufferFlush bufferFlushCallback)**  
*Registers the callback for the BufferFlush event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetEnterCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_Enter enterCallback)**  
*Registers the callback for the Enter event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetLeaveCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_Leave leaveCallback)**  
*Registers the callback for the Leave event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMeasurementOnOffCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_MeasurementOnOff measurementOnOffCallback)**

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

*Registers the callback for the MeasurementOnOff event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMetricCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_Metric metricCallback)`

*Registers the callback for the Metric event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiCollectiveBegin mpiCollectiveBeginCallback)`

*Registers the callback for the MpiCollectiveBegin event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiCollectiveEnd mpiCollectiveEndCallback)`

*Registers the callback for the MpiCollectiveEnd event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIrecvCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiIrecv mpiIrecvCallback)`

*Registers the callback for the MpiIrecv event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIrecvRequestCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiIrecvRequest mpiIrecvRequestCallback)`

*Registers the callback for the MpiIrecvRequest event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIsendCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiIsend mpiIsendCallback)`

*Registers the callback for the MpiIsend event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiIsendCompleteCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiIsendComplete mpiIsendCompleteCallback)`

*Registers the callback for the MpiIsendComplete event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRecvCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiRecv mpiRecvCallback)`

*Registers the callback for the MpiRecv event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRequestCancelledCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiRequestCancelled mpiRequestCancelledCallback)`

*Registers the callback for the MpiRequestCancelled event.*

- `OTF2_ErrorCode OTF2_EvtReaderCallbacks_SetMpiRequestTestCallback (OTF2_EvtReaderCallbacks *evtReaderCallbacks, OTF2_EvtReaderCallback_MpiRequestTest mpiRequestTestCallback)`

*Registers the callback for the MpiRequestTest event.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpISendCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_MpISend mpiSendCallback)**

*Registers the callback for the MpISend event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpAcquireLockCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_OmpAcquireLock ompAcquireLockCallback)**

*Registers the callback for the OmpAcquireLock event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpForkCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_OmpFork ompForkCallback)**

*Registers the callback for the OmpFork event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpJoinCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_OmpJoin ompJoinCallback)**

*Registers the callback for the OmpJoin event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpReleaseLockCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_OmpReleaseLock ompReleaseLockCallback)**

*Registers the callback for the OmpReleaseLock event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpTaskCompleteCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_OmpTaskComplete ompTaskCompleteCallback)**

*Registers the callback for the OmpTaskComplete event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpTaskCreateCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_OmpTaskCreate ompTaskCreateCallback)**

*Registers the callback for the OmpTaskCreate event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpTaskSwitchCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_OmpTaskSwitch ompTaskSwitchCallback)**

*Registers the callback for the OmpTaskSwitch event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetParameterIntCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_ParameterInt parameterIntCallback)**

*Registers the callback for the ParameterInt event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetParameterStringCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_ParameterString parameterStringCallback)**

*Registers the callback for the ParameterString event.*

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetParameterUnsignedIntCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_ParameterUnsignedInt parameterUnsignedInt parameterUnsignedIntCallback)**  
*Registers the callback for the ParameterUnsignedInt event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaAcquireLockCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaAcquireLock rmaAcquireLockCallback)**  
*Registers the callback for the RmaAcquireLock event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaAtomicCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaAtomic rmaAtomicCallback)**  
*Registers the callback for the RmaAtomic event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaCollectiveBeginCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaCollectiveBegin rmaCollectiveBeginCallback)**  
*Registers the callback for the RmaCollectiveBegin event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaCollectiveEndCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaCollectiveEnd rmaCollectiveEndCallback)**  
*Registers the callback for the RmaCollectiveEnd event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaGetCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaGet rmaGetCallback)**  
*Registers the callback for the RmaGet event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaGroupSyncCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaGroupSync rmaGroupSyncCallback)**  
*Registers the callback for the RmaGroupSync event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaOpCompleteBlockingCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaOpCompleteBlocking rmaOpCompleteBlockingCallback)**  
*Registers the callback for the RmaOpCompleteBlocking event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaOpCompleteNonBlockingCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaOpCompleteNonBlocking rmaOpCompleteNonBlockingCallback)**  
*Registers the callback for the RmaOpCompleteNonBlocking event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaOpCompleteRemoteCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaOpCompleteRemote rmaOpCompleteRemoteCallback)**  
*Registers the callback for the RmaOpCompleteRemote event.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaOpTestCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaOpTest rmaOpTestCallback)**

*Registers the callback for the RmaOpTest event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaPutCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaPut rmaPutCallback)**

*Registers the callback for the RmaPut event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaReleaseLockCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaReleaseLock rmaReleaseLockCallback)**

*Registers the callback for the RmaReleaseLock event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaRequestLockCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaRequestLock rmaRequestLockCallback)**

*Registers the callback for the RmaRequestLock event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaSyncCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaSync rmaSyncCallback)**

*Registers the callback for the RmaSync event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaTryLockCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaTryLock rmaTryLockCallback)**

*Registers the callback for the RmaTryLock event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaWaitChangeCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaWaitChange rmaWaitChangeCallback)**

*Registers the callback for the RmaWaitChange event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaWinCreateCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaWinCreate rmaWinCreateCallback)**

*Registers the callback for the RmaWinCreate event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaWinDestroyCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaWinDestroy rmaWinDestroyCallback)**

*Registers the callback for the RmaWinDestroy event.*
- **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadAcquireLockCallback (OTF2\_EvtReaderCallbacks \*evtReaderCallbacks, OTF2\_EvtReaderCallback\_ThreadAcquireLock threadAcquireLockCallback)**

*Registers the callback for the ThreadAcquireLock event.*

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

### J.12.1 Detailed Description

This defines the callbacks for the event reader.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Source Template:**

*templates/OTF2\_EvtReaderCallbacks.tpl.h*

### **Maintainer:**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### **J.12.2 Typedef Documentation**

**J.12.2.1 `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-BufferFlush)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp stopTime)`**

Callback for the BufferFlush event record.

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

### **Parameters**

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

J.12.2.2 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-Enter)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RegionRef region)**

Callback for the Enter event record.

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

### Parameters

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

J.12.2.3 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-Leave)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RegionRef region)**

Callback for the Leave event record.

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

### Parameters

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

## APPENDIX J. FILE DOCUMENTATION

---

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

```
J.12.2.4 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-  
  MeasurementOnOff)(OTF2_LocationRef location, OTF2_TimeStamp  
  time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
  *attributeList, OTF2_MeasurementMode measurementMode)
```

Callback for the MeasurementOnOff event record.

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

### Parameters

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

```
J.12.2.5 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
Metric)(OTF2_LocationRef location, OTF2_TimeStamp time,  
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type  
*typeIDs, const OTF2_MetricValue *metricValues)
```

Callback for the Metric event record.

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

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.12.2.6 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp  
time, uint64_t eventPosition, void *userData, OTF2_AttributeList  
*attributeList)
```

Callback for the MpiCollectiveBegin event record.

## APPENDIX J. FILE DOCUMENTATION

---

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

### Parameters

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

```
J.12.2.7 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)
```

Callback for the MpiCollectiveEnd event record.

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

### Parameters

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

<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 <i>communicator</i> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.12.2.8 `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_MpiIrecv)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`**

Callback for the MpiIrecv event record.

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

### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.0

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.12.2.9** **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-  
MpiIrecvRequest)(OTF2\_LocationRef location, OTF2\_TimeStamp  
time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList  
\*attributeList, uint64\_t requestID)**

Callback for the MpiIrecvRequest event record.

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

### **Parameters**

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

### **Since**

Version 1.0

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.12.2.10** **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-  
MpiIsend)(OTF2\_LocationRef location, OTF2\_TimeStamp time,  
uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList,  
uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t  
msgLength, uint64\_t requestID)**

Callback for the MpiIsend event record.

A MpiIsend record indicates that a MPI message send process was initiated (MPI\_-ISEND). It keeps the necessary information for this event: receiver of the message,

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.12.2.11** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

Callback for the MpiIsendComplete event record.

Signals the completion of non-blocking send request.

### Parameters

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

## APPENDIX J. FILE DOCUMENTATION

---

<i>attributeList</i>	Additional attributes for this event.
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.12.2.12 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-  
MpRecv)(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 MpRecv event record.

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

### Parameters

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.12.2.13 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_MpiRequestCancelled)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)**

Callback for the MpiRequestCancelled event record.

This events appears if the program canceled a request.

### Parameters

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

J.12.2.14 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_MpiRequestTest)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)**

Callback for the MpiRequestTest event record.

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

### Parameters

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

## APPENDIX J. FILE DOCUMENTATION

---

<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <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](#).

```
J.12.2.15 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
    MpISend)(OTF2_LocationRef location, OTF2_TimeStamp time,
    uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,
    uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t
    msgLength)
```

Callback for the MpISend event record.

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

**Parameters**

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.12.2.16 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
OmpAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, uint32_t lockID, uint32_t acquisitionOrder)
```

Callback for the OmpAcquireLock event record.

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

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

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterEvtCallbacks</i> or <i>OTF2_EvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## APPENDIX J. FILE DOCUMENTATION

J.12.2.17 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_OmpFork)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t numberOfRequestedThreads)**

Callback for the OmpFork event record.

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

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

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_EvtReader_SetCallbacks</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](#).

J.12.2.18 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_OmpJoin)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList)**

Callback for the OmpJoin event record.

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

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

### Parameters

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

**J.12.2.19** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_OmpReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the OmpReleaseLock event record.

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

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

### Parameters

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

```
J.12.2.20 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
OmpTaskComplete)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, uint64_t taskID)
```

Callback for the OmpTaskComplete event record.

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

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

### Parameters

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

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.12.2.21 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, uint64_t taskID)
```

Callback for the OmpTaskCreate event record.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

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

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.12.2.22** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, uint64_t taskID)`

Callback for the OmpTaskSwitch event record.

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

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

### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the now active task instance.

### **Since**

Version 1.0

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.12.2.23** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, int64_t value)`

Callback for the ParameterInt event record.

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

### **Parameters**

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

### **Since**

Version 1.0

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

J.12.2.24 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-ParameterString)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, OTF2StringRef 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](#).

J.12.2.25 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-ParameterUnsignedInt)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, uint64\_t value)**

Callback for the ParameterUnsignedInt event record.

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

## APPENDIX J. FILE DOCUMENTATION

---

### Parameters

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

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.12.2.26 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
    RmaAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp
    time, uint64_t eventPosition, void *userData, OTF2_AttributeList
    *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId,
    OTF2_LockType lockType)
```

Callback for the RmaAcquireLock event record.

An RmaAcquireLock record denotes the time a lock was aquired 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.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.12.2.27 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.
<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*.

J.12.2.28 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-RmaCollectiveBegin)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList)**

Callback for the RmaCollectiveBegin event record.

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

### Parameters

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

J.12.2.29 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-RmaCollectiveEnd)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CollectiveOp collectiveOp, OTF2\_RmaSyncLevel syncLevel, OTF2\_RmaWinRef win, uint32\_t root, uint64\_t bytesSent, uint64\_t bytesReceived)**

Callback for the RmaCollectiveEnd event record.

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

### Parameters

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

**J.12.2.30** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_RmaGet)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)`

Callback for the RmaGet event record.

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

### Parameters

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

```
J.12.2.31 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
RmaGroupSync)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win,
OTF2_GroupRef group)
```

Callback for the RmaGroupSync event record.

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

**Parameters**

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterEvtCallbacks</a> or <a href="#">OTF2_-EvtReader_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.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.12.2.32 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
RmaOpCompleteBlocking)(OTF2_LocationRef location,  
OTF2_TimeStamp time, uint64_t eventPosition, void *userData,  
OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t  
matchingId)
```

Callback for the RmaOpCompleteBlocking event record.

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

### Parameters

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

J.12.2.33 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-RmaOpCompleteNonBlocking)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)**

Callback for the RmaOpCompleteNonBlocking event record.

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

### Parameters

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

J.12.2.34 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-RmaOpCompleteRemote)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)**

Callback for the RmaOpCompleteRemote event record.

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

### Parameters

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

**J.12.2.35** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_RmaOpTest)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

Callback for the RmaOpTest event record.

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

### Parameters

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

**J.12.2.36** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_--RmaPut)(OTF2_LocationRef location, OTF2_TimeStamp time, uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)`

Callback for the RmaPut event record.

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

### Parameters

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

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

```
J.12.2.37 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
RmaReleaseLock)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)
```

Callback for the RmaReleaseLock event record.

An RmaReleaseLock record denotes the time the lock was released.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <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 released, if multiple locks are defined on a window.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.12.2.38 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
RmaRequestLock)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId,
OTF2_LockType lockType)
```

Callback for the RmaRequestLock event record.

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

### Parameters

## APPENDIX J. FILE DOCUMENTATION

---

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

```
J.12.2.39 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_
RmaSync)(OTF2_LocationRef location, OTF2_TimeStamp time,
uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,
OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaSyncType
syncType)
```

Callback for the RmaSync event record.

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

**Parameters**

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

<i>win</i>	ID of the window used for this operation. References a <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](#).

```
J.12.2.40 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_  
          RmaTryLock)(OTF2_LocationRef location, OTF2_TimeStamp time,  
          uint64_t eventPosition, void *userData, OTF2_AttributeList *attributeList,  
          OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType  
          lockType)
```

Callback for the RmaTryLock event record.

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

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <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 aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.12.2.41 **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="#">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.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.12.2.42 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_-RmaWinCreate)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)**

Callback for the RmaWinCreate event record.

An RmaWinCreate record denotes the creation of an RMA window.

### Parameters

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

```
J.12.2.43 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
RmaWinDestroy)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, OTF2_RmaWinRef win)
```

Callback for the RmaWinDestroy event record.

An RmaWinDestroy record denotes the destruction of an RMA window.

### Parameters

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

J.12.2.44 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_ - ThreadAcquireLock)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model, uint32\_t lockID, uint32\_t acquisitionOrder)**

Callback for the ThreadAcquireLock event record.

An ThreadAcquireLock record marks that a thread acquires an lock.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <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 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.12.2.45 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_ - ThreadFork)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model, uint32\_t numberOfRequestedThreads)**

Callback for the ThreadFork event record.

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

### Parameters

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

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.12.2.46** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_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.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.12.2.47 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_ - ThreadReleaseLock)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model, uint32\_t lockID, uint32\_t acquisitionOrder)**

Callback for the ThreadReleaseLock event record.

An ThreadReleaseLock record marks that a thread releases an lock.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>eventPosition</i>	The event position of this event in the trace. Starting with 1.
<i>userData</i>	User data as set by <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 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.12.2.48 **typedef OTF2\_CallbackCode( \* OTF2\_EvtReaderCallback\_ - ThreadTaskComplete)(OTF2\_LocationRef location, OTF2\_TimeStamp time, uint64\_t eventPosition, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam, uint32\_t creatingThread, uint32\_t generationNumber)**

Callback for the ThreadTaskComplete event record.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

### Parameters

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

**J.12.2.49** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
ThreadTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t
generationNumber)`

Callback for the ThreadTaskCreate event record.

An ThreadTaskCreate record marks that a task is/will be created and will be processed by the specified thread team.

### Parameters

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

## APPENDIX J. FILE DOCUMENTATION

---

<i>userData</i>	User data as set by <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. (This is redundant, remove?)
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

**Since**

Version 1.2

**Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.12.2.50 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_
ThreadTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t
generationNumber)
```

Callback for the ThreadTaskSwitch event record.

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

**Parameters**

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

<i>creatingThread</i>	Creating thread of this task.
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.12.2.51 typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
ThreadTeamBegin)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, OTF2_CommRef threadTeam)
```

Callback for the ThreadTeamBegin event record.

### Parameters

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

**J.12.2.52** `typedef OTF2_CallbackCode( * OTF2_EvtReaderCallback_-
ThreadTeamEnd)(OTF2_LocationRef location, OTF2_TimeStamp
time, uint64_t eventPosition, void *userData, OTF2_AttributeList
*attributeList, OTF2_CommRef threadTeam)`

Callback for the ThreadTeamEnd event record.

### Parameters

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

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

Callback for an unknown event record.

### Parameters

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

## **J.12 OTF2\_EvtReaderCallbacks.h File Reference**

---

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

### **J.12.3 Function Documentation**

#### **J.12.3.1 void OTF2\_EvtReaderCallbacks\_Clear ( OTF2\_EvtReaderCallbacks \* evtReaderCallbacks )**

Clears a struct for the event callbacks.

##### **Parameters**

<i>evtReader-</i>	Handle to a struct previously allocated with <a href="#">OTF2_EvtReaderCallbacks_New</a> .
-------------------	--

#### **J.12.3.2 void OTF2\_EvtReaderCallbacks\_Delete ( OTF2\_EvtReaderCallbacks \* evtReaderCallbacks )**

Deallocates a struct for the event callbacks.

##### **Parameters**

<i>evtReader-</i>	Handle to a struct previously allocated with <a href="#">OTF2_EvtReaderCallbacks_New</a> .
-------------------	--

#### **J.12.3.3 OTF2\_EvtReaderCallbacks\* OTF2\_EvtReaderCallbacks\_New ( void )**

Allocates a new struct for the event callbacks.

##### **Returns**

A newly allocated struct of type [OTF2\\_EvtReaderCallbacks](#).

#### **J.12.3.4 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetBufferFlushCallback ( OTF2\_EvtReaderCallbacks \* evtReaderCallbacks, OTF2\_EvtReaderCallback\_BufferFlush bufferFlushCallback )**

Registers the callback for the BufferFlush event.

##### **Parameters**

## **APPENDIX J. FILE DOCUMENTATION**

---

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

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.12.3.5 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetEnterCallback**  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks,`  
`OTF2_EvtReaderCallback_Enter enterCallback` )

Registers the callback for the Enter event.

### **Parameters**

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

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.12.3.6 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetLeaveCallback**  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks,`  
`OTF2_EvtReaderCallback_Leave leaveCallback` )

Registers the callback for the Leave event.

### **Parameters**

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

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

### J.12.3.7 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMeasurementOnOffCallback

```
( OTF2_EvtReaderCallbacks * evtReaderCallbacks, OTF2_-  
    EvtReaderCallback_MeasurementOnOff measurementOnOffCallback  
)
```

Registers the callback for the MeasurementOnOff event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>measure- mentOnOff- Callback</i>	Function which should be called for all MeasurementOnOff events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

### J.12.3.8 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMetricCallback

```
( OTF2_EvtReaderCallbacks * evtReaderCallbacks,  
    OTF2_EvtReaderCallback_Metric metricCallback )
```

Registers the callback for the Metric event.

### Parameters

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

## APPENDIX J. FILE DOCUMENTATION

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.12.3.9 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpiCollectiveBeginCallback**  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks, OTF2_-`  
`EvtReaderCallback_MpiCollectiveBegin mpiCollectiveBeginCallback`  
)

Registers the callback for the MpiCollectiveBegin event.

### Parameters

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

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.12.3.10 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpiCollectiveEndCallback**  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks, OTF2_-`  
`EvtReaderCallback_MpiCollectiveEnd mpiCollectiveEndCallback`  
)

Registers the callback for the MpiCollectiveEnd event.

### Parameters

<code>evtReader- Callbacks</code>	Struct for all callbacks.
<code>mpiCollec- tiveEnd- Callback</code>	Function which should be called for all MpiCollectiveEnd events.

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

#### J.12.3.11 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpiRecvCallback

( `OTF2_EvtReaderCallbacks * evtReaderCallbacks,`  
`OTF2_EvtReaderCallback_MpiRecv mpirecvCallback` )

Registers the callback for the MpiRecv event.

### Parameters

<code>evtReader- Callbacks</code>	Struct for all callbacks.
<code>mpirecv- Callback</code>	Function which should be called for all MpiRecv events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

#### J.12.3.12 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpiRecvRequestCallback

( `OTF2_EvtReaderCallbacks * evtReaderCallbacks,`  
`OTF2_EvtReaderCallback_MpiRecvRequest mpirecvRequestCallback`  
)

Registers the callback for the MpiRecvRequest event.

### Parameters

<code>evtReader- Callbacks</code>	Struct for all callbacks.
<code>mpirecvRe- questCall- back</code>	Function which should be called for all MpiRecvRequest events.

### Returns

**OTF2\_SUCCESS** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.12.3.13 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpilSendCallback**  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks,`  
`OTF2_EvtReaderCallback_MpiIsend mpilSendCallback` )

Registers the callback for the MpiIsend event.

**Parameters**

<code>evtReader- Callbacks</code>	Struct for all callbacks.
<code>mpiIsend- Callback</code>	Function which should be called for all MpiIsend events.

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.12.3.14 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpilSendCompleteCallback**  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks,`  
`OTF2_EvtReaderCallback_MpiIsendComplete  
mpilSendCompleteCallback` )

Registers the callback for the MpiIsendComplete event.

**Parameters**

<code>evtReader- Callbacks</code>	Struct for all callbacks.
<code>mpiIsend- Complete- Callback</code>	Function which should be called for all MpiIsendComplete events.

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

J.12.3.15 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpiRecvCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_MpiRecv mpiRecvCallback** )

Registers the callback for the MpiRecv event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.16 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_-SetMpiRequestCancelledCallback** ( **OTF2\_EvtReaderCallbacks \***  
**evtReaderCallbacks, OTF2\_EvtReaderCallback\_-**  
**MpiRequestCancelled mpiRequestCancelledCallback**  
)

Registers the callback for the MpiRequestCancelled event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.12.3.17 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpiRequestTestCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_MpiRequestTest mpiRequestTestCallback** )

Registers the callback for the MpiRequestTest event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.18 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetMpiSendCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_MpiSend mpiSendCallback** )

Registers the callback for the MpiSend event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

J.12.3.19 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpAcquireLockCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks, OTF2\_-**  
**EvtReaderCallback\_OmpAcquireLock ompAcquireLockCallback**  
)

Registers the callback for the OmpAcquireLock event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.20 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpForkCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_OmpFork ompForkCallback** )

Registers the callback for the OmpFork event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.12.3.21 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpJoinCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_OmpJoin ompJoinCallback** )

Registers the callback for the OmpJoin event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.22 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpReleaseLockCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_OmpReleaseLock ompReleaseLockCallback**  
)

Registers the callback for the OmpReleaseLock event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

J.12.3.23 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpTaskCompleteCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_OmpTaskComplete**  
**ompTaskCompleteCallback** )

Registers the callback for the OmpTaskComplete event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.24 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpTaskCreateCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_OmpTaskCreate** *ompTaskCreateCallback* )

Registers the callback for the OmpTaskCreate event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.12.3.25 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetOmpTaskSwitchCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_OmpTaskSwitch ompTaskSwitchCallback** )

Registers the callback for the OmpTaskSwitch event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.26 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetParameterIntCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_ParameterInt parameterIntCallback** )

Registers the callback for the ParameterInt event.

### Parameters

<i>evtReader- Callbacks</i>	Struct for all callbacks.
<i>parameter- IntCallback</i>	Function which should be called for all ParameterInt events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

J.12.3.27 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetParameterStringCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_ParameterString parameterStringCallback** )

Registers the callback for the ParameterString event.

### Parameters

<i>evtReader- Callbacks</i>	Struct for all callbacks.
<i>parameter- StringCall- back</i>	Function which should be called for all ParameterString events.

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

J.12.3.28 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_-  
SetParameterUnsignedIntCallback** ( **OTF2\_EvtReaderCallbacks**  
\* **evtReaderCallbacks**, **OTF2\_EvtReaderCallback\_-  
ParameterUnsignedInt parameterUnsignedIntCallback**  
)

Registers the callback for the ParameterUnsignedInt event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

## APPENDIX J. FILE DOCUMENTATION

J.12.3.29 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaAcquireLockCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaAcquireLock rmaAcquireLockCallback**  
)

Registers the callback for the RmaAcquireLock event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.30 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaAtomicCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaAtomic rmaAtomicCallback** )

Registers the callback for the RmaAtomic event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

J.12.3.31 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaCollectiveBeginCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaCollectiveBegin**  
**rmaCollectiveBeginCallback** )

Registers the callback for the RmaCollectiveBegin event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.32 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaCollectiveEndCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks, OTF2\_-**  
**EvtReaderCallback\_RmaCollectiveEnd rmaCollectiveEndCallback**  
)

Registers the callback for the RmaCollectiveEnd event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

**J.12.3.33 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaGetCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaGet rmaGetCallback** )

Registers the callback for the RmaGet event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.12.3.34 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaGroupSyncCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaGroupSync rmaGroupSyncCallback** )

Registers the callback for the RmaGroupSync event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

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

Registers the callback for the RmaOpCompleteBlocking event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

```
J.12.3.36 OTF2_ErrorCode OTF2_EvtReaderCallbacks_-
    SetRmaOpCompleteNonBlockingCallback ( OTF2_EvtReaderCallbacks
        * evtReaderCallbacks, OTF2_EvtReaderCallback_-
            RmaOpCompleteNonBlocking rmaOpCompleteNonBlockingCallback
    )
```

Registers the callback for the RmaOpCompleteNonBlocking event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

## APPENDIX J. FILE DOCUMENTATION

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.12.3.37 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaOpCompleteRemoteCallback ( OTF2\_EvtReaderCallbacks \* evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaOpCompleteRemote rmaOpCompleteRemoteCallback )**

Registers the callback for the RmaOpCompleteRemote event.

### Parameters

<code>evtReader-Callbacks</code>	Struct for all callbacks.
<code>rmaOp-CompleteR-remoteCall-back</code>	Function which should be called for all RmaOpCompleteRemote events.

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.12.3.38 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaOpTestCallback ( OTF2\_EvtReaderCallbacks \* evtReaderCallbacks, OTF2\_EvtReaderCallback\_RmaOpTest rmaOpTestCallback )**

Registers the callback for the RmaOpTest event.

### Parameters

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

### Returns

***OTF2\_SUCCESS*** if successful

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.12.3.39 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaPutCallback**  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks,`  
`OTF2_EvtReaderCallback_RmaPut rmaPutCallback` )

Registers the callback for the RmaPut event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.12.3.40 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaReleaseLockCallback**  
( `OTF2_EvtReaderCallbacks * evtReaderCallbacks,`  
`OTF2_EvtReaderCallback_RmaReleaseLock rmaReleaseLockCallback`  
)

Registers the callback for the RmaReleaseLock event.

### Parameters

<code>evtReader- Callbacks</code>	Struct for all callbacks.
<code>rmaRe- leaseLock- Callback</code>	Function which should be called for all RmaReleaseLock events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.12.3.41 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaRequestLockCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaRequestLock rmaRequestLockCallback**  
)

Registers the callback for the RmaRequestLock event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.42 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaSyncCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaSync rmaSyncCallback** )

Registers the callback for the RmaSync event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

J.12.3.43 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaTryLockCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaTryLock rmaTryLockCallback** )

Registers the callback for the RmaTryLock event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

J.12.3.44 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaWaitChangeCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaWaitChange rmaWaitChangeCallback** )

Registers the callback for the RmaWaitChange event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

## APPENDIX J. FILE DOCUMENTATION

J.12.3.45 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaWinCreateCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaWinCreate rmaWinCreateCallback** )

Registers the callback for the RmaWinCreate event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

J.12.3.46 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetRmaWinDestroyCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_RmaWinDestroy rmaWinDestroyCallback** )

Registers the callback for the RmaWinDestroy event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

**J.12.3.47 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadAcquireLockCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_ThreadAcquireLock**  
**threadAcquireLockCallback** )

Registers the callback for the ThreadAcquireLock event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.12.3.48 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadForkCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_ThreadFork** *threadForkCallback* )

Registers the callback for the ThreadFork event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.12.3.49 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadJoinCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_ThreadJoin threadJoinCallback** )

Registers the callback for the ThreadJoin event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.12.3.50 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadReleaseLockCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_ThreadReleaseLock**  
**threadReleaseLockCallback** )

Registers the callback for the ThreadReleaseLock event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.12 OTF2\_EvtReaderCallbacks.h File Reference

---

**J.12.3.51 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_-  
SetThreadTaskCompleteCallback ( OTF2\_EvtReaderCallbacks \*  
evtReaderCallbacks, OTF2\_EvtReaderCallback\_ThreadTaskComplete  
threadTaskCompleteCallback )**

Registers the callback for the ThreadTaskComplete event.

### Parameters

<i>evtReader- Callbacks</i>	Struct for all callbacks.
<i>thread- TaskCom- pleteCall- back</i>	Function which should be called for all ThreadTaskComplete events.

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

**J.12.3.52 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadTaskCreateCallback  
( OTF2\_EvtReaderCallbacks \* evtReaderCallbacks, OTF2\_-  
EvtReaderCallback\_ThreadTaskCreate threadTaskCreateCallback  
)**

Registers the callback for the ThreadTaskCreate event.

### Parameters

<i>evtReader- Callbacks</i>	Struct for all callbacks.
<i>thread- TaskCreate- Callback</i>	Function which should be called for all ThreadTaskCreate events.

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

## APPENDIX J. FILE DOCUMENTATION

J.12.3.53 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadTaskSwitchCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks, OTF2\_-**  
**EvtReaderCallback\_ThreadTaskSwitch threadTaskSwitchCallback**  
)

Registers the callback for the ThreadTaskSwitch event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

J.12.3.54 **OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadTeamBeginCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks, OTF2\_-**  
**EvtReaderCallback\_ThreadTeamBegin threadTeamBeginCallback**  
)

Registers the callback for the ThreadTeamBegin event.

### Parameters

<i>evtReader-Callbacks</i>	Struct for all callbacks.
<i>thread-TeamBegin-Callback</i>	Function which should be called for all ThreadTeamBegin events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

## J.13 OTF2\_EvtWriter.h File Reference

---

**J.12.3.55 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetThreadTeamEndCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_ThreadTeamEnd threadTeamEndCallback** )

Registers the callback for the ThreadTeamEnd event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.12.3.56 OTF2\_ErrorCode OTF2\_EvtReaderCallbacks\_SetUnknownCallback**  
( **OTF2\_EvtReaderCallbacks \* evtReaderCallbacks,**  
**OTF2\_EvtReaderCallback\_Unknown unknownCallback** )

Registers the callback for the Unknown event.

### Parameters

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

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.13 OTF2\_EvtWriter.h File Reference

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

## **APPENDIX J. FILE DOCUMENTATION**

---

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

### **TypeDefs**

- **typedef struct OTF2\_EvtWriter\_struct OTF2\_EvtWriter**

*Keeps all necessary information about the event writer. See OTF2\_EvtWriter\_struct for detailed information.*

### **Functions**

- **OTF2\_ErrorCode OTF2\_EvtWriter\_BufferFlush (OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp time, OTF2\_TimeStamp stopTime)**

*Records an BufferFlush event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_ClearRewindPoint (OTF2\_EvtWriter \*writer, uint32\_t rewindId)**

*Please give me a documantation.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_Enter (OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp time, OTF2\_RegionRef region)**

*Records an Enter event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_GetLocationID (const OTF2\_EvtWriter \*writer, OTF2\_LocationRef \*locationID)**

*Function to get the location ID of a writer object.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_GetNumberOfEvents (OTF2\_EvtWriter \*writer, uint64\_t \*numberOfEvents)**

*Get the number of events.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_GetUserData (const OTF2\_EvtWriter \*writer, void \*\*userData)**

*Function to get the location of a writer object.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_Leave (OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp time, OTF2\_RegionRef region)**

*Records an Leave event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_MeasurementOnOff (OTF2\_EvtWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp time, OTF2\_MeasurementMode measurementMode)**

## J.13 OTF2\_EvtWriter.h File Reference

---

*Records an MeasurementOnOff event.*

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

*Records an Metric event.*

- `OTF2_ErrorCode OTF2_EvtWriter_MpiCollectiveBegin (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time)`

*Records an MpiCollectiveBegin event.*

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

*Records an MpiCollectiveEnd event.*

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

*Records an MpiIrecv event.*

- `OTF2_ErrorCode OTF2_EvtWriter_MpiIrecvRequest (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint64_t requestID)`

*Records an MpiIrecvRequest event.*

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

*Records an MpiIsend event.*

- `OTF2_ErrorCode OTF2_EvtWriter_MpiIsendComplete (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint64_t requestID)`

*Records an MpiIsendComplete event.*

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

*Records an MpiRecv event.*

- `OTF2_ErrorCode OTF2_EvtWriter_MpiRequestCancelled (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint64_t requestID)`

*Records an MpiRequestCancelled event.*

---

## APPENDIX J. FILE DOCUMENTATION

- `OTF2_ErrorCode OTF2_EvtWriter_MpiRequestTest (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint64_t requestID)`

*Records an MpiRequestTest event.*
- `OTF2_ErrorCode OTF2_EvtWriter_MpiSend (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

*Records an MpiSend event.*
- `OTF2_ErrorCode OTF2_EvtWriter_OmpAcquireLock (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint32_t lockID, uint32_t acquisitionOrder)`

*Records an OmpAcquireLock event.*
- `OTF2_ErrorCode OTF2_EvtWriter_OmpFork (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint32_t numberOfRequestedThreads)`

*Records an OmpFork event.*
- `OTF2_ErrorCode OTF2_EvtWriter_OmpJoin (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time)`

*Records an OmpJoin event.*
- `OTF2_ErrorCode OTF2_EvtWriter_OmpReleaseLock (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint32_t lockID, uint32_t acquisitionOrder)`

*Records an OmpReleaseLock event.*
- `OTF2_ErrorCode OTF2_EvtWriter_OmpTaskComplete (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint64_t taskID)`

*Records an OmpTaskComplete event.*
- `OTF2_ErrorCode OTF2_EvtWriter_OmpTaskCreate (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint64_t taskID)`

*Records an OmpTaskCreate event.*
- `OTF2_ErrorCode OTF2_EvtWriter_OmpTaskSwitch (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, uint64_t taskID)`

*Records an OmpTaskSwitch event.*
- `OTF2_ErrorCode OTF2_EvtWriter_ParameterInt (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_ParameterRef parameter, int64_t value)`

*Records an ParameterInt event.*

## J.13 OTF2\_EvtWriter.h File Reference

---

- `OTF2_ErrorCode OTF2_EvtWriter_ParameterString (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_ParameterRef parameter, OTF2_StringRef string)`

*Records an ParameterString event.*
- `OTF2_ErrorCode OTF2_EvtWriter_ParameterUnsignedInt (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_ParameterRef parameter, uint64_t value)`

*Records an ParameterUnsignedInt event.*
- `OTF2_ErrorCode OTF2_EvtWriter_Rewind (OTF2_EvtWriter *writer, uint32_t rewindId)`

*Please give me a documentation.*
- `OTF2_ErrorCode OTF2_EvtWriter_RmaAcquireLock (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)`

*Records an RmaAcquireLock event.*
- `OTF2_ErrorCode OTF2_EvtWriter_RmaAtomic (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)`

*Records an RmaAtomic event.*
- `OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveBegin (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time)`

*Records an RmaCollectiveBegin event.*
- `OTF2_ErrorCode OTF2_EvtWriter_RmaCollectiveEnd (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)`

*Records an RmaCollectiveEnd event.*
- `OTF2_ErrorCode OTF2_EvtWriter_RmaGet (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)`

*Records an RmaGet event.*
- `OTF2_ErrorCode OTF2_EvtWriter_RmaGroupSync (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)`

*Records an RmaGroupSync event.*
- `OTF2_ErrorCode OTF2_EvtWriter_RmaOpCompleteBlocking (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win, uint64_t matchingId)`

*Records an RmaOpCompleteBlocking event.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaOpCompleteNonBlocking** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win, **uint64\_t** matchingId)  
*Records an RmaOpCompleteNonBlocking event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaOpCompleteRemote** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win, **uint64\_t** matchingId)  
*Records an RmaOpCompleteRemote event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaOpTest** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win, **uint64\_t** matchingId)  
*Records an RmaOpTest event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaPut** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win, **uint32\_t** remote, **uint64\_t** bytes, **uint64\_t** matchingId)  
*Records an RmaPut event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaReleaseLock** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win, **uint32\_t** remote, **uint64\_t** lockId)  
*Records an RmaReleaseLock event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaRequestLock** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win, **uint32\_t** remote, **uint64\_t** lockId, **OTF2\_LockType** lockType)  
*Records an RmaRequestLock event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaSync** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win, **uint32\_t** remote, **OTF2\_RmaSyncType** syncType)  
*Records an RmaSync event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaTryLock** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win, **uint32\_t** remote, **uint64\_t** lockId, **OTF2\_LockType** lockType)  
*Records an RmaTryLock event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaWaitChange** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win)  
*Records an RmaWaitChange event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_RmaWinCreate** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_RmaWinRef** win)  
*Records an RmaWinCreate event.*

## J.13 OTF2\_EvtWriter.h File Reference

---

- `OTF2_ErrorCode OTF2_EvtWriter_RmaWinDestroy (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_RmaWinRef win)`

*Records an RmaWinDestroy event.*

- `OTF2_ErrorCode OTF2_EvtWriter_SetLocationID (OTF2_EvtWriter *writer, OTF2_LocationRef location)`

*The location ID is not always known on measurement 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 documentation.*

- `OTF2_ErrorCode OTF2_EvtWriter_ThreadAcquireLock (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

*Records an ThreadAcquireLock event.*

- `OTF2_ErrorCode OTF2_EvtWriter_ThreadFork (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)`

*Records an ThreadFork event.*

- `OTF2_ErrorCode OTF2_EvtWriter_ThreadJoin (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_Paradigm model)`

*Records an ThreadJoin event.*

- `OTF2_ErrorCode OTF2_EvtWriter_ThreadReleaseLock (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_Paradigm model, uint32_t lockID, uint32_t acquisitionOrder)`

*Records an ThreadReleaseLock event.*

- `OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskComplete (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Records an ThreadTaskComplete event.*

- `OTF2_ErrorCode OTF2_EvtWriter_ThreadTaskCreate (OTF2_EvtWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp time, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t generationNumber)`

*Records an ThreadTaskCreate event.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadTaskSwitch** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_CommRef** threadTeam, **uint32\_t** creatingThread, **uint32\_t** generationNumber)  
*Records an ThreadTaskSwitch event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadTeamBegin** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_CommRef** threadTeam)  
*Records an ThreadTeamBegin event.*
- **OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadTeamEnd** (**OTF2\_EvtWriter** \*writer, **OTF2\_AttributeList** \*attributeList, **OTF2\_TimeStamp** time, **OTF2\_CommRef** threadTeam)  
*Records an ThreadTeamEnd event.*

### **J.13.1 Detailed Description**

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

#### **Source Template:**

*templates/OTF2\_EvtWriter.tpl.h*

#### **Maintainer:**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

#### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### **J.13.2 Function Documentation**

#### **J.13.2.1 OTF2\_ErrorCode OTF2\_EvtWriter\_BufferFlush ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_TimeStamp stopTime )**

Records an BufferFlush event.

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

#### **Parameters**

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

## J.13 OTF2\_EvtWriter.h File Reference

---

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>stopTime</i>	The time the buffer flush finished.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.13.2.2 OTF2\_ErrorCode OTF2\_EvtWriter\_ClearRewindPoint ( OTF2\_EvtWriter \* *writer*, uint32\_t *rewindId* )

Please give me a documentation.

### Parameters

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

### Since

Version 1.1

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.13.2.3 OTF2\_ErrorCode OTF2\_EvtWriter\_Enter ( OTF2\_EvtWriter \* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*, OTF2\_RegionRef *region* )

Records an Enter event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>region</i>	Needs to be defined in a definition record References a <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.

**J.13.2.4 OTF2\_ErrorCode OTF2\_EvtWriter\_GetLocationID ( const OTF2\_EvtWriter \* writer, OTF2\_LocationRef \* locationID )**

Function to get the location ID of a writer object.

### **Parameters**

<i>writer</i>	Writer object which has to be deleted
<i>locationID</i>	Pointer to a variable where the ID is returned in

### **Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.13.2.5 OTF2\_ErrorCode OTF2\_EvtWriter\_GetNumberOfEvents ( OTF2\_EvtWriter \* writer, uint64\_t \* numberOfEvents )**

Get the number of events.

Get the number of events written with this event writer. You should call this function right before closing the event writer to get the correct number of stored event records.

### **Parameters**

	<i>writer</i>	Writer object.
<i>out</i>	<i>numberO-</i> <i>fEvents</i>	Return pointer to the number of events.

### **Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## J.13 OTF2\_EvtWriter.h File Reference

---

**J.13.2.6 OTF2\_ErrorCode OTF2\_EvtWriter\_GetUserData ( const OTF2\_EvtWriter \* *writer*, void \*\* *userData* )**

Function to get the location of a writer object.

### Parameters

	<i>writer</i>	Writer object.
out	<i>userData</i>	Pointer to a variable where the pointer to the location is returned in.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.7 OTF2\_ErrorCode OTF2\_EvtWriter\_Leave ( OTF2\_EvtWriter \* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*, OTF2\_RegionRef *region* )**

Records an Leave event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>region</i>	Needs to be defined in a definition record References a <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.

**J.13.2.8 OTF2\_ErrorCode OTF2\_EvtWriter\_MeasurementOnOff ( OTF2\_EvtWriter \* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*, OTF2\_MeasurementMode *measurementMode* )**

Records an MeasurementOnOff event.

## **APPENDIX J. FILE DOCUMENTATION**

---

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>measurementMode</i>	Is the measurement turned on ( <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.

**J.13.2.9 OTF2\_ErrorCode OTF2\_EvtWriter\_Metric ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_MetricRef metric, uint8\_t numberofMetrics, const OTF2\_Type \*  
typeIDs, const OTF2\_MetricValue \* metricValues )**

Records an Metric event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>metric</i>	Could be a metric class or a metric instance. References a <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.
<i>metricValues</i>	List of metric values.

## J.13 OTF2\_EvtWriter.h File Reference

---

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.13.2.10 OTF2\_ErrorCode OTF2\_EvtWriter\_MpiCollectiveBegin ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time )

Records an MpiCollectiveBegin event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.13.2.11 OTF2\_ErrorCode OTF2\_EvtWriter\_MpiCollectiveEnd ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_CollectiveOp collectiveOp, OTF2\_CommRef communicator, uint32\_t root, uint64\_t sizeSent, uint64\_t sizeReceived )

Records an MpiCollectiveEnd event.

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

### Parameters

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

## APPENDIX J. FILE DOCUMENTATION

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>communicator</i>	Communicator References a <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 <i>communicator</i> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.13.2.12 OTF2\_ErrorCode OTF2\_EvtWriter\_MpiRecv ( 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 MpiRecv event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>sender</i>	MPI rank of sender in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <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

## J.13 OTF2\_EvtWriter.h File Reference

---

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.13 OTF2\_ErrorCode OTF2\_EvtWriter\_MpiRecvRequest ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
uint64\_t requestID )**

Records an MpiRecvRequest event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the requested receive

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.14 OTF2\_ErrorCode OTF2\_EvtWriter\_MpiSend ( 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 MpiSend event.

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

### Parameters

---

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>receiver</i>	MPI rank of receiver in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <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.

**J.13.2.15 OTF2\_ErrorCode OTF2\_EvtWriter\_MpiSendComplete ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, uint64\_t requestID )**

Records an MpiSendComplete event.

Signals the completion of non-blocking send request.

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

### **Since**

Version 1.0

### **Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

## J.13 OTF2\_EvtWriter.h File Reference

---

**J.13.2.16 OTF2\_ErrorCode OTF2\_EvtWriter.MpiRecv ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
uint32\_t sender, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t  
msgLength )**

Records an MpiRecv event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>sender</i>	MPI rank of sender in <i>communicator</i> .
<i>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.

**J.13.2.17 OTF2\_ErrorCode OTF2\_EvtWriter.MpiRequestCancelled (**  
**OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp time, uint64\_t requestID )**

Records an MpiRequestCancelled event.

This events appears if the program canceled a request.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.18 OTF2\_ErrorCode OTF2\_EvtWriter\_MpiRequestTest ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
uint64\_t requestID )**

Records an MpiRequestTest event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.19 OTF2\_ErrorCode OTF2\_EvtWriter\_MpiSend ( OTF2\_EvtWriter \* writer,  
OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, uint32\_t  
receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t  
msgLength )**

Records an MpiSend event.

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

### Parameters

## J.13 OTF2\_EvtWriter.h File Reference

---

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>receiver</i>	MPI rank of receiver in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <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.

**J.13.2.20 OTF2\_ErrorCode OTF2\_EvtWriter\_OmpAcquireLock ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
uint32\_t lockID, uint32\_t acquisitionOrder )**

Records an OmpAcquireLock event.

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

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.21 OTF2\_ErrorCode OTF2\_EvtWriter\_OmpFork ( OTF2\_EvtWriter \* writer,  
OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, uint32\_t  
numberOfRequestedThreads )**

Records an OmpFork event.

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

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>numberOfRequestedThreads</i>	Requested size of the team.

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.22 OTF2\_ErrorCode OTF2\_EvtWriter\_OmpJoin ( OTF2\_EvtWriter \* writer,  
OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time )**

Records an OmpJoin event.

## J.13 OTF2\_EvtWriter.h File Reference

---

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

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.13.2.23 OTF2\_ErrorCode OTF2\_EvtWriter\_OmpReleaseLock ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
uint32\_t lockID, uint32\_t acquisitionOrder )**

Records an OmpReleaseLock event.

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

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.0

### **Deprecated**

In version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.24 OTF2\_ErrorCode OTF2\_EvtWriter\_OmpTaskComplete ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, uint64\_t taskID )**

Records an OmpTaskComplete event.

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

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	Identifier of the completed task instance.

### **Since**

Version 1.0

### **Deprecated**

In version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.13 OTF2\_EvtWriter.h File Reference

---

**J.13.2.25 OTF2\_ErrorCode OTF2\_EvtWriter\_OmpTaskCreate ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
uint64\_t taskID )**

Records an OmpTaskCreate event.

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

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	Identifier of the newly created task instance.

### Since

Version 1.0

### Deprecated

In version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.13.2.26 OTF2\_ErrorCode OTF2\_EvtWriter\_OmpTaskSwitch ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
uint64\_t taskID )**

Records an OmpTaskSwitch event.

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

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

### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>taskID</i>	Identifier of the now active task instance.

### **Since**

Version 1.0

### **Deprecated**

In version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.27 OTF2\_ErrorCode OTF2\_EvtWriter\_ParameterInt ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_ParameterRef parameter, int64\_t value )**

Records an ParameterInt event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>parameter</i>	Parameter ID. References a <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.

## J.13 OTF2\_EvtWriter.h File Reference

---

**J.13.2.28 OTF2\_ErrorCode OTF2\_EvtWriter\_ParameterString ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_ParameterRef parameter, OTF2\_StringRef string )**

Records an ParameterString event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>parameter</i>	Parameter ID. References a <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.

**J.13.2.29 OTF2\_ErrorCode OTF2\_EvtWriter\_ParameterUnsignedInt (**  
**OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp time, OTF2\_ParameterRef parameter, uint64\_t value**  
**)**

Records an ParameterUnsignedInt event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>parameter</i>	Parameter ID. References a <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.

**J.13.2.30 OTF2\_ErrorCode OTF2\_EvtWriter\_Rewind ( OTF2\_EvtWriter \* writer,  
                  uint32\_t rewindId )**

Please give me a documentation.

### **Parameters**

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

### **Since**

Version 1.1

### **Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.13.2.31 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaAcquireLock ( OTF2\_EvtWriter  
                  \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp  
                  time, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId,  
                  OTF2\_LockType lockType )**

Records an RmaAcquireLock event.

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

### **Parameters**

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

## J.13 OTF2\_EvtWriter.h File Reference

---

<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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 aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.13.2.32 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

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.33 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaCollectiveBegin (**  
**OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp time )**

Records an RmaCollectiveBegin event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.34 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaCollectiveEnd ( OTF2\_EvtWriter**  
**\* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,**  
**OTF2\_CollectiveOp collectiveOp, OTF2\_RmaSyncLevel syncLevel,**  
**OTF2\_RmaWinRef win, uint32\_t root, uint64\_t bytesSent, uint64\_t**  
**bytesReceived )**

Records an RmaCollectiveEnd event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>collectiveOp</i>	Determines which collective operation it is.
<i>syncLevel</i>	Synchronization level of this collective operation.

## J.13 OTF2\_EvtWriter.h File Reference

---

<i>win</i>	ID of the window used for this operation. References a <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.

**J.13.2.35 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaGet ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t bytes, uint64\_t matchingId )**

Records an RmaGet event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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.

**J.13.2.36 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaGroupSync ( OTF2\_EvtWriter  
\* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp  
*time*, OTF2\_RmaSyncLevel *syncLevel*, OTF2\_RmaWinRef *win*,  
OTF2\_GroupRef *group* )**

Records an RmaGroupSync event.

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

#### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	ID of the window used for this operation. References a <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.

**J.13.2.37 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaOpCompleteBlocking ( OTF2\_EvtWriter \* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*, OTF2\_RmaWinRef *win*, uint64\_t *matchingId* )**

Records an RmaOpCompleteBlocking event.

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

#### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

## J.13 OTF2\_EvtWriter.h File Reference

---

<i>win</i>	ID of the window used for this operation. References a <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.

**J.13.2.38 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaOpCompleteNonBlocking (**  
**OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp time, OTF2\_RmaWinRef win, uint64\_t matchingId )**

Records an RmaOpCompleteNonBlocking event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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.

## **APPENDIX J. FILE DOCUMENTATION**

---

**J.13.2.39 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaOpCompleteRemote (**  
**OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp time, OTF2\_RmaWinRef win, uint64\_t matchingId )**

Records an RmaOpCompleteRemote event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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.

**J.13.2.40 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaOpTest ( OTF2\_EvtWriter \***  
**writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,**  
**OTF2\_RmaWinRef win, uint64\_t matchingId )**

Records an RmaOpTest event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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.

## J.13 OTF2\_EvtWriter.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

```
J.13.2.41 OTF2_ErrorCode OTF2_EvtWriter_RmaPut ( OTF2_EvtWriter *
    writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,
    OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId
)
```

Records an RmaPut event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

```
J.13.2.42 OTF2_ErrorCode OTF2_EvtWriter_RmaReleaseLock ( OTF2_EvtWriter *
    writer, OTF2_AttributeList * attributeList, OTF2_TimeStamp time,
    OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId )
```

Records an RmaReleaseLock event.

An RmaReleaseLock record denotes the time the lock was released.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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.

**J.13.2.43 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaRequestLock ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType )**

Records an RmaRequestLock event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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 aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

### **Since**

Version 1.2

## J.13 OTF2\_EvtWriter.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.44 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaSync ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_RmaWinRef win, uint32\_t remote, OTF2\_RmaSyncType  
syncType )**

Records an RmaSync event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.45 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaTryLock ( OTF2\_EvtWriter  
\* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp  
time, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId,  
OTF2\_LockType lockType )**

Records an RmaTryLock event.

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

### Parameters

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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 aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

### **Since**

Version 1.2

### **Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.13.2.46 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaWaitChange ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_RmaWinRef win )**

Records an RmaWaitChange event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window used for this operation. References a <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.

## J.13 OTF2\_EvtWriter.h File Reference

---

**J.13.2.47 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaWinCreate ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_RmaWinRef win )**

Records an RmaWinCreate event.

An RmaWinCreate record denotes the creation of an RMA window.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window created. References a <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.

**J.13.2.48 OTF2\_ErrorCode OTF2\_EvtWriter\_RmaWinDestroy ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_RmaWinRef win )**

Records an RmaWinDestroy event.

An RmaWinDestroy record denotes the destruction of an RMA window.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>win</i>	ID of the window 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

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.13.2.49 OTF2\_ErrorCode OTF2\_EvtWriter\_SetLocationID ( OTF2\_EvtWriter \* writer, OTF2\_LocationRef location )**

The location ID is not always known on measurement start, and only needed on the first buffer flush to generate the file name. This function enables setting of the location ID after generating the buffer object.

### **Parameters**

<i>writer</i>	Writer object.
<i>location</i>	Location ID.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.13.2.50 OTF2\_ErrorCode OTF2\_EvtWriter\_SetUserData ( OTF2\_EvtWriter \* writer, void \* userData )**

Function to set user defined data to a writer object.

### **Parameters**

<i>writer</i>	Writer object.
<i>userData</i>	User provided data. Can be queried with <a href="#">OTF2_EvtWriter_GetUserData</a> .

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.13.2.51 OTF2\_ErrorCode OTF2\_EvtWriter\_StoreRewindPoint ( OTF2\_EvtWriter \* writer, uint32\_t rewindId )**

Please give me a documentation.

### **Parameters**

<i>writer</i>	Writer object.
<i>rewindId</i>	Generic attributes for the event.

## J.13 OTF2\_EvtWriter.h File Reference

---

### Since

Version 1.1

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.52 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadAcquireLock ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_Paradigm model, uint32\_t lockID, uint32\_t acquisitionOrder )**

Records an ThreadAcquireLock event.

An ThreadAcquireLock record marks that a thread acquires an lock.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.53 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadFork ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_Paradigm model, uint32\_t numberOfRequestedThreads )**

Records an ThreadFork event.

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

### Parameters

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

## **APPENDIX J. FILE DOCUMENTATION**

---

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

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.54 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadJoin ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*,  
OTF2\_Paradigm *model* )**

Records an ThreadJoin event.

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

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.55 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadReleaseLock ( OTF2\_EvtWriter  
\* *writer*, OTF2\_AttributeList \* *attributeList*, OTF2\_TimeStamp *time*,  
OTF2\_Paradigm *model*, uint32\_t *lockID*, uint32\_t *acquisitionOrder* )**

Records an ThreadReleaseLock event.

## J.13 OTF2\_EvtWriter.h File Reference

---

An ThreadReleaseLock record marks that a thread releases an lock.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

### Returns

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**J.13.2.56 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadTaskComplete (**  
**OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp time, OTF2\_CommRef threadTeam, uint32\_t**  
**creatingThread, uint32\_t generationNumber )**

Records an ThreadTaskComplete event.

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <a href="#">Comm</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.

**J.13.2.57 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadTaskCreate ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_CommRef threadTeam, uint32\_t creatingThread, uint32\_t generationNumber )**

Records an ThreadTaskCreate event.

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

**Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <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. (This is redundant, remove?)
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.13.2.58 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadTaskSwitch ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_CommRef threadTeam, uint32\_t creatingThread, uint32\_t generationNumber )**

Records an ThreadTaskSwitch event.

## J.13 OTF2\_EvtWriter.h File Reference

---

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

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <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.

**J.13.2.59 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadTeamBegin ( OTF2\_EvtWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time, OTF2\_CommRef threadTeam )**

Records an ThreadTeamBegin event.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <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.

**J.13.2.60 OTF2\_ErrorCode OTF2\_EvtWriter\_ThreadTeamEnd ( OTF2\_EvtWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp time,  
OTF2\_CommRef threadTeam )**

Records an ThreadTeamEnd event.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the event.
<i>time</i>	The time for this event.
<i>threadTeam</i>	Thread team References a <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.

## J.14 OTF2\_GeneralDefinitions.h File Reference

This header file provides general definitions which should be accessible in all internal and external modules.

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

### Defines

- #define [OTF2\\_CHUNK\\_SIZE\\_MAX](#) ( uint64\_t )( 1024 \* 1024 \* 16 )  
*Defines the maximum size of a chunk.*
- #define [OTF2\\_CHUNK\\_SIZE\\_MIN](#) ( uint64\_t )( 256 \* 1024 )  
*Defines the minimum size of a chunk.*
- #define [OTF2\\_UNDEFINED\\_ATTRIBUTE](#)( ( OTF2\_AttributeRef )OTF2\_-UNDEFINED\_UINT32 )

## J.14 OTF2\_GeneralDefinitions.h File Reference

---

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

- #define OTF2\_UNDEFINED\_CALLPATH (( OTF2\_CallpathRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_CALLSITE (( OTF2\_CallsiteRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_COMM(( OTF2\_CommRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_GROUP(( OTF2\_GroupRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_LOCATION (( OTF2\_LocationRef )OTF2\_UNDEFINED\_UINT64 )

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

- #define OTF2\_UNDEFINED\_LOCATION\_GROUP(( OTF2\_LocationGroupRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_METRIC(( OTF2\_MetricRef )OTF2\_UNDEFINED\_UINT32 )

*The invalid value for a reference to a [MetricClass](#), or a [MetricInstance](#) definition.*

- #define OTF2\_UNDEFINED\_METRIC\_MEMBER(( OTF2\_MetricMemberRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_PARAMETER(( OTF2\_ParameterRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_REGION(( OTF2\_RegionRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_RMA\_WIN (( OTF2\_RmaWinRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_STRING(( OTF2\_StringRef )OTF2\_UNDEFINED\_UINT32 )

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

- #define OTF2\_UNDEFINED\_SYSTEM\_TREE\_NODE(( OTF2\_SystemTreeNodeRef )OTF2\_UNDEFINED\_UINT32 )

## **APPENDIX J. FILE DOCUMENTATION**

---

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

- `#define OTF2_UNDEFINED_TYPE OTF2_UNDEFINED_UINT8`

### **OTF2 library version.**

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

### **Standard undefined values for basic data types.**

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

## **Typedefs**

- `typedef uint32_t OTF2_AttributeRef`  
*Type used to indicate a reference to a [Attribute](#) definition.*
- `typedef uint32_t OTF2_CallpathRef`  
*Type used to indicate a reference to a [Callpath](#) definition.*
- `typedef uint32_t OTF2_CallsiteRef`  
*Type used to indicate a reference to a [Callsite](#) definition.*
- `typedef uint32_t OTF2_CommRef`  
*Type used to indicate a reference to a [Comm](#) definition.*
- `typedef uint8_t OTF2_Compression`  
*Defines which compression is used. Please see [OTF2\\_Compression\\_enum](#) for a detailed description.*
- `typedef struct OTF2_DefReader_struct OTF2_DefReader`  
*OTF2 local definition reader handle.*
- `typedef struct OTF2_EvtReader_struct OTF2_EvtReader`  
*OTF2 local event reader handle.*
- `typedef uint8_t OTF2_FileMode`  
*Defines how to interact with files. Please see [OTF2\\_FileMode\\_enum](#) for a detailed description.*

## J.14 OTF2\_GeneralDefinitions.h File Reference

---

- **typedef uint8\_t OTF2\_FileSubstrate**  
*Defines which file substrate is used. Please see [OTF2\\_FileSubstrate\\_enum](#) for a detailed description.*
- **typedef uint8\_t OTF2\_FileType**  
*Defines which file type is used. Please see [OTF2\\_FileType\\_enum](#) for a detailed description.*
- **typedef uint8\_t OTF2\_FlushType**  
*Defines whether the recorded data is flushed to a file or not. Please see [OTF2\\_FlushType\\_enum](#) for a detailed description.*
- **typedef struct OTF2\_GlobalDefReader\_struct OTF2\_GlobalDefReader**  
*OTF2 global definition reader handle.*
- **typedef struct OTF2\_GlobalEvtReader\_struct OTF2\_GlobalEvtReader**  
*OTF2 global event reader handle.*
- **typedef struct OTF2\_GlobalSnapReader\_struct OTF2\_GlobalSnapReader**  
*OTF2 global snap reader handle.*
- **typedef uint32\_t OTF2\_GroupRef**  
*Type used to indicate a reference to a [Group](#) definition.*
- **typedef uint32\_t OTF2\_LocationGroupRef**  
*Type used to indicate a reference to a [LocationGroup](#) definition.*
- **typedef uint64\_t OTF2\_LocationRef**  
*Type used to indicate a reference to a [Location](#) definition.*
- **typedef uint8\_t OTF2\_MappingType**  
*Wrapper for enum [OTF2\\_MappingType\\_enum](#).*
- **typedef struct OTF2\_MarkerReader\_struct OTF2\_MarkerReader**  
*OTF2 marker reader handle.*
- **typedef uint32\_t OTF2\_MetricMemberRef**  
*Type used to indicate a reference to a [MetricMember](#) definition.*
- **typedef uint32\_t OTF2\_MetricRef**  
*Type used to indicate a reference to a [MetricClass](#), or a [MetricInstance](#) definition.*
- **typedef uint8\_t OTF2\_Paradigm**  
*Wrapper for enum [OTF2\\_Paradigm\\_enum](#).*
- **typedef uint32\_t OTF2\_ParameterRef**  
*Type used to indicate a reference to a [Parameter](#) definition.*
- **typedef uint32\_t OTF2\_RegionRef**  
*Type used to indicate a reference to a [Region](#) definition.*
- **typedef uint32\_t OTF2\_RmaWinRef**  
*Type used to indicate a reference to a [RmaWin](#) definition.*
- **typedef struct OTF2\_SnapReader\_struct OTF2\_SnapReader**

*OTF2 local snap reader handle.*

- **typedef uint32\_t OTF2\_StringRef**

*Type used to indicate a reference to a [String](#) definition.*

- **typedef uint32\_t OTF2\_SystemTreeNodeRef**

*Type used to indicate a reference to a [SystemTreeNode](#) definition.*

- **typedef uint8\_t OTF2\_ThumbnailType**

*Wrapper for enum [OTF2\\_ThumbnailType\\_enum](#).*

- **typedef uint64\_t OTF2\_TimeStamp**

*OTF2 time stamp.*

- **typedef uint8\_t OTF2\_Type**

*Wrapper for enum [OTF2\\_Type\\_enum](#).*

### Enumerations

- **enum OTF2\_CallbackCode {**

**OTF2\_CALLBACK\_SUCCESS** = 0,

**OTF2\_CALLBACK\_INTERRUPT** = !**OTF2\_CALLBACK\_SUCCESS** }

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

- **enum OTF2\_Compression\_enum {**

**OTF2\_COMPRESSION\_UNDEFINED** = 0,

**OTF2\_COMPRESSION\_NONE** = 1,

**OTF2\_COMPRESSION\_ZLIB** = 2 }

*Defines which compression is used.*

- **enum OTF2\_FileMode\_enum {**

**OTF2\_FILEMODE\_WRITE** = 0,

**OTF2\_FILEMODE\_READ** = 1,

**OTF2\_FILEMODE MODIFY** = 2 }

*Defines how to interact with files.*

- **enum OTF2\_FileSubstrate\_enum {**

**OTF2\_SUBSTRATE\_UNDEFINED** = 0,

**OTF2\_SUBSTRATE\_POSIX** = 1,

**OTF2\_SUBSTRATE\_SION** = 2,

**OTF2\_SUBSTRATE\_NONE** = 3 }

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

## J.14 OTF2\_GeneralDefinitions.h File Reference

---

- enum OTF2\_FileType\_enum {  
    OTF2\_FILETYPE\_ANCHOR = 0,  
    OTF2\_FILETYPE\_GLOBAL\_DEFS = 1,  
    OTF2\_FILETYPE\_LOCAL\_DEFS = 2,  
    OTF2\_FILETYPE\_EVENTS = 3,  
    OTF2\_FILETYPE\_SNAPSHOTS = 4,  
    OTF2\_FILETYPE\_THUMBNAIL = 5,  
    OTF2\_FILETYPE\_MARKER = 6 }

*Defines which file type is used.*

- enum OTF2\_FlushType\_enum {  
    OTF2\_NO\_FLUSH = 0,  
    OTF2\_FLUSH = 1 }

*Defines whether the recorded data is flushed to a file or not.*

- enum OTF2\_MappingType\_enum {  
    OTF2\_MAPPING\_STRING = 0,  
    OTF2\_MAPPING\_ATTRIBUTE = 1,  
    OTF2\_MAPPING\_LOCATION = 2,  
    OTF2\_MAPPING\_REGION = 3,  
    OTF2\_MAPPING\_GROUP = 4,  
    OTF2\_MAPPING\_METRIC = 5,  
    OTF2\_MAPPING\_COMM = 6,  
    OTF2\_MAPPING\_PARAMETER = 7,  
    OTF2\_MAPPING\_RMA\_WIN = 8,  
    OTF2\_MAPPING\_MAX = 9 }

*Possible mappings from local to global identifiers.*

- enum OTF2\_Paradigm\_enum {  
    OTF2\_PARADIGM\_UNKNOWN = 0,  
    OTF2\_PARADIGM\_USER = 1,  
    OTF2\_PARADIGM\_COMPILER = 2,  
    OTF2\_PARADIGM\_OPENMP = 3,  
    OTF2\_PARADIGM\_MPI = 4,  
    OTF2\_PARADIGM\_CUDA = 5,  
    OTF2\_PARADIGM\_MEASUREMENT\_SYSTEM = 6 }

*List of known paradigms.*

- enum `OTF2_ThumbnailType_enum` {  
    `OTF2_THUMBNAIL_TYPE_REGION` = 0,  
    `OTF2_THUMBNAIL_TYPE_METRIC` = 1,  
    `OTF2_THUMBNAIL_TYPE_ATTRIBUTES` = 2 }

*Type of definitions used as metric in an thumbnail.*

- enum `OTF2_Type_enum` {  
    `OTF2_TYPE_NONE` = 0,  
    `OTF2_TYPE_UINT8` = 1,  
    `OTF2_TYPE_UINT16` = 2,  
    `OTF2_TYPE_UINT32` = 3,  
    `OTF2_TYPE_UINT64` = 4,  
    `OTF2_TYPE_INT8` = 5,  
    `OTF2_TYPE_INT16` = 6,  
    `OTF2_TYPE_INT32` = 7,  
    `OTF2_TYPE_INT64` = 8,  
    `OTF2_TYPE_FLOAT` = 9,  
    `OTF2_TYPE_DOUBLE` = 10 }

*OTF2 basic data types.*

### **J.14.1 Detailed Description**

This header file provides general definitions which should be accessible in all internal and external modules.

#### **Source Template:**

*templates/OTF2\_GeneralDefinitions.tpl.h*

#### **Maintainer:**

Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

#### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

## **J.14 OTF2\_GeneralDefinitions.h File Reference**

---

### **J.14.2 Define Documentation**

#### **J.14.2.1 #define OTF2\_UNDEFINED\_TYPE OTF2\_UNDEFINED\_UINT8**

Undefined value for enums

### **J.14.3 Enumeration Type Documentation**

#### **J.14.3.1 enum OTF2\_CallbackCode**

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

Returning *OTF2\_CALLBACK\_INTERRUPT* will stop reading more events, if functions like:

- [OTF2\\_Reader\\_ReadLocalEvents](#)
- [OTF2\\_Reader\\_ReadAllLocalEvents](#)
- [OTF2\\_Reader\\_ReadLocalEventsBackward](#)
- [OTF2\\_Reader\\_ReadGlobalEvents](#)
- [OTF2\\_Reader\\_ReadAllGlobalEvents](#)
- [OTF2\\_Reader\\_ReadLocalDefinitions](#)
- [OTF2\\_Reader\\_ReadAllLocalDefinitions](#)
- [OTF2\\_Reader\\_ReadGlobalDefinitions](#)
- [OTF2\\_Reader\\_ReadAllGlobalDefinitions](#) where called. The return value for these functions is *OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK* in this case. It is valid to call any reader functions in such a condition again.

#### **Enumerator:**

***OTF2\_CALLBACK\_SUCCESS*** Record reading can continue.

***OTF2\_CALLBACK\_INTERRUPT*** Interrupt record reading. Control returns to the caller of the read function with error *OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK* to signal this. The actual value can be any except *OTF2\_CALLBACK\_SUCCESS*.

### **J.14.3.2 enum OTF2\_Compression\_enum**

Defines which compression is used.

#### **Enumerator:**

***OTF2\_COMPRESSION\_UNDEFINED*** Undefined.

***OTF2\_COMPRESSION\_NONE*** No compression is used.

***OTF2\_COMPRESSION\_ZLIB*** Use zlib compression.

### **J.14.3.3 enum OTF2\_FileMode\_enum**

Defines how to interact with files.

#### **Enumerator:**

***OTF2\_FILEMODE\_WRITE*** Open a file in write-only mode.

***OTF2\_FILEMODE\_READ*** Open a file in read-only mode.

***OTF2\_FILEMODE MODIFY*** Open a file in write-read mode.

### **J.14.3.4 enum OTF2\_FileSubstrate\_enum**

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

#### **Enumerator:**

***OTF2\_SUBSTRATE\_UNDEFINED*** Undefined.

***OTF2\_SUBSTRATE\_POSIX*** Use standard posix file interface.

***OTF2\_SUBSTRATE\_SION*** Use the interface of the sionlib to write many logical files into few physical files.

***OTF2\_SUBSTRATE\_NONE*** Do not use any file interface. No data is written to a file.

### **J.14.3.5 enum OTF2\_FileType\_enum**

Defines which file type is used.

---

## **J.14 OTF2\_GeneralDefinitions.h File Reference**

---

### **Enumerator:**

***OTF2\_FILETYPE\_ANCHOR*** Represents the type for the anchor file (.otf2).

Does has a undefined location ID.

***OTF2\_FILETYPE\_GLOBAL\_DEFS*** Represents the type for the global definition file (.def). Does has a undefined location ID.

***OTF2\_FILETYPE\_LOCAL\_DEFS*** Represents the type for a local definition file (.def).

***OTF2\_FILETYPE\_EVENTS*** Represents the type for a event file (.evt).

***OTF2\_FILETYPE\_SNAPSHOTS*** Represents the type for a snapshot file (.snap).

***OTF2\_FILETYPE\_THUMBNAIL*** Represents the type for a thumb file (.thumb).

***OTF2\_FILETYPE\_MARKER*** Represents the type for a marker file (.marker).

### **J.14.3.6 enum OTF2\_FlushType\_enum**

Defines whether the recorded data is flushed to a file or not.

### **Enumerator:**

***OTF2\_NO\_FLUSH*** Flushing will be supressed when running out of memory.

***OTF2\_FLUSH*** Recorded data is flushed when running out of memory.

### **J.14.3.7 enum OTF2\_MappingType\_enum**

Possible mappings from local to global identifiers.

### **Since**

Version 1.0

### **Enumerator:**

***OTF2\_MAPPING\_STRING*** Mapping of string identifiers.

***OTF2\_MAPPING\_ATTRIBUTE*** Mapping of attribute identifiers.

***OTF2\_MAPPING\_LOCATION*** Mapping of location identifiers.

***OTF2\_MAPPING\_REGION*** Mapping of region identifiers.

***OTF2\_MAPPING\_GROUP*** Mapping of group identifiers.

***OTF2\_MAPPING\_METRIC*** Mapping of metric identifiers.

***OTF2\_MAPPING\_COMM*** Mapping of MPI communicator identifiers.

***OTF2\_MAPPING\_PARAMETER*** Mapping of parameter identifiers.

***OTF2\_MAPPING\_RMA\_WIN*** Mapping of RMA window identifiers.

***OTF2\_MAPPING\_MAX*** Max entry.

### **J.14.3.8 enum OTF2\_Paradigm\_enum**

List of known paradigms.

#### **Since**

Version 1.1

#### **Enumerator:**

***OTF2\_PARADIGM\_UNKNOWN*** An unknown paradigm.

***OTF2\_PARADIGM\_USER*** Regions generated through user instrumentation.

***OTF2\_PARADIGM\_COMPILER*** Regions generated through compiler instrumentation.

***OTF2\_PARADIGM\_OPENMP*** Regions referring to OpenMP directives and API functions.

***OTF2\_PARADIGM\_MPI*** Regions referring to MPI functions.

***OTF2\_PARADIGM\_CUDA*** Regions referring to CUDA API functions.

***OTF2\_PARADIGM\_MEASUREMENT\_SYSTEM*** Regions used by the measurement software.

#### **Since**

Version 1.2.

### **J.14.3.9 enum OTF2\_ThumbnailType\_enum**

Type of definitions used as metric in an thumbnail.

#### **Since**

Version 1.2

## **J.15 OTF2\_GlobalDefReader.h File Reference**

---

### **Enumerator:**

- OTF2\_THUMBNAIL\_TYPE\_REGION*** The referenced definitions are of type *Region*.
- OTF2\_THUMBNAIL\_TYPE\_METRIC*** The referenced definitions are of type *MetricMember*.
- OTF2\_THUMBNAIL\_TYPE\_ATTRIBUTES*** The referenced definitions are of type *Attribute*.

### **J.14.3.10 enum OTF2\_Type\_enum**

OTF2 basic data types.

#### **Since**

Version 1.0

### **Enumerator:**

- OTF2\_TYPE\_NONE*** Undefined type.
- OTF2\_TYPE\_UINT8*** Unsigned 8-bit integer.
- OTF2\_TYPE\_UINT16*** Unsigned 16-bit integer.
- OTF2\_TYPE\_UINT32*** Unsigned 32-bit integer.
- OTF2\_TYPE\_UINT64*** Unsigned 64-bit integer.
- OTF2\_TYPE\_INT8*** Signed 8-bit integer.
- OTF2\_TYPE\_INT16*** Signed 16-bit integer.
- OTF2\_TYPE\_INT32*** Signed 32-bit integer.
- OTF2\_TYPE\_INT64*** Signed 64-bit integer.
- OTF2\_TYPE\_FLOAT*** 32-bit floating point value.
- OTF2\_TYPE\_DOUBLE*** 64-bit floating point value.

## **J.15 OTF2\_GlobalDefReader.h File Reference**

This is the definition reader.

```
#include <stddef.h>
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_GlobalDefReaderCallbacks.h>
```

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Functions**

- **OTF2\_ErrorCode OTF2\_GlobalDefReader\_ReadDefinitions** (**OTF2\_GlobalDefReader** \**reader*, **uint64\_t** *recordsToRead*, **uint64\_t** \**recordsRead*)  
*Reads the given number of records from the global definition reader.*
- **OTF2\_ErrorCode OTF2\_GlobalDefReader\_SetCallbacks** (**OTF2\_GlobalDefReader** \**reader*, const **OTF2\_GlobalDefReaderCallbacks** \**callbacks*, void \**userData*)

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

#### **J.15.1 Detailed Description**

This is the definition reader.

#### **Maintainer:**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

#### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

#### **J.15.2 Function Documentation**

##### **J.15.2.1 OTF2\_ErrorCode OTF2\_GlobalDefReader\_ReadDefinitions (** **OTF2\_GlobalDefReader \* reader, uint64\_t recordsToRead, uint64\_t \*** **recordsRead )**

Reads the given number of records from the global definition reader.

#### **Parameters**

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

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.15.2.2 OTF2\_ErrorCode OTF2\_GlobalDefReader\_SetCallbacks ( OTF2\_GlobalDefReader \* reader, const OTF2\_GlobalDefReaderCallbacks \* callbacks, void \* userData )**

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

### Parameters

<i>reader</i>	This given reader object will be setted up with new callback functions.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <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.

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

This defines the callbacks for the global definition reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
```

### Typedefs

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalDefReaderCallback\_Attribute)(void \*userData, OTF2\_AttributeRef self, OTF2StringRef name, OTF2\_Type type)**

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

## **APPENDIX J. FILE DOCUMENTATION**

---

- **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, OTF2StringRef sourceFile, uint32\_t lineNumber, OTF2\_RegionRef enteredRegion, OTF2\_RegionRef leftRegion)**  
*Function pointer definition for the callback which is triggered by a [Callsite](#) definition record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalDefReaderCallback\_ClockProperties )(void \*userData, uint64\_t timerResolution, uint64\_t globalOffset, uint64\_t traceLength)**  
*Function pointer definition for the callback which is triggered by a [ClockProperties](#) definition record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalDefReaderCallback\_Comm )(void \*userData, OTF2\_CommRef self, OTF2StringRef 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, OTF2StringRef 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, OTF2StringRef 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, OTF2StringRef name, OTF2\_-LocationGroupType locationGroupType, OTF2\_SystemTreeNodeRef systemTreeParent)**  
*Function pointer definition for the callback which is triggered by a [Location-Group](#) definition record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalDefReaderCallback\_MetricClass )(void \*userData, OTF2\_MetricRef self, uint8\_t numberOfMetrics, const OTF2\_MetricMemberRef \*metricMembers, OTF2\_MetricOccurrence metricOccurrence, OTF2RecorderKind recorderKind)**

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

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

- `typedef OTF2_CallbackCode(* OTF2_GlobalDefReaderCallback_MetricClassRecorder)(void *userData, OTF2_MetricRef metricClass, OTF2_LocationRef recorder)`

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

- `typedef OTF2_CallbackCode(* OTF2_GlobalDefReaderCallback_MetricInstance)(void *userData, OTF2_MetricRef self, OTF2_MetricRef metricClass, OTF2_LocationRef recorder, OTF2_MetricScope metricScope, uint64_t scope)`

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

- `typedef OTF2_CallbackCode(* OTF2_GlobalDefReaderCallback_MetricMember)(void *userData, OTF2_MetricMemberRef self, OTF2StringRef name, OTF2StringRef 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, OTF2StringRef 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, OTF2StringRef name, OTF2StringRef canonicalName, OTF2StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2StringRef 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, OTF2StringRef 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, OTF2StringRef 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, OTF2StringRef name, OTF2StringRef className, OTF2_SystemTreeNodeRef parent)`

## **APPENDIX J. FILE DOCUMENTATION**

---

*Function pointer definition for the callback which is triggered by a [SystemTreeNode](#)-[ode](#) definition record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalDefReaderCallback\_SystemTreeNodeDomain)(void \*userData, OTF2\_SystemTreeNodeRef systemTreeNode, OTF2\_SystemTreeDomain systemTreeDomain)**

*Function pointer definition for the callback which is triggered by a [SystemTreeNode](#)-[odeDomain](#) definition record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalDefReaderCallback\_SystemTreeNodeProperty)(void \*userData, OTF2\_SystemTreeNodeRef systemTreeNode, OTF2StringRef name, OTF2StringRef value)**

*Function pointer definition for the callback which is triggered by a [SystemTreeNode](#)-[odeProperty](#) definition record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalDefReaderCallback\_Unknown)(void \*userData)**

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

- **typedef struct OTF2\_GlobalDefReaderCallbacks\_struct OTF2\_GlobalDefReaderCallbacks**

*Opaque struct which holds all global definition record callbacks.*

### **Functions**

- **void OTF2\_GlobalDefReaderCallbacks\_Clear (OTF2\_GlobalDefReaderCallbacks \*globalDefReaderCallbacks)**  
*Clears a struct for the global definition callbacks.*
- **void OTF2\_GlobalDefReaderCallbacks\_Delete (OTF2\_GlobalDefReaderCallbacks \*globalDefReaderCallbacks)**  
*Deallocates a struct for the global definition callbacks.*
- **OTF2\_GlobalDefReaderCallbacks \* OTF2\_GlobalDefReaderCallbacks\_New (void)**  
*Allocates a new struct for the global definition callbacks.*
- **OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetAttributeCallback (OTF2\_GlobalDefReaderCallbacks \*globalDefReaderCallbacks, OTF2\_GlobalDefReaderCallback\_Attribute attributeCallback)**  
*Registers the callback for the [Attribute](#) definition.*
- **OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetCallpathCallback (OTF2\_GlobalDefReaderCallbacks \*globalDefReaderCallbacks, OTF2\_GlobalDefReaderCallback\_Callpath callpathCallback)**  
*Registers the callback for the [Callpath](#) definition.*
- **OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetCallsiteCallback (OTF2\_GlobalDefReaderCallbacks \*globalDefReaderCallbacks, OTF2\_GlobalDefReaderCallback\_Callsite callsiteCallback)**

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

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

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetClockPropertiesCallback](#)  
([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_ClockProperties](#) clockPropertiesCallback)

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

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetCommCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_Comm](#) commCallback)

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

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetGroupCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_Group](#) groupCallback)

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

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetLocationCallback](#) ([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_Location](#) locationCallback)

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

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetLocationGroupCallback](#)  
([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_LocationGroup](#) locationGroupCallback)

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

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetMetricClassCallback](#)  
([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_MetricClass](#) metricClassCallback)

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

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetMetricClassRecorderCallback](#)  
([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_MetricClassRecorder](#) metricClassRecorderCallback)

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

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetMetricInstanceCallback](#)  
([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_MetricInstance](#) metricInstanceCallback)

*Registers the callback for the [MetricInstance](#) definition.*

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetMetricMemberCallback](#)  
([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_MetricMember](#) metricMemberCallback)

*Registers the callback for the [MetricMember](#) definition.*

- [OTF2\\_ErrorCode OTF2\\_GlobalDefReaderCallbacks\\_SetParameterCallback](#)  
([OTF2\\_GlobalDefReaderCallbacks](#) \*globalDefReaderCallbacks, [OTF2\\_GlobalDefReaderCallback\\_Parameter](#) parameterCallback)

*Registers the callback for the [Parameter](#) definition.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetRegionCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Region regionCallback)`  
*Registers the callback for the `Region` definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetRmaWinCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_RmaWin rmaWinCallback)`  
*Registers the callback for the `RmaWin` definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetStringCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_String stringCallback)`  
*Registers the callback for the `String` definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_SystemTreeNode systemTreeNodeCallback)`  
*Registers the callback for the `SystemTreeNode` definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodeDomainCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_SystemTreeNodeDomain systemTreeNodeDomainCallback)`  
*Registers the callback for the `SystemTreeNodeDomain` definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetSystemTreeNodePropertyCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_SystemTreeNodeProperty systemTreeNodePropertyCallback)`  
*Registers the callback for the `SystemTreeNodeProperty` definition.*
- `OTF2_ErrorCode OTF2_GlobalDefReaderCallbacks_SetUnknownCallback (OTF2_GlobalDefReaderCallbacks *globalDefReaderCallbacks, OTF2_GlobalDefReaderCallback_Unknown unknownCallback)`  
*Registers the callback for an unknown definition.*

### **J.16.1 Detailed Description**

This defines the callbacks for the global definition reader.

#### **Source Template:**

*templates/OTF2\_GlobalDefReaderCallbacks.tmpl.h*

#### **Maintainer:**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

## **J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference**

---

### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### **J.16.2 Typedef Documentation**

**J.16.2.1 `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_Attribute)(void *userData, OTF2_AttributeRef self, OTF2StringRef name, OTF2_Type type)`**

Function pointer definition for the callback which is triggered by a [Attribute](#) definition record.

#### **Parameters**

<i>userData</i>	User data as set by <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>type</i>	Type of the attribute value.

#### **Since**

Version 1.0

#### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.16.2.2 `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_Callpath)(void *userData, OTF2_CallpathRef self, OTF2_CallpathRef parent, OTF2_RegionRef region)`**

Function pointer definition for the callback which is triggered by a [Callpath](#) definition record.

#### **Parameters**

<i>userData</i>	User data as set by <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>	References a <a href="#">Callpath</a> definition.
<i>region</i>	References a <a href="#">Region</a> definition.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.0

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.16.2.3** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-  
Callsite)(void *userData, OTF2_CallsiteRef self, OTF2StringRef  
sourceFile, uint32_t lineNumber, OTF2_RegionRef enteredRegion,  
OTF2_RegionRef leftRegion)`

Function pointer definition for the callback which is triggered by a [Callsite](#) definition record.

### **Parameters**

<i>userData</i>	User data as set by <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*.

**J.16.2.4** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-  
ClockProperties)(void *userData, uint64_t timerResolution, uint64_t  
globalOffset, uint64_t traceLength)`

Function pointer definition for the callback which is triggered by a [ClockProperties](#) definition record.

Defines the timer resolution and time range of this trace. There will be no event with a timestamp less than *globalOffset*, and no event with timestamp greater than (*globalOffset* + *traceLength*).

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Parameters

<i>userData</i>	User data as set by <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](#).

**J.16.2.5 `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_`**  
**`Comm)(void *userData, OTF2_CommRef self, OTF2_StringRef name,`**  
**`OTF2_GroupRef group, OTF2_CommRef parent)`**

Function pointer definition for the callback which is triggered by a [Comm](#) definition record.

### Parameters

<i>userData</i>	User data as set by <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 <code>OTF2_GROUP_TYPE_MPI_GROUP</code> or <code>OTF2_GROUP_TYPE_MPI_COMM_SELF</code> . 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*.

J.16.2.6 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalDefReaderCallback\_Group)(void \*userData, OTF2\_GroupRef self, OTF2StringRef name, OTF2\_GroupType groupType, OTF2\_Paradigm paradigm, OTF2\_GroupFlag groupFlags, uint32\_t numberOfMembers, const uint64\_t \*members)**

Function pointer definition for the callback which is triggered by a [Group](#) definition record.

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_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*.

J.16.2.7 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalDefReaderCallback\_Location)(void \*userData, OTF2\_LocationRef self, OTF2StringRef name, OTF2\_LocationType locationType, uint64\_t numberOfEvents, OTF2\_LocationGroupRef locationGroup)**

Function pointer definition for the callback which is triggered by a [Location](#) definition record.

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Parameters

<i>userData</i>	User data as set by <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>numberO- fEvents</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](#).

**J.16.2.8 `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-  
LocationGroup)(void *userData, OTF2_LocationGroupRef self,  
OTF2StringRef name, OTF2_LocationGroupType locationGroupType,  
OTF2_SystemTreeNodeRef systemTreeParent)`**

Function pointer definition for the callback which is triggered by a [LocationGroup](#) definition record.

### Parameters

<i>userData</i>	User data as set by <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>sys- temTreePar- ent</i>	Parent of this location group in the system tree. References a <a href="#">Sys- temTreeNode</a> definition.

### Since

Version 1.0

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.16.2.9** **typedef OTF2\_CallbackCode( \* OTF2\_GlobalDefReaderCallback\_-MetricClass)(void \*userData, OTF2\_MetricRef self, uint8\_t numberOfMetrics, const OTF2\_MetricMemberRef \*metricMembers, OTF2\_MetricOccurrence metricOccurrence, OTF2\_RecorderKind recorderKind)**

Function pointer definition for the callback which is triggered by a [MetricClass](#) definition record.

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

### **Parameters**

<i>userData</i>	User data as set by <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

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.16.2.10** **typedef OTF2\_CallbackCode( \* OTF2\_GlobalDefReaderCallback\_-MetricClassRecorder)(void \*userData, OTF2\_MetricRef metricClass, OTF2\_LocationRef recorder)**

Function pointer definition for the callback which is triggered by a [MetricClassRecorder](#) definition record.

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Parameters

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_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](#).

**J.16.2.11** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-MetricInstance)(void *userData, OTF2_MetricRef self,  
OTF2_MetricRef metricClass, OTF2_LocationRef recorder,  
OTF2_MetricScope metricScope, uint64_t scope)`

Function pointer definition for the callback which is triggered by a [MetricInstance](#) definition record.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type [OTF2\\_METRIC\\_ASYNCROUS](#).

### 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>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

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.16.2.12 typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-
MetricMember)(void *userData, OTF2_MetricMemberRef
self, OTF2_StringRef name, OTF2_StringRef description,
OTF2_MetricType metricType, OTF2_MetricMode metricMode,
OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent,
OTF2_StringRef unit)
```

Function pointer definition for the callback which is triggered by a [MetricMember](#) definition record.

A metric is defined by a metric member definition. A metric member is always a member of a metric class. Therefore, a single metric is a special case of a metric class with only one member. It is not allowed to reference a metric member id in a metric event, but only metric class IDs.

### **Parameters**

<i>userData</i>	User data as set by <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.
<i>valueType</i>	Type of the value: int64_t, uint64_t, or double.
<i>metricBase</i>	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor base <sup>exponent</sup> , 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.

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.16.2.13** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-Parameter)(void *userData, OTF2_ParameterRef self, OTF2StringRef name, OTF2_ParameterType parameterType)`

Function pointer definition for the callback which is triggered by a [Parameter](#) definition record.

### Parameters

<i>userData</i>	User data as set by <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*.

**J.16.2.14** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-Region)(void *userData, OTF2_RegionRef self, OTF2StringRef name, OTF2StringRef canonicalName, OTF2StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2StringRef sourceFile, uint32_t beginLineNumber, uint32_t endLineNumber)`

Function pointer definition for the callback which is triggered by a [Region](#) definition record.

### Parameters

## APPENDIX J. FILE DOCUMENTATION

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_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](#).

```
J.16.2.15 typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-
RmaWin)(void *userData, OTF2_RmaWinRef self, OTF2StringRef
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.

## **J.16 OTF2\_GlobalDefReaderCallbacks.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*.

**J.16.2.16 `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-String)(void *userData, OTF2_StringRef self, const char *string)`**

Function pointer definition for the callback which is triggered by a [String](#) definition record.

### **Parameters**

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_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*.

**J.16.2.17 `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-SystemTreeNode)(void *userData, OTF2_SystemTreeNodeRef self, OTF2_StringRef name, OTF2_StringRef className, OTF2_SystemTreeNodeRef parent)`**

Function pointer definition for the callback which is triggered by a [SystemTreeNode](#) definition record.

### **Parameters**

---

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>userData</i>	User data as set by <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](#).

**J.16.2.18** `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_ - SystemTreeNodeDomain)(void *userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_SystemTreeDomain systemTreeDomain)`

Function pointer definition for the callback which is triggered by a [SystemTreeNodeDomain](#) definition record.

### **Parameters**

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_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.

### **Since**

Version 1.2

### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## **J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference**

---

**J.16.2.19 `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-SystemTreeNodeProperty)(void *userData, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_StringRef name, OTF2_StringRef value)`**

Function pointer definition for the callback which is triggered by a [SystemTreeNodeProperty](#) definition record.

### **Parameters**

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_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>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\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.16.2.20 `typedef OTF2_CallbackCode( * OTF2_GlobalDefReaderCallback_-Unknown)(void *userData)`**

Function pointer definition for the callback which is triggered by an unknown definition record.

### **Parameters**

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalDefCallbacks</a> or <a href="#">OTF2_GlobalDefReader_SetCallbacks</a> .
-----------------	---

### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## **APPENDIX J. FILE DOCUMENTATION**

---

### **J.16.3 Function Documentation**

**J.16.3.1 void OTF2\_GlobalDefReaderCallbacks\_Clear ( OTF2\_-  
GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*  
)**

Clears a struct for the global definition callbacks.

#### **Parameters**

<i>globalDef- Reader- Callbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_- GlobalDefReaderCallbacks_New</a> .
---	--

**J.16.3.2 void OTF2\_GlobalDefReaderCallbacks\_Delete ( OTF2\_-  
GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*  
)**

Deallocates a struct for the global definition callbacks.

#### **Parameters**

<i>globalDef- Reader- Callbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_- GlobalDefReaderCallbacks_New</a> .
---	--

**J.16.3.3 OTF2\_GlobalDefReaderCallbacks\* OTF2\_GlobalDefReaderCallbacks\_New (  
void )**

Allocates a new struct for the global definition callbacks.

#### **Returns**

A newly allocated struct of type [OTF2\\_GlobalDefReaderCallbacks](#).

**J.16.3.4 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetAttributeCallback  
( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*,  
OTF2\_GlobalDefReaderCallback\_Attribute *attributeCallback* )**

Registers the callback for the [Attribute](#) definition.

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>attribute- Callback</i>	Function which should be called for all <a href="#">Attribute</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

### J.16.3.5 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 <a href="#">Callpath</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

### J.16.3.6 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetCallsiteCallback ( `OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks,` `OTF2_GlobalDefReaderCallback_Callsite callsiteCallback` )

Registers the callback for the [Callsite](#) definition.

### Parameters

---

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>globalDefReaderCallbacks</i>	Struct for all callbacks.
<i>callsiteCallback</i>	Function which should be called for all <a href="#">Callsite</a> definitions.

### **Returns**

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

**J.16.3.7 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetClockPropertiesCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_ClockProperties *clockPropertiesCallback* )**

Registers the callback for the [ClockProperties](#) definition.

### **Parameters**

<i>globalDefReaderCallbacks</i>	Struct for all callbacks.
<i>clockPropertiesCallback</i>	Function which should be called for all <a href="#">ClockProperties</a> definitions.

### **Returns**

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

**J.16.3.8 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetCommCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_Comm *commCallback* )**

Registers the callback for the [Comm](#) definition.

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>commCall- back</i>	Function which should be called for all <a href="#">Comm</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

**J.16.3.9 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetGroupCallback**  
( `OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks,`  
`OTF2_GlobalDefReaderCallback_Group groupCallback` )

Registers the callback for the [Group](#) definition.

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>groupCall- back</i>	Function which should be called for all <a href="#">Group</a> definitions.

### Returns

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

**J.16.3.10 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetLocationCallback**  
( `OTF2_GlobalDefReaderCallbacks * globalDefReaderCallbacks,`  
`OTF2_GlobalDefReaderCallback_Location locationCallback` )

Registers the callback for the [Location](#) definition.

### Parameters

---

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>location-Callback</i>	Function which should be called for all <a href="#">Location</a> definitions.

### **Returns**

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

**J.16.3.11 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetLocationGroupCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_LocationGroup *locationGroupCallback* )**

Registers the callback for the [LocationGroup](#) definition.

### **Parameters**

<i>globalDef-Reader-Callbacks</i>	Struct for all callbacks.
<i>location-GroupCallback</i>	Function which should be called for all <a href="#">LocationGroup</a> definitions.

### **Returns**

[\*\*OTF2\\_SUCCESS\*\*](#) if successful

[\*\*OTF2\\_ERROR\\_INVALID\\_ARGUMENT\*\*](#) for an invalid `defReaderCallbacks` argument

**J.16.3.12 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetMetricClassCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_MetricClass *metricClassCallback* )**

Registers the callback for the [MetricClass](#) definition.

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>metric- ClassCall- back</i>	Function which should be called for all <a href="#">MetricClass</a> definitions.

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid `defReaderCallbacks` argument

**J.16.3.13 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_-  
SetMetricClassRecorderCallback ( OTF2\_GlobalDefReaderCallbacks  
\* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_-  
MetricClassRecorder *metricClassRecorderCallback*  
)**

Registers the callback for the [MetricClassRecorder](#) definition.

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>metric- Class- Recorder- Callback</i>	Function which should be called for all <a href="#">MetricClassRecorder</a> definitions.

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.16.3.14 **OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_-  
SetMetricInstanceCallback ( OTF2\_GlobalDefReaderCallbacks \*  
globalDefReaderCallbacks, OTF2\_GlobalDefReaderCallback\_-  
MetricInstance metricInstanceCallback )**

Registers the callback for the [MetricInstance](#) definition.

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>metricIn- stanceCall- back</i>	Function which should be called for all <a href="#">MetricInstance</a> definitions.

### Returns

[OTF2\\_SUCCESS](#) if successful  
[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid defReaderCallbacks argument

J.16.3.15 **OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_-  
SetMetricMemberCallback ( OTF2\_GlobalDefReaderCallbacks \*  
globalDefReaderCallbacks, OTF2\_GlobalDefReaderCallback\_-  
MetricMember metricMemberCallback )**

Registers the callback for the [MetricMember](#) definition.

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>metricMem- berCallback</i>	Function which should be called for all <a href="#">MetricMember</a> definitions.

### Returns

[OTF2\\_SUCCESS](#) if successful  
[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid defReaderCallbacks argument

## J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference

---

J.16.3.16 **OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetParameterCallback**  
( **OTF2\_GlobalDefReaderCallbacks \* globalDefReaderCallbacks,**  
**OTF2\_GlobalDefReaderCallback\_Parameter parameterCallback** )

Registers the callback for the [Parameter](#) definition.

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>parameter- Callback</i>	Function which should be called for all <a href="#">Parameter</a> definitions.

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid `defReaderCallbacks` argument

J.16.3.17 **OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetRegionCallback**  
( **OTF2\_GlobalDefReaderCallbacks \* globalDefReaderCallbacks,**  
**OTF2\_GlobalDefReaderCallback\_Region regionCallback** )

Registers the callback for the [Region](#) definition.

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>regionCall- back</i>	Function which should be called for all <a href="#">Region</a> definitions.

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

**J.16.3.18 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetRmaWinCallback**  
( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*,  
OTF2\_GlobalDefReaderCallback\_RmaWin *rmaWinCallback* )

Registers the callback for the [RmaWin](#) definition.

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>rmaWin- Callback</i>	Function which should be called for all <a href="#">RmaWin</a> definitions.

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid *defReaderCallbacks* argument

**J.16.3.19 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetStringCallback**  
( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*,  
OTF2\_GlobalDefReaderCallback\_String *stringCallback* )

Registers the callback for the [String](#) definition.

### Parameters

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>stringCall- back</i>	Function which should be called for all <a href="#">String</a> definitions.

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid *defReaderCallbacks* argument

## **J.16 OTF2\_GlobalDefReaderCallbacks.h File Reference**

---

**J.16.3.20 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_-  
SetSystemTreeNodeCallback ( OTF2\_GlobalDefReaderCallbacks  
\* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_-  
SystemTreeNode *systemTreeNodeCallback* )**

Registers the callback for the [SystemTreeNode](#) definition.

### **Parameters**

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>sys- temTreeN- odeCall- back</i>	Function which should be called for all <a href="#">SystemTreeNode</a> definitions.

### **Returns**

**[OTF2\\_SUCCESS](#)** if successful  
**[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#)** for an invalid `defReaderCallbacks` argument

**J.16.3.21 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_-  
SetSystemTreeNodeDomainCallback ( OTF2\_GlobalDefReaderCallbacks  
\* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_-  
SystemTreeNodeDomain *systemTreeNodeDomainCallback*  
)**

Registers the callback for the [SystemTreeNodeDomain](#) definition.

### **Parameters**

<i>globalDef- Reader- Callbacks</i>	Struct for all callbacks.
<i>sys- temTreeN- odeDo- mainCall- back</i>	Function which should be called for all <a href="#">SystemTreeNodeDomain</a> definitions.

## APPENDIX J. FILE DOCUMENTATION

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.16.3.22 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetSystemTreeNodePropertyCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_ SystemTreeNodeProperty *systemTreeNodePropertyCallback* )**

Registers the callback for the `SystemTreeNodeProperty` definition.

### Parameters

<code>globalDefReaderCallbacks</code>	Struct for all callbacks.
<code>systemTreeNodePropertyCallback</code>	Function which should be called for all <code>SystemTreeNodeProperty</code> definitions.

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.16.3.23 OTF2\_ErrorCode OTF2\_GlobalDefReaderCallbacks\_SetUnknownCallback ( OTF2\_GlobalDefReaderCallbacks \* *globalDefReaderCallbacks*, OTF2\_GlobalDefReaderCallback\_Unknown *unknownCallback* )**

Registers the callback for an unknown definition.

### Parameters

<code>globalDefReaderCallbacks</code>	Struct for all callbacks.
<code>unknownCallback</code>	Function which should be called for all Unknown definitions.

## J.17 OTF2\_GlobalDefWriter.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.17 OTF2\_GlobalDefWriter.h File Reference

This layer always writes globally defined OTF2 definition records and is used to write either the global definitions in addition to local definitions or write all definitions as globally valid in combination with OTF2\_GlobalEventWriter. Global definitions are stored in one global definition file, which makes it nearly impossible to write them in a distributed manner. It is therefore only allowed to get such a writer from an OTF2\_ArchiveHandler which is marked as OTF2\_MASTER.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
```

### Typedefs

- **typedef struct OTF2\_GlobalDefWriter\_struct OTF2\_GlobalDefWriter**

*Typedef of the struct which keeps all necessary information of a global definition writer. Can be used to reference these structs from external.*

### Functions

- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_GetNumberOfDefinitions (OTF2\_GlobalDefWriter \*writerHandle, uint64\_t \*numberOfDefinitions)**  
*Returns the current number of written definitions of a global definition writer.*
- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_GetNumberOfLocations (OTF2\_GlobalDefWriter \*writerHandle, uint64\_t \*numberOfLocations)**  
*Returns the current number of written location definitions of a global definition writer.*
- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteAttribute (OTF2\_GlobalDefWriter \*writerHandle, OTF2\_AttributeRef self, OTF2StringRef name, OTF2\_Type type)**  
*Writes a `Attribute` definition record into the `GlobalDefWriter`.*
- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteCallpath (OTF2\_GlobalDefWriter \*writerHandle, OTF2\_CallpathRef self, OTF2\_CallpathRef parent, OTF2\_RegionRef region)**

## **APPENDIX J. FILE DOCUMENTATION**

---

*Writes a [Callpath](#) definition record into the GlobalDefWriter.*

- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteCallsite (OTF2\_GlobalDefWriter \*writerHandle, OTF2\_CallsiteRef self, OTF2StringRef sourceFile, uint32\_t lineNumber, OTF2\_RegionRef enteredRegion, OTF2\_RegionRef leftRegion)**

*Writes a [Callsite](#) definition record into the GlobalDefWriter.*

- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteClockProperties (OTF2\_GlobalDefWriter \*writerHandle, uint64\_t timerResolution, uint64\_t globalOffset, uint64\_t traceLength)**

*Writes a [ClockProperties](#) definition record into the GlobalDefWriter.*

- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteComm (OTF2\_GlobalDefWriter \*writerHandle, OTF2\_CommRef self, OTF2StringRef 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, OTF2StringRef 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, OTF2StringRef 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, OTF2StringRef name, OTF2\_LocationGroupType locationGroupType, OTF2\_SystemTreeNodeRef systemTreeParent)**

*Writes a [LocationGroup](#) definition record into the GlobalDefWriter.*

- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteMetricClass (OTF2\_GlobalDefWriter \*writerHandle, OTF2\_MetricRef self, uint8\_t numberOfMetrics, const OTF2\_MetricMemberRef \*metricMembers, OTF2\_MetricOccurrence metricOccurrence, OTF2RecorderKind recorderKind)**

*Writes a [MetricClass](#) definition record into the GlobalDefWriter.*

- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteMetricClassRecorder (OTF2\_GlobalDefWriter \*writerHandle, OTF2\_MetricRef metricClass, OTF2\_LocationRef recorder)**

*Writes a [MetricClassRecorder](#) definition record into the GlobalDefWriter.*

- **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteMetricInstance (OTF2\_GlobalDefWriter \*writerHandle, OTF2\_MetricRef self, OTF2\_MetricRef metricClass, OTF2\_LocationRef recorder, OTF2\_MetricScope metricScope, uint64\_t scope)**

## J.17 OTF2\_GlobalDefWriter.h File Reference

---

Writes a *MetricInstance* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteMetricMember (OTF2_GlobalDefWriter *writerHandle, OTF2_MetricMemberRef self, OTF2StringRef name, OTF2StringRef description, OTF2_MetricType metricType, OTF2_MetricMode metricMode, OTF2_Type valueType, OTF2_MetricBase metricBase, int64_t exponent, OTF2StringRef unit)`

Writes a *MetricMember* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteParameter (OTF2_GlobalDefWriter *writerHandle, OTF2_ParameterRef self, OTF2StringRef name, OTF2ParameterType parameterType)`

Writes a *Parameter* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteRegion (OTF2_GlobalDefWriter *writerHandle, OTF2_RegionRef self, OTF2StringRef name, OTF2StringRef canonicalName, OTF2StringRef description, OTF2_RegionRole regionRole, OTF2_Paradigm paradigm, OTF2_RegionFlag regionFlags, OTF2StringRef 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, OTF2StringRef name, OTF2_CommRef comm)`

Writes a *RmaWin* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteString (OTF2_GlobalDefWriter *writerHandle, OTF2StringRef self, const char *string)`

Writes a *String* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNode (OTF2_GlobalDefWriter *writerHandle, OTF2_SystemTreeNodeRef self, OTF2StringRef name, OTF2StringRef className, OTF2_SystemTreeNodeRef parent)`

Writes a *SystemTreeNode* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNodeDomain (OTF2_GlobalDefWriter *writerHandle, OTF2_SystemTreeNodeRef systemTreeNode, OTF2_SystemDomain systemTreeDomain)`

Writes a *SystemTreeNodeDomain* definition record into the *GlobalDefWriter*.

- `OTF2_ErrorCode OTF2_GlobalDefWriter_WriteSystemTreeNodeProperty (OTF2_GlobalDefWriter *writerHandle, OTF2_SystemTreeNodeRef systemTreeNode, OTF2StringRef name, OTF2StringRef value)`

Writes a *SystemTreeNodeProperty* definition record into the *GlobalDefWriter*.

### J.17.1 Detailed Description

This layer always writes globally defined OTF2 definition records and is used to write either the global definitions in addition to local definitions or write all def-

## **APPENDIX J. FILE DOCUMENTATION**

---

initions as globally valid in combination with OTF2\_GlobalEventWriter. Global definitions are stored in one global definition file, which makes it nearly impossible to write them in a distributed manner. It is therefore only allowed to get such a writer from an OTF2\_ArchiveHandler which is marked as OTF2\_MASTER.

### **Source Template:**

*templates/OTF2\_GlobalDefWriter tmpl.h*

### **Maintainer:**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

## **J.17.2 Function Documentation**

### **J.17.2.1 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_GetNumberOfDefinitions (**

**OTF2\_GlobalDefWriter \* writerHandle, uint64\_t \* numberOfDefinitions )**

Returns the current number of written definitions of a global definition writer.

#### **Parameters**

	<i>writerHandle</i>	Handle to the global definition writer.
out	<i>numberOfDefinitions</i>	Storage for the number of definitions.

#### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### **J.17.2.2 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_GetNumberOfLocations (**

**OTF2\_GlobalDefWriter \* writerHandle, uint64\_t \* numberOfLocations )**

Returns the current number of written location definitions of a global definition writer.

#### **Parameters**

## J.17 OTF2\_GlobalDefWriter.h File Reference

---

	<i>writerHandle</i>	Handle to the global definition writer.
out	<i>numberOfLocations</i>	Storage for the number of locations.

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.17.2.3 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteAttribute (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_AttributeRef self,**  
**OTF2\_StringRef name, OTF2\_Type type )**

Writes a [Attribute](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">Attribute</a> definition.
<i>name</i>	Name of the attribute. References a <a href="#">String</a> definition.
<i>type</i>	Type of the attribute value.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.17.2.4 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteCallpath (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_CallpathRef self,**  
**OTF2\_CallpathRef parent, OTF2\_RegionRef region )**

Writes a [Callpath](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
---------------------	--------------------

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>self</i>	The unique identifier for this <a href="#">Callpath</a> definition.
<i>parent</i>	References a <a href="#">Callpath</a> definition.
<i>region</i>	References a <a href="#">Region</a> definition.

### **Since**

Version 1.0

### **Returns**

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.17.2.5 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteCallsite (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_CallsiteRef self,**  
**OTF2StringRef sourceFile, uint32\_t lineNumber, OTF2\_RegionRef**  
**enteredRegion, OTF2\_RegionRef leftRegion )**

Writes a [Callsite](#) definition record into the GlobalDefWriter.

### **Parameters**

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <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.

**J.17.2.6 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.

## J.17 OTF2\_GlobalDefWriter.h File Reference

---

Defines the timer resolution and time range of this trace. There will be no event with a timestamp less than *globalOffset*, and no event with timestamp greater than (*globalOffset* + *traceLength*).

### Parameters

<i>writerHandle</i>	The writer handle.
<i>timerResolution</i>	Ticks per seconds.
<i>globalOffset</i>	A timestamp smaller than all event timestamps.
<i>traceLength</i>	A timespan which includes the timespan between the smallest and greatest timestamp of all event timestamps.

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.17.2.7 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteComm (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_CommRef self,**  
**OTF2StringRef name, OTF2\_GroupRef group, OTF2\_CommRef**  
**parent )**

Writes a [Comm](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <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 <code>OTF2_GROUP_TYPE_MPI_GROUP</code> or <code>OTF2_GROUP_TYPE_MPI_COMM_SELF</code> . References a <a href="#">Group</a> definition.
<i>parent</i>	The parent MPI communicator from which this communicator was created, if any. Use <code>OTF2_UNDEFINED_COMM</code> to indicate no parent. References a <a href="#">Comm</a> definition.

## APPENDIX J. FILE DOCUMENTATION

### Since

Version 1.0

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.17.2.8 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteGroup (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_GroupRef**  
**self, OTF2StringRef name, OTF2\_GroupType groupType,**  
**OTF2\_Paradigm paradigm, OTF2\_GroupFlag groupFlags, uint32\_t**  
**numberOfMembers, const uint64\_t \* members )**

Writes a [Group](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <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.

**J.17.2.9 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.

## J.17 OTF2\_GlobalDefWriter.h File Reference

---

### 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="#">LocationGroup</a> definition.

### Since

Version 1.0

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.17.2.10 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteLocationGroup (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_LocationGroupRef**  
**self, OTF2StringRef name, OTF2\_LocationGroupType**  
**locationGroupType, OTF2\_SystemTreeNodeRef systemTreeParent )**

Writes a [LocationGroup](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">LocationGroup</a> definition.
<i>name</i>	Name of the group. References a <a href="#">String</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="#">SystemTreeNode</a> definition.

### Since

Version 1.0

## APPENDIX J. FILE DOCUMENTATION

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

J.17.2.11 **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteMetricClass (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_MetricRef self, uint8\_t**  
**numberOfMetrics, const OTF2\_MetricMemberRef \* metricMembers,**  
**OTF2\_MetricOccurrence metricOccurrence, OTF2\_RecorderKind**  
**recorderKind )**

Writes a [MetricClass](#) definition record into the GlobalDefWriter.

For a metric class it is implicitly given that the event stream that records the metric is also the scope. A metric class can contain multiple different metrics.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <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.

J.17.2.12 **OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteMetricClassRecorder (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_MetricRef metricClass,**  
**OTF2\_LocationRef recorder )**

Writes a [MetricClassRecorder](#) definition record into the GlobalDefWriter.

### Parameters

## J.17 OTF2\_GlobalDefWriter.h File Reference

---

<i>writerHandle</i>	The writer handle.
<i>metricClass</i>	Parent <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.

**J.17.2.13 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteMetricInstance  
( OTF2\_GlobalDefWriter \* *writerHandle*, OTF2\_MetricRef  
  *self*, OTF2\_MetricRef *metricClass*, OTF2\_LocationRef *recorder*,  
  OTF2\_MetricScope *metricScope*, uint64\_t *scope* )**

Writes a [MetricInstance](#) definition record into the GlobalDefWriter.

A metric instance is used to define metrics that are recorded at one location for multiple locations or for another location. The occurrence of a metric instance is implicitly of type [OTF2\\_METRIC\\_ASYNCROUS](#).

### 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.

**J.17.2.14 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteMetricMember (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_MetricMemberRef**  
**self, OTF2StringRef name, OTF2StringRef description,**  
**OTF2\_MetricType metricType, OTF2\_MetricMode metricMode,**  
**OTF2\_Type valueType, OTF2\_MetricBase metricBase, int64\_t exponent,**  
**OTF2StringRef 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: int64_t, uint64_t, or double.
<i>metricBase</i>	The recorded values should be handled in this given base, either binary or decimal. This information can be used if the value needs to be scaled.
<i>exponent</i>	The values inside the Metric events should be scaled by the factor base <sup>exponent</sup> , 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

## J.17 OTF2\_GlobalDefWriter.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.17.2.15 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteParameter (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_ParameterRef self,**  
**OTF2StringRef name, OTF2\_ParameterType parameterType )**

Writes a [Parameter](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">Parameter</a> definition.
<i>name</i>	Name of the parameter (variable name etc.) References a <a href="#">String</a> definition.
<i>parameterType</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.

**J.17.2.16 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteRegion (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_RegionRef**  
**self, OTF2StringRef name, OTF2StringRef canonicalName,**  
**OTF2StringRef description, OTF2\_RegionRole regionRole,**  
**OTF2\_Paradigm paradigm, OTF2\_RegionFlag regionFlags,**  
**OTF2StringRef sourceFile, uint32\_t beginLineNumber, uint32\_t**  
**endLineNumber )**

Writes a [Region](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <a href="#">Region</a> definition.

## APPENDIX J. FILE DOCUMENTATION

---

<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\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.17.2.17 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteRmaWin (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_RmaWinRef self,**  
**OTF2StringRef 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.

## J.17 OTF2\_GlobalDefWriter.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.17.2.18 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteString (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2StringRef self, const**  
**char \* string )**

Writes a [String](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <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.

**J.17.2.19 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteSystemTreeNode (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_SystemTreeNodeRef**  
**self, OTF2StringRef name, OTF2StringRef className,**  
**OTF2\_SystemTreeNodeRef parent )**

Writes a [SystemTreeNode](#) definition record into the GlobalDefWriter.

### Parameters

<i>writerHandle</i>	The writer handle.
<i>self</i>	The unique identifier for this <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\_SUCCESS* if successful, an error code if an error occurs.

**J.17.2.20 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteSystemTreeNodeDomain (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_SystemTreeNodeRef**  
**systemTreeNode, OTF2\_SystemTreeDomain systemTreeDomain )**

Writes a [SystemTreeNodeDomain](#) definition record into the GlobalDefWriter.

**Parameters**

<i>writerHandle</i>	The writer handle.
<i>systemTreeNode</i>	Parent <a href="#">SystemTreeNode</a> definition to which this one is a supplementary definition. References a <a href="#">SystemTreeNode</a> definition.

**Since**

Version 1.2

**Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.17.2.21 OTF2\_ErrorCode OTF2\_GlobalDefWriter\_WriteSystemTreeNodeProperty (**  
**OTF2\_GlobalDefWriter \* writerHandle, OTF2\_SystemTreeNodeRef**  
**systemTreeNode, OTF2\_StringRef name, OTF2\_StringRef value )**

Writes a [SystemTreeNodeProperty](#) definition record into the GlobalDefWriter.

**Parameters**

<i>writerHandle</i>	The writer handle.
<i>systemTreeNode</i>	Parent <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.

## **J.18 OTF2\_GlobalEvtReader.h File Reference**

---

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## **J.18 OTF2\_GlobalEvtReader.h File Reference**

This is the global event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_EvtReader.h>
#include <otf2/OTF2_GlobalEvtReaderCallbacks.h>
```

### **Functions**

- **OTF2\_ErrorCode OTF2\_GlobalEvtReader\_HasEvent (OTF2\_GlobalEvtReader \*reader, int \*flag)**  
*Has more events.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReader\_ReadEvent (OTF2\_GlobalEvtReader \*reader)**  
*Triggers the callback for the next event record.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReader\_ReadEvents (OTF2\_GlobalEvtReader \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)**  
*Reads the given number of records from the global event reader.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReader\_SetCallbacks (OTF2\_GlobalEvtReader \*reader, const OTF2\_GlobalEvtReaderCallbacks \*callbacks, void \*userData)**

*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### **J.18.1 Detailed Description**

This is the global event reader.

#### **Maintainer:**

Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

Used to read from multiple local event readers, and provide them in a timely ordered sequence.

### **J.18.2 Function Documentation**

#### **J.18.2.1 OTF2\_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 event to read. To 0 if not. Otherwise the value is undefined.

##### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.18.2.2 OTF2\_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.
---------------	---

##### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## **J.18 OTF2\_GlobalEvtReader.h File Reference**

---

**J.18.2.3 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 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.

**J.18.2.4 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="#">OTF2_GlobalEvtReaderCallbacks_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.

### J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

This defines the callbacks for the global event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

#### TypeDefs

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_BufferFlush )(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp stopTime)`

*Callback for the BufferFlush event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_Enter )(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

*Callback for the Enter event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_Leave )(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

*Callback for the Leave event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MeasurementOnOff )(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_MeasurementMode measurementMode)`

*Callback for the MeasurementOnOff event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_Metric )(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)`

*Callback for the Metric event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiCollectiveBegin )(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList)`

*Callback for the MpiCollectiveBegin event record.*

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)`

*Callback for the MpiCollectiveEnd event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiIrecv)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`

*Callback for the MpiIrecv event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiIrecvRequest)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

*Callback for the MpiIrecvRequest event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiIsend)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`

*Callback for the MpiIsend event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiIsendComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

*Callback for the MpiIsendComplete event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiRecv)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

*Callback for the MpiRecv event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiRequestCancelled)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

*Callback for the MpiRequestCancelled event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiRequestTest)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

*Callback for the MpiRequestTest event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_MpiSend)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

## **APPENDIX J. FILE DOCUMENTATION**

---

*Callback for the MpiSend event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpAcquireLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t lockID, uint32\_t acquisitionOrder)**

*Callback for the OmpAcquireLock event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpFork)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t numberRequestedThreads)**

*Callback for the OmpFork event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpJoin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList)**

*Callback for the OmpJoin event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpReleaseLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t lockID, uint32\_t acquisitionOrder)**

*Callback for the OmpReleaseLock event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpTaskComplete)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)**

*Callback for the OmpTaskComplete event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpTaskCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)**

*Callback for the OmpTaskCreate event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_OmpTaskSwitch)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)**

*Callback for the OmpTaskSwitch event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ParameterInt)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, int64\_t value)**

*Callback for the ParameterInt event record.*

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ParameterString)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, OTF2\_StringRef string)**

*Callback for the ParameterString event record.*

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef parameter, uint64_t value)`

*Callback for the ParameterUnsignedInt event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)`

*Callback for the RmaAcquireLock event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaAtomic)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, OTF2_RmaAtomicType type, uint64_t bytesSent, uint64_t bytesReceived, uint64_t matchingId)`

*Callback for the RmaAtomic event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList)`

*Callback for the RmaCollectiveBegin event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t bytesReceived)`

*Callback for the RmaCollectiveEnd event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaGet)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)`

*Callback for the RmaGet event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaGroupSync)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaSyncLevel syncLevel, OTF2_RmaWinRef win, OTF2_GroupRef group)`

*Callback for the RmaGroupSync event record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalEvtReaderCallback_RmaOpCompleteBlocking)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

*Callback for the RmaOpCompleteBlocking event record.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteNonBlocking)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)**

*Callback for the RmaOpCompleteNonBlocking event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteRemote)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)**

*Callback for the RmaOpCompleteRemote event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaOpTest)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)**

*Callback for the RmaOpTest event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaPut)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t bytes, uint64\_t matchingId)**

*Callback for the RmaPut event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaReleaseLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId)**

*Callback for the RmaReleaseLock event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaRequestLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType)**

*Callback for the RmaRequestLock event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaSync)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, OTF2\_RmaSyncType syncType)**

*Callback for the RmaSync event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaTryLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, uint64\_t lockId, OTF2\_LockType lockType)**

*Callback for the RmaTryLock event record.*

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaWaitChange)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)**

*Callback for the RmaWaitChange event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaWinCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)**

*Callback for the RmaWinCreate event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_RmaWinDestroy)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)**

*Callback for the RmaWinDestroy event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadAcquireLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model, uint32\_t lockID, uint32\_t acquisitionOrder)**

*Callback for the ThreadAcquireLock event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadFork)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model, uint32\_t num-berOfRequestedThreads)**

*Callback for the ThreadFork event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadJoin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model)**

*Callback for the ThreadJoin event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadReleaseLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_Paradigm model, uint32\_t lockID, uint32\_t acquisitionOrder)**

*Callback for the ThreadReleaseLock event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadTaskComplete)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam, uint32\_t creatingThread, uint32\_t generationNumber)**

*Callback for the ThreadTaskComplete event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadTaskCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam, uint32\_t creatingThread, uint32\_t generationNumber)**

*Callback for the ThreadTaskCreate event record.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadTaskSwitch)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam, uint32\_t creatingThread, uint32\_t generationNumber)**

*Callback for the ThreadTaskSwitch event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadTeamBegin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam)**

*Callback for the ThreadTeamBegin event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_ThreadTeamEnd)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam)**

*Callback for the ThreadTeamEnd event record.*
- **typedef OTF2\_CallbackCode(\* OTF2\_GlobalEvtReaderCallback\_Unknown)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList)**

*Callback for an unknown event record.*
- **typedef struct OTF2\_GlobalEvtReaderCallbacks\_struct OTF2\_GlobalEvtReaderCallbacks**

*Opaque struct which holds all event record callbacks.*

### **Functions**

- **void OTF2\_GlobalEvtReaderCallbacks\_Clear(OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks)**

*Clears a struct for the global event callbacks.*
- **void OTF2\_GlobalEvtReaderCallbacks\_Delete(OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks)**

*Deallocates a struct for the global event callbacks.*
- **OTF2\_GlobalEvtReaderCallbacks \* OTF2\_GlobalEvtReaderCallbacks\_New(void)**

*Allocates a new struct for the event callbacks.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetBufferFlushCallback(OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_BufferFlush bufferFlushCallback)**

*Registers the callback for the BufferFlush event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetEnterCallback(OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_Enter enterCallback)**

*Registers the callback for the Enter event.*

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetLeaveCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_Leave leaveCallback)**

*Registers the callback for the Leave event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMeasurementOnOffCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_MeasurementOnOff measurementOnOffCallback)**

*Registers the callback for the MeasurementOnOff event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMetricCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_Metric metricCallback)**

*Registers the callback for the Metric event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiCollectiveBeginCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_MpiCollectiveBegin mpiCollectiveBeginCallback)**

*Registers the callback for the MpiCollectiveBegin event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiCollectiveEndCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_MpiCollectiveEnd mpiCollectiveEndCallback)**

*Registers the callback for the MpiCollectiveEnd event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiIrecvCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_MpiIrecv mpiIrecvCallback)**

*Registers the callback for the MpiIrecv event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiIrecvRequestCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_MpiIrecvRequest mpiIrecvRequestCallback)**

*Registers the callback for the MpiIrecvRequest event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiIsendCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_MpiIsend mpiIsendCallback)**

*Registers the callback for the MpiIsend event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiIsendCompleteCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_MpiIsendComplete mpiIsendCompleteCallback)**

*Registers the callback for the MpiIsendComplete event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiRecvCallback (OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_MpiRecv mpiRecvCallback)**

*Registers the callback for the MpiRecv event.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMp IRequestCancelledCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_Mp IRequestCancelled** mpiRequestCancelledCallback)  
*Registers the callback for the Mp IRequestCancelled event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMp IRequestTestCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_Mp IRequestTest** mpiRequestTestCallback)  
*Registers the callback for the Mp IRequestTest event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMp SendCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_Mp Send** mpiSendCallback)  
*Registers the callback for the Mp Send event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpAcquireLockCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_OmpAcquireLock** ompAcquireLockCallback)  
*Registers the callback for the OmpAcquireLock event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpForkCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_OmpFork** ompForkCallback)  
*Registers the callback for the OmpFork event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpJoinCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_OmpJoin** ompJoinCallback)  
*Registers the callback for the OmpJoin event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpReleaseLockCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_OmpReleaseLock** ompReleaseLockCallback)  
*Registers the callback for the OmpReleaseLock event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskCompleteCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_OmpTaskComplete** ompTaskCompleteCallback)  
*Registers the callback for the OmpTaskComplete event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskCreateCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_OmpTaskCreate** ompTaskCreateCallback)  
*Registers the callback for the OmpTaskCreate event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskSwitchCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_OmpTaskSwitch** ompTaskSwitchCallback)  
*Registers the callback for the OmpTaskSwitch event.*

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetParameterIntCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ParameterInt** parameterIntCallback)  
  
*Registers the callback for the ParameterInt event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetParameterStringCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ParameterString** parameterStringCallback)  
  
*Registers the callback for the ParameterString event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetParameterUnsignedIntCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ParameterUnsignedInt** parameterUnsignedIntCallback)  
  
*Registers the callback for the ParameterUnsignedInt event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaAcquireLockCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaAcquireLock** rmaAcquireLockCallback)  
  
*Registers the callback for the RmaAcquireLock event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaAtomicCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaAtomic** rmaAtomicCallback)  
  
*Registers the callback for the RmaAtomic event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaCollectiveBeginCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaCollectiveBegin** rmaCollectiveBeginCallback)  
  
*Registers the callback for the RmaCollectiveBegin event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaCollectiveEndCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaCollectiveEnd** rmaCollectiveEndCallback)  
  
*Registers the callback for the RmaCollectiveEnd event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaGetCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaGet** rmaGetCallback)  
  
*Registers the callback for the RmaGet event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaGroupSyncCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaGroupSync** rmaGroupSyncCallback)  
  
*Registers the callback for the RmaGroupSync event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteBlockingCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteBlocking** rmaOpCompleteBlockingCallback)  
  
*Registers the callback for the RmaOpCompleteBlocking event.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteNonBlockingCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteNonBlocking** rmaOpCompleteNonBlockingCallback)  
*Registers the callback for the RmaOpCompleteNonBlocking event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteRemoteCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteRemote** rmaOpCompleteRemoteCallback)  
*Registers the callback for the RmaOpCompleteRemote event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpTestCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaOpTest** rmaOpTestCallback)  
*Registers the callback for the RmaOpTest event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaPutCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaPut** rmaPutCallback)  
*Registers the callback for the RmaPut event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaReleaseLockCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaReleaseLock** rmaReleaseLockCallback)  
*Registers the callback for the RmaReleaseLock event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaRequestLockCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaRequestLock** rmaRequestLockCallback)  
*Registers the callback for the RmaRequestLock event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaSyncCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaSync** rmaSyncCallback)  
*Registers the callback for the RmaSync event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaTryLockCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaTryLock** rmaTryLockCallback)  
*Registers the callback for the RmaTryLock event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaWaitChangeCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaWaitChange** rmaWaitChangeCallback)  
*Registers the callback for the RmaWaitChange event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaWinCreateCallback** (**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaWinCreate** rmaWinCreateCallback)  
*Registers the callback for the RmaWinCreate event.*

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaWinDestroyCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_RmaWinDestroy** rmaWinDestroyCallback)  
*Registers the callback for the RmaWinDestroy event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadAcquireLockCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadAcquireLock** threadAcquireLockCallback)  
*Registers the callback for the ThreadAcquireLock event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadForkCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadFork** threadForkCallback)  
*Registers the callback for the ThreadFork event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadJoinCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadJoin** threadJoinCallback)  
*Registers the callback for the ThreadJoin event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadReleaseLockCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadReleaseLock** threadReleaseLockCallback)  
*Registers the callback for the ThreadReleaseLock event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskCompleteCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadTaskComplete** threadTaskCompleteCallback)  
*Registers the callback for the ThreadTaskComplete event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskCreateCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadTaskCreate** threadTaskCreateCallback)  
*Registers the callback for the ThreadTaskCreate event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskSwitchCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadTaskSwitch** threadTaskSwitchCallback)  
*Registers the callback for the ThreadTaskSwitch event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadTeamBeginCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadTeamBegin** threadTeamBeginCallback)  
*Registers the callback for the ThreadTeamBegin event.*
- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadTeamEndCallback**  
(**OTF2\_GlobalEvtReaderCallbacks** \*globalEvtReaderCallbacks, **OTF2\_GlobalEvtReaderCallback\_ThreadTeamEnd** threadTeamEndCallback)  
*Registers the callback for the ThreadTeamEnd event.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetUnknownCallback  
(OTF2\_GlobalEvtReaderCallbacks \*globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_-  
Unknown unknownCallback)**  
*Registers the callback for unknown events.*

### **J.19.1 Detailed Description**

This defines the callbacks for the global event reader.

#### **Source Template:**

*templates/OTF2\_GlobalEvtReaderCallbacks.tmpl.h*

#### **Maintainer:**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>

#### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### **J.19.2 Typedef Documentation**

- **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

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.19.2.2** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
Enter)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void
*userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

Callback for the Enter event record.

An enter record indicates that the program enters a code region.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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*.

**J.19.2.3** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
Leave)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void
*userData, OTF2_AttributeList *attributeList, OTF2_RegionRef region)`

Callback for the Leave event record.

A leave record indicates that the program leaves a code region.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>attributeList</i>	Additional attributes for this event.
<i>region</i>	Needs to be defined in a definition record References a <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](#).

```
J.19.2.4 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-  
MeasurementOnOff)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, OTF2_MeasurementMode measurementMode)
```

Callback for the MeasurementOnOff event record.

This event signals where the measurement system turned measurement on or off.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_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](#).

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

```
J.19.2.5 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
Metric)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void  
*userData, OTF2_AttributeList *attributeList, OTF2_MetricRef metric,  
uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue  
*metricValues)
```

Callback for the Metric event record.

A metric event is always stored at the location that recorded the metric. A metric event can reference a metric class or metric instance. Therefore, metric classes and instances share same ID space. Synchronous metrics are always located right before the according enter and leave event.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.
<i>metricValues</i>	List of metric values.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.19.2.6 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
MpiCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp  
time, void *userData, OTF2_AttributeList *attributeList)
```

Callback for the MpiCollectiveBegin event record.

A MpiCollectiveBegin record marks the begin of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.).

## APPENDIX J. FILE DOCUMENTATION

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

J.19.2.7 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-MpiCollectiveEnd)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CollectiveOp collectiveOp, OTF2\_CommRef communicator, uint32\_t root, uint64\_t sizeSent, uint64\_t sizeReceived)**

Callback for the MpiCollectiveEnd event record.

A MpiCollectiveEnd record marks the end of an MPI collective operation (MPI\_GATHER, MPI\_SCATTER etc.). It keeps the necessary information for this event: type of collective operation, communicator, the root of this collective operation. You can optionally add further information like sent and received bytes.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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>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 <i>communicator</i> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.19.2.8 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
    MpIrecv)(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 MpIrecv event record.

A MpIrecv 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 <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
<i>requestID</i>	ID of the related request

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## APPENDIX J. FILE DOCUMENTATION

J.19.2.9 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-  
MpIrecvRequest)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t requestID)**

Callback for the MpIrecvRequest event record.

Signals the request of an receive, which can be completed later.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

J.19.2.10 **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,  
uint64\_t requestID)**

Callback for the MpIsend event record.

A MpIsend record indicates that a MPI message send process was initiated (MPI\_-ISEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

### 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.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>receiver</i>	MPI rank of receiver in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <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](#).

**J.19.2.11** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_MpiIsendComplete)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, uint64_t requestID)`

Callback for the MpiIsendComplete event record.

Signals the completion of non-blocking send request.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

J.19.2.12 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-  
MpRecv)(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 MpRecv event record.

A MpRecv record indicates that a MPI message was received (MPI\_RECV). It keeps the necessary information for this event: sender of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the receive buffer).

#### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 <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\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

J.19.2.13 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-  
MpRequestCancelled)(OTF2\_LocationRef locationID,  
OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList  
\*attributeList, uint64\_t requestId)**

Callback for the MpRequestCancelled event record.

This events appears if the program canceled a request.

#### Parameters

---

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.19.2.14** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-  
 MpRequestTest)(OTF2_LocationRef locationID, OTF2_TimeStamp  
 time, void *userData, OTF2_AttributeList *attributeList, uint64_t requestID)`

Callback for the MpRequestTest 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="#">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

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.19.2.15 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_MpiSend)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)**

Callback for the MpiSend event record.

A MpiSend record indicates that a MPI message send process was initiated (MPI\_SEND). It keeps the necessary information for this event: receiver of the message, communicator, and the message tag. You can optionally add further information like the message length (size of the send buffer).

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 <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\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

J.19.2.16 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_OmpAcquireLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t lockID, uint32\_t acquisitionOrder)**

Callback for the OmpAcquireLock event record.

An OmpAcquireLock record marks that a thread acquires an OpenMP lock.

This event record is superseded by the [ThreadAcquireLock](#) event record and should not be used when the [ThreadAcquireLock](#) event record is in use record.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.19.2.17** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-OmpFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, uint32_t numberOfRequestedThreads)`

Callback for the OmpFork event record.

An OmpFork record marks that an OpenMP Thread forks a thread team.

This event record is superseded by the [ThreadFork](#) event record and should not be used when the [ThreadFork](#) event record is in use.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>numberOfRequestedThreads</i>	Requested size of the team.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.0

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.19.2.18** **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-  
OmpJoin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void  
\*userData, OTF2\_AttributeList \*attributeList)**

Callback for the OmpJoin event record.

An OmpJoin record marks that a team of threads is joint and only the master thread continues execution.

This event record is superseded by the *ThreadJoin* event record and should not be used when the *ThreadJoin* event record is in use.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.

### **Since**

Version 1.0

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.19.2.19** **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-  
OmpReleaseLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
time, void \*userData, OTF2\_AttributeList \*attributeList, uint32\_t lockID,  
uint32\_t acquisitionOrder)**

Callback for the OmpReleaseLock event record.

An OmpReleaseLock record marks that a thread releases an OpenMP lock.

This event record is superseded by the *ThreadReleaseLock* event record and should not be used when the *ThreadReleaseLock* event record is in use.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.19.2.20** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-OmpTaskComplete)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, uint64_t taskID)`

Callback for the OmpTaskComplete event record.

An OmpTaskComplete record indicates that the execution of an OpenMP task has finished.

This event record is superseded by the [ThreadTaskComplete](#) event record and should not be used when the [ThreadTaskComplete](#) event record is in use.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 completed task instance.

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.19.2.21 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-  
OmpTaskCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)**

Callback for the OmpTaskCreate event record.

An OmpTaskCreate record marks that an OpenMP Task was/will be created in the current region.

This event record is superseded by the *ThreadTaskCreate* event record and should not be used when the *ThreadTaskCreate* event record is in use.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalEvtCallbacks</i> or <i>OTF2_GlobalEvtReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>taskID</i>	Identifier of the newly created task instance.

### Since

Version 1.0

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.19.2.22 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-  
OmpTaskSwitch)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
time, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t taskID)**

Callback for the OmpTaskSwitch event record.

An OmpTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

This event record is superseded by the *ThreadTaskSwitch* event record and should not be used when the *ThreadTaskSwitch* event record is in use.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.19.2.23 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp time,
void *userData, OTF2_AttributeList *attributeList, OTF2_ParameterRef
parameter, int64_t value)
```

Callback for the ParameterInt event record.

A ParameterInt record marks that in the current region, the specified integer parameter has the specified value.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## APPENDIX J. FILE DOCUMENTATION

J.19.2.24 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-ParameterString)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, OTF2StringRef string)**

Callback for the ParameterString event record.

A ParameterString record marks that in the current region, the specified string parameter has the specified value.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

J.19.2.25 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-ParameterUnsignedInt)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_ParameterRef parameter, uint64\_t value)**

Callback for the ParameterUnsignedInt event record.

A ParameterUnsignedInt record marks that in the current region, the specified unsigned integer parameter has the specified value.

### Parameters

*locationID* The location where this event happened.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

### Since

Version 1.0

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.19.2.26 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
RmaAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp
time, void *userData, OTF2_AttributeList *attributeList,
OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType
lockType)
```

Callback for the RmaAcquireLock event record.

An RmaAcquireLock record denotes the time a lock was aquired by the process.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.19.2.27 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-RmaAtomic)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint32\_t remote, OTF2\_RmaAtomicType type, uint64\_t bytesSent, uint64\_t bytesReceived, uint64\_t matchingId)**

Callback for the RmaAtomic event record.

An RmaAtomic record denotes the time a atomic operation was issued.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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*.

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

```
J.19.2.28 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
          RmaCollectiveBegin)(OTF2_LocationRef locationID,  
          OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
          *attributeList)
```

Callback for the RmaCollectiveBegin event record.

An RmaCollectiveBegin record denotes the beginnig of a collective RMA operation.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

```
J.19.2.29 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_  
          RmaCollectiveEnd)(OTF2_LocationRef locationID,  
          OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
          *attributeList, OTF2_CollectiveOp collectiveOp, OTF2_RmaSyncLevel  
          syncLevel, OTF2_RmaWinRef win, uint32_t root, uint64_t bytesSent, uint64_t  
          bytesReceived)
```

Callback for the RmaCollectiveEnd event record.

"An RmaCollectiveEnd record denotes the end of a collective RMA operation.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

## APPENDIX J. FILE DOCUMENTATION

---

<i>syncLevel</i>	Synchronization level of this collective operation.
<i>win</i>	ID of the window used for this operation. References a <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](#).

```
J.19.2.30 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
    RmaGet)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void
    *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win,
    uint32_t remote, uint64_t bytes, uint64_t matchingId)
```

Callback for the RmaGet event record.

An RmaGet record denotes the time a put operation was issued.

**Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 received from target.
<i>matchingId</i>	ID used for matching the appropriate completion record.

**Since**

Version 1.2

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.19.2.31 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-RmaGroupSync)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaSyncLevel syncLevel, OTF2\_RmaWinRef win, OTF2\_GroupRef group)**

Callback for the RmaGroupSync event record.

An RmaGroupSync record denotes the synchronization with a subgroup of processes on a window.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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*.

J.19.2.32 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-RmaOpCompleteBlocking)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win, uint64\_t matchingId)**

Callback for the RmaOpCompleteBlocking event record.

## **APPENDIX J. FILE DOCUMENTATION**

---

An RmaOpCompleteBlocking record denotes the local completion of a blocking RMA operation.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

**J.19.2.33** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-RmaOpCompleteNonBlocking)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, OTF2_RmaWinRef win, uint64_t matchingId)`

Callback for the RmaOpCompleteNonBlocking event record.

An RmaOpCompleteNonBlocking record denotes the local completion of a non-blocking RMA operation.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Since**

Version 1.2

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.19.2.34 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_
- RmaOpCompleteRemote)(OTF2_LocationRef locationID,
OTF2_TimeStamp time, void *userData, OTF2_AttributeList
*attributeList, OTF2_RmaWinRef win, uint64_t matchingId)
```

Callback for the RmaOpCompleteRemote event record.

An RmaOpCompleteRemote record denotes the local completion of an RMA operation.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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*.

```
J.19.2.35 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_
- RmaOpTest)(OTF2_LocationRef locationID, OTF2_TimeStamp time,
void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef
win, uint64_t matchingId)
```

Callback for the RmaOpTest event record.

## **APPENDIX J. FILE DOCUMENTATION**

---

An RmaOpTest record denotes that a non-blocking RMA operation has been tested for completion unsuccessfully.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

**J.19.2.36** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-RmaPut)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t bytes, uint64_t matchingId)`

Callback for the RmaPut event record.

An RmaPut record denotes the time a put operation was issued.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.19.2.37 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
    RmaReleaseLock)(OTF2_LocationRef locationID, OTF2_TimeStamp
    time, void *userData, OTF2_AttributeList *attributeList,
    OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId)
```

Callback for the RmaReleaseLock event record.

An RmaReleaseLock record denotes the time the lock was released.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 released, if multiple locks are defined on a window.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.19.2.38 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
    RmaRequestLock)(OTF2_LocationRef locationID, OTF2_TimeStamp
    time, void *userData, OTF2_AttributeList *attributeList,
    OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType
    lockType)
```

Callback for the RmaRequestLock event record.

## **APPENDIX J. FILE DOCUMENTATION**

---

An RmaRequestLock record denotes the time a lock was requested and with it the earliest time it could have been granted. It is used to mark (possibly) non-blocking lock request, as defined by the MPI standard.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 aquired, if multiple locks are defined on a window.
<i>lockType</i>	Type of lock aquired.

### **Since**

Version 1.2

### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.19.2.39** **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.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>remote</i>	Rank of the locked remote process.
<i>syncType</i>	Type of synchronization.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.19.2.40** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-RmaTryLock)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_RmaWinRef win, uint32_t remote, uint64_t lockId, OTF2_LockType lockType)`

Callback for the RmaTryLock event record.

An RmaTryLock record denotes the time of an unsuccessful attempt to acquire the lock.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 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 J. FILE DOCUMENTATION

J.19.2.41 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-RmaWaitChange)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)**

Callback for the RmaWaitChange event record.

An RmaWaitChange record denotes the change of a window that was waited for.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

J.19.2.42 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-RmaWinCreate)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_RmaWinRef win)**

Callback for the RmaWinCreate event record.

An RmaWinCreate record denotes the creation of an RMA window.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.19.2.43 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
RmaWinDestroy)(OTF2_LocationRef locationID, OTF2_TimeStamp
time, void *userData, OTF2_AttributeList *attributeList,
OTF2_RmaWinRef win)
```

Callback for the RmaWinDestroy event record.

An RmaWinDestroy record denotes the destruction of an RMA window.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 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*.

```
J.19.2.44 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
ThreadAcquireLock)(OTF2_LocationRef locationID,
OTF2_TimeStamp time, void *userData, OTF2_AttributeList
*attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t
acquisitionOrder)
```

Callback for the ThreadAcquireLock event record.

An ThreadAcquireLock record marks that a thread acquires an lock.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

### **Since**

Version 1.2

### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.19.2.45** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_ - ThreadFork)(OTF2_LocationRef locationID, OTF2_TimeStamp time, void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm model, uint32_t numberOfRequestedThreads)`

Callback for the ThreadFork event record.

An ThreadFork record marks that an thread forks a thread team.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.
<i>numberOfRequestedThreads</i>	Requested size of the team.

### **Since**

Version 1.2

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.19.2.46** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
ThreadJoin)(OTF2_LocationRef locationID, OTF2_TimeStamp time,
void *userData, OTF2_AttributeList *attributeList, OTF2_Paradigm
model)`

Callback for the ThreadJoin event record.

An ThreadJoin record marks that a team of threads is joint and only the master thread continues execution.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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*.

**J.19.2.47** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
ThreadReleaseLock)(OTF2_LocationRef locationID,
OTF2_TimeStamp time, void *userData, OTF2_AttributeList
*attributeList, OTF2_Paradigm model, uint32_t lockID, uint32_t
acquisitionOrder)`

Callback for the ThreadReleaseLock event record.

An ThreadReleaseLock record marks that a thread releases an lock.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.

## APPENDIX J. FILE DOCUMENTATION

---

<i>userData</i>	User data as set by <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.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.19.2.48** `typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-ThreadTaskComplete)(OTF2_LocationRef locationID,  
OTF2_TimeStamp time, void *userData, OTF2_AttributeList  
*attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t  
generationNumber)`

Callback for the ThreadTaskComplete event record.

An ThreadTaskComplete record indicates that the execution of an OpenMP task has finished.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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.

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Since**

Version 1.2

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.19.2.49 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
ThreadTaskCreate)(OTF2_LocationRef locationID,
OTF2_TimeStamp time, void *userData, OTF2_AttributeList
*attributeList, OTF2_CommRef threadTeam, uint32_t creatingThread, uint32_t
generationNumber)
```

Callback for the ThreadTaskCreate event record.

An ThreadTaskCreate record marks that an task in was/will be created and will be processed by the specified thread team.

### **Parameters**

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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. (This is redundant, remove?)
<i>generationNumber</i>	Thread-private generation number of task's creating thread.

### **Since**

Version 1.2

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## APPENDIX J. FILE DOCUMENTATION

J.19.2.50 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-ThreadTaskSwitch)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam, uint32\_t creatingThread, uint32\_t generationNumber)**

Callback for the ThreadTaskSwitch event record.

An ThreadTaskSwitch record indicates that the execution of the current task will be suspended and another task starts/restarts its execution. Please note that this may change the current call stack of the executing location.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

J.19.2.51 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-ThreadTeamBegin)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_CommRef threadTeam)**

Callback for the ThreadTeamBegin event record.

### Parameters

<i>locationID</i>	The location where this event happened.
-------------------	---

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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](#).

```
J.19.2.52 typedef OTF2_CallbackCode( * OTF2_GlobalEvtReaderCallback_-
    ThreadTeamEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp
    time, void *userData, OTF2_AttributeList *attributeList,
    OTF2_CommRef threadTeam)
```

Callback for the ThreadTeamEnd event record.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <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 J. FILE DOCUMENTATION

J.19.2.53 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalEvtReaderCallback\_-Unknown)(OTF2\_LocationRef locationID, OTF2\_TimeStamp time, void \*userData, OTF2\_AttributeList \*attributeList)**

Callback for an unknown event record.

### Parameters

<i>locationID</i>	The location where this event happened.
<i>time</i>	The time when this event happened.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalEvtCallbacks</a> or <a href="#">OTF2_GlobalEvtReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## J.19.3 Function Documentation

J.19.3.1 **void OTF2\_GlobalEvtReaderCallbacks\_Clear ( OTF2\_-GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks* )**

Clears a struct for the global event callbacks.

### Parameters

<i>globalEvtReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_-GlobalEvtReaderCallbacks_New</a> .
---------------------------------	---

J.19.3.2 **void OTF2\_GlobalEvtReaderCallbacks\_Delete ( OTF2\_-GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks* )**

Deallocates a struct for the global event callbacks.

### Parameters

<i>globalEvtReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_-GlobalEvtReaderCallbacks_New</a> .
---------------------------------	---

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **J.19.3.3 OTF2\_GlobalEvtReaderCallbacks\* OTF2\_GlobalEvtReaderCallbacks\_New ( void )**

Allocates a new struct for the event callbacks.

#### **Returns**

A newly allocated struct of type [OTF2\\_GlobalEvtReaderCallbacks](#).

### **J.19.3.4 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetBufferFlushCallback ( OTF2\_GlobalEvtReaderCallbacks \* globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_BufferFlush bufferFlushCallback )**

Registers the callback for the BufferFlush event.

#### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>bufferFlushCallback</i>	Function which should be called for all BufferFlush events.

#### **Returns**

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid `defReaderCallbacks` argument

### **J.19.3.5 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetEnterCallback ( OTF2\_GlobalEvtReaderCallbacks \* globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_Enter enterCallback )**

Registers the callback for the Enter event.

#### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter events.

## APPENDIX J. FILE DOCUMENTATION

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.6 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetLeaveCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_Leave *leaveCallback* )**

Registers the callback for the Leave event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>leaveCallback</i>	Function which should be called for all Leave events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.7 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-SetMeasurementOnOffCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-MeasurementOnOff *measurementOnOffCallback* )**

Registers the callback for the MeasurementOnOff event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>measurementOnOff-Callback</i>	Function which should be called for all MeasurementOnOff events.

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.8 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMetricCallback  
( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*,  
OTF2\_GlobalEvtReaderCallback\_Metric *metricCallback* )**

Registers the callback for the Metric event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.9 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiCollectiveBeginCallback ( OTF2\_GlobalEvtReaderCallbacks  
\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-  
MpiCollectiveBegin *mpiCollectiveBeginCallback*  
)**

Registers the callback for the MpiCollectiveBegin event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBegin-Callback</i>	Function which should be called for all MpiCollectiveBegin events.

## APPENDIX J. FILE DOCUMENTATION

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.10 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiCollectiveEndCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_MpiCollectiveEnd *mpiCollectiveEndCallback* )**

Registers the callback for the MpiCollectiveEnd event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveEnd-Callback</i>	Function which should be called for all MpiCollectiveEnd events.

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.11 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpilrecvCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_MpiIrecv *mpilrecvCallback* )**

Registers the callback for the Mpilrecv event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpilrecv-Callback</i>	Function which should be called for all Mpilrecv events.

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

### **J.19.3.12 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiRecvRequestCallback ( OTF2\_GlobalEvtReaderCallbacks \***

***globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_MpiRecvRequest *mpirecvRequestCallback* )**

Registers the callback for the MpiRecvRequest event.

### **Parameters**

<b><i>glob-alEvtReaderCallbacks</i></b>	Struct for all callbacks.
<b><i>mpirecvRequestCallback</i></b>	Function which should be called for all MpiRecvRequest events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

### **J.19.3.13 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpIsendCallback ( OTF2\_GlobalEvtReaderCallbacks \***

***globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_MpiIsend *mpisendCallback* )**

Registers the callback for the MpIsend event.

### **Parameters**

<b><i>glob-alEvtReaderCallbacks</i></b>	Struct for all callbacks.
<b><i>mpisendCallback</i></b>	Function which should be called for all MpIsend events.

## APPENDIX J. FILE DOCUMENTATION

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

J.19.3.14 **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpIsendCompleteCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_MpiIsendComplete *mpIsendCompleteCallback* )**

Registers the callback for the MpiIsendComplete event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpIsendCompleteCallback</i>	Function which should be called for all MpiIsendComplete events.

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

J.19.3.15 **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpIRecvCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_MpiIRecv *mpiIRecvCallback* )**

Registers the callback for the MpiIRecv event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIRecvCallback</i>	Function which should be called for all MpiIRecv events.

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.16 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiRequestCancelledCallback ( OTF2\_GlobalEvtReaderCallbacks  
\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-  
MpiRequestCancelled *mpiRequestCancelledCallback*  
)**

Registers the callback for the MpiRequestCancelled event.

### **Parameters**

<i>glob-</i> <i>alEvtRead-</i> <i>erCallbacks</i>	Struct for all callbacks.
<i>mpiRe-</i> <i>questCan-</i> <i>celledCall-</i> <i>back</i>	Function which should be called for all MpiRequestCancelled events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.17 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetMpiRequestTestCallback ( OTF2\_GlobalEvtReaderCallbacks \*  
globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_-  
MpiRequestTest *mpiRequestTestCallback* )**

Registers the callback for the MpiRequestTest event.

### **Parameters**

<i>glob-</i> <i>alEvtRead-</i> <i>erCallbacks</i>	Struct for all callbacks.
---	---------------------------

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>mpiRequestTestCallback</i>	Function which should be called for all MpiRequestTest events.
-------------------------------	--

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.18 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetMpiSendCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_MpiSendCallback *mpiSendCallback* )**

Registers the callback for the MpiSend event.

### **Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSendCallback</i>	Function which should be called for all MpiSend events.

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.19 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpAcquireLockCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_OmpAcquireLock *ompAcquireLockCallback* )**

Registers the callback for the OmpAcquireLock event.

### **Parameters**

<i>globalEvtReaderCallbacks</i>	Struct for all callbacks.
---------------------------------	---------------------------

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>ompAcquireLockCallback</i>	Function which should be called for all OmpAcquireLock events.
-------------------------------	--

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.20 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpForkCallback**  
( `OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks,`  
`OTF2_GlobalEvtReaderCallback_OmpFork ompForkCallback` )

Registers the callback for the OmpFork event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompForkCallback</i>	Function which should be called for all OmpFork events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.21 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpJoinCallback**  
( `OTF2_GlobalEvtReaderCallbacks * globalEvtReaderCallbacks,`  
`OTF2_GlobalEvtReaderCallback_OmpJoin ompJoinCallback` )

Registers the callback for the OmpJoin event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
----------------------------------	---------------------------

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>ompJoin-Callback</i>	Function which should be called for all OmpJoin events.
-------------------------	---

### **Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**J.19.3.22 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpReleaseLockCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ompReleaseLock *ompReleaseLockCallback* )**

Registers the callback for the OmpReleaseLock event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>ompReleaseLockCallback</i>	Function which should be called for all OmpReleaseLock events.

### **Returns**

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid `defReaderCallbacks` argument

**J.19.3.23 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskCompleteCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ompTaskComplete *ompTaskCompleteCallback* )**

Registers the callback for the OmpTaskComplete event.

### **Parameters**

---

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskCompleteCallback</i>	Function which should be called for all OmpTaskComplete events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.24 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskCreateCallback ( OTF2\_GlobalEvtReaderCallbacks \* globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_OmpTaskCreate ompTaskCreateCallback )**

Registers the callback for the OmpTaskCreate event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskCreateCallback</i>	Function which should be called for all OmpTaskCreate events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.19.3.25 **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskSwitchCallback ( OTF2\_GlobalEvtReaderCallbacks \*  
globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_  
OmpTaskSwitch *ompTaskSwitchCallback* )**

Registers the callback for the OmpTaskSwitch event.

### Parameters

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
<i>omp- TaskSwitch- Callback</i>	Function which should be called for all OmpTaskSwitch events.

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

J.19.3.26 **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetParameterIntCallback  
( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*,  
OTF2\_GlobalEvtReaderCallback\_ParameterInt *parameterIntCallback*  
)**

Registers the callback for the ParameterInt event.

### Parameters

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
<i>parameter- IntCallback</i>	Function which should be called for all ParameterInt events.

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

**J.19.3.27 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetParameterStringCallback ( OTF2\_GlobalEvtReaderCallbacks \*  
globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_-  
ParameterString parameterStringCallback )**

Registers the callback for the ParameterString event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterStringCallback</i>	Function which should be called for all ParameterString events.

### **Returns**

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

**J.19.3.28 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetParameterUnsignedIntCallback ( OTF2\_GlobalEvtReaderCallbacks  
\* globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_-  
ParameterUnsignedInt parameterUnsignedIntCallback  
)**

Registers the callback for the ParameterUnsignedInt event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt events.

### **Returns**

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks

## APPENDIX J. FILE DOCUMENTATION

argument

**J.19.3.29 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaAcquireLockCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_RmaAcquireLock *rmaAcquireLockCallback* )**

Registers the callback for the RmaAcquireLock event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaAcquireLockCallback</i>	Function which should be called for all RmaAcquireLock events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**J.19.3.30 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaAtomicCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_RmaAtomic *rmaAtomicCallback* )**

Registers the callback for the RmaAtomic event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaAtomicCallback</i>	Function which should be called for all RmaAtomic events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks*

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

argument

```
J.19.3.31 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-
SetRmaCollectiveBeginCallback ( OTF2_GlobalEvtReaderCallbacks_
* globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-
RmaCollectiveBegin rmaCollectiveBeginCallback
)
```

Registers the callback for the RmaCollectiveBegin event.

### Parameters

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
<i>rmaCollec- tiveBegin- Callback</i>	Function which should be called for all RmaCollectiveBegin events.

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

```
J.19.3.32 OTF2_ErrorCode OTF2_GlobalEvtReaderCallbacks_-
SetRmaCollectiveEndCallback ( OTF2_GlobalEvtReaderCallbacks_
* globalEvtReaderCallbacks, OTF2_GlobalEvtReaderCallback_-
RmaCollectiveEnd rmaCollectiveEndCallback )
```

Registers the callback for the RmaCollectiveEnd event.

### Parameters

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
<i>rmaCollec- tiveEnd- Callback</i>	Function which should be called for all RmaCollectiveEnd events.

## APPENDIX J. FILE DOCUMENTATION

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.33 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaGetCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_RmaGet *rmaGetCallback* )**

Registers the callback for the RmaGet event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaGetCallback</i>	Function which should be called for all RmaGet events.

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.34 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaGroupSyncCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_RmaGroupSync *rmaGroupSyncCallback* )**

Registers the callback for the RmaGroupSync event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaGroupSyncCallback</i>	Function which should be called for all RmaGroupSync events.

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.35 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaOpCompleteBlockingCallback ( OTF2\_GlobalEvtReaderCallbacks  
\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-  
RmaOpCompleteBlocking *rmaOpCompleteBlockingCallback*  
)**

Registers the callback for the RmaOpCompleteBlocking event.

### **Parameters**

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
<i>rmaOp- Complete- Blocking- Callback</i>	Function which should be called for all RmaOpCompleteBlocking events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.36 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaOpCompleteNonBlockingCallback ( OTF2\_-  
GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*,  
OTF2\_GlobalEvtReaderCallback\_RmaOpCompleteNonBlocking  
*rmaOpCompleteNonBlockingCallback* )**

Registers the callback for the RmaOpCompleteNonBlocking event.

### **Parameters**

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
---	---------------------------

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>rmaOp- Com- pleteNon- Blocking- Callback</i>	Function which should be called for all RmaOpCompleteNonBlocking events.
---	--

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

**J.19.3.37 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaOpCompleteRemoteCallback ( OTF2\_GlobalEvtReaderCallbacks  
\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-  
RmaOpCompleteRemote *rmaOpCompleteRemoteCallback*  
)**

Registers the callback for the RmaOpCompleteRemote event.

### **Parameters**

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
<i>rmaOp- CompleteR- emoteCall- back</i>	Function which should be called for all RmaOpCompleteRemote events.

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

**J.19.3.38 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpTestCallback  
( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*,  
OTF2\_GlobalEvtReaderCallback\_RmaOpTest *rmaOpTestCallback* )**

Registers the callback for the RmaOpTest event.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaOpTest-Callback</i>	Function which should be called for all RmaOpTest events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.39 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaPutCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback *rmaPutCallback* )**

Registers the callback for the RmaPut event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaPut-Callback</i>	Function which should be called for all RmaPut events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.40 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-SetRmaReleaseLockCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-RmaReleaseLock *rmaReleaseLockCallback* )**

Registers the callback for the RmaReleaseLock event.

### Parameters

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaReleaseLockCallback</i>	Function which should be called for all RmaReleaseLock events.

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.41 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaRequestLockCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_RmaRequestLock *rmaRequestLockCallback* )**

Registers the callback for the RmaRequestLock event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaRequestLockCallback</i>	Function which should be called for all RmaRequestLock events.

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.19.3.42 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaSyncCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_RmaSync *rmaSyncCallback* )**

Registers the callback for the RmaSync event.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaSyncCallback</i>	Function which should be called for all RmaSync events.

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

### J.19.3.43 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetRmaTryLockCallback

( *OTF2\_GlobalEvtReaderCallbacks* \* *globalEvtReaderCallbacks*,  
*OTF2\_GlobalEvtReaderCallback\_RmaTryLock rmaTryLockCallback* )

Registers the callback for the RmaTryLock event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaTryLockCallback</i>	Function which should be called for all RmaTryLock events.

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INVALID\_ARGUMENT* for an invalid *defReaderCallbacks* argument

### J.19.3.44 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-

*SetRmaWaitChangeCallback* ( *OTF2\_GlobalEvtReaderCallbacks*\* *globalEvtReaderCallbacks*, *OTF2\_GlobalEvtReaderCallback\_-RmaWaitChange rmaWaitChangeCallback* )

Registers the callback for the RmaWaitChange event.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWaitChangeCallback</i>	Function which should be called for all RmaWaitChange events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.45 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaWinCreateCallback ( OTF2\_GlobalEvtReaderCallbacks \*  
globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_-  
RmaWinCreate rmaWinCreateCallback )**

Registers the callback for the RmaWinCreate event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWinCreateCallback</i>	Function which should be called for all RmaWinCreate events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.46 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetRmaWinDestroyCallback ( OTF2\_GlobalEvtReaderCallbacks \*  
globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_-  
RmaWinDestroy rmaWinDestroyCallback )**

Registers the callback for the RmaWinDestroy event.

## J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference

---

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>rmaWinDestroyCallback</i>	Function which should be called for all RmaWinDestroy events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**J.19.3.47** **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadAcquireLockCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ThreadAcquireLock *threadAcquireLockCallback* )**

Registers the callback for the ThreadAcquireLock event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadAcquireLockCallback</i>	Function which should be called for all ThreadAcquireLock events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**J.19.3.48** **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadForkCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ThreadFork *threadForkCallback* )**

Registers the callback for the ThreadFork event.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadFork-Callback</i>	Function which should be called for all ThreadFork events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid *defReaderCallbacks* argument

**J.19.3.49 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadJoinCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ThreadJoin *threadJoinCallback* )**

Registers the callback for the ThreadJoin event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadJoin-Callback</i>	Function which should be called for all ThreadJoin events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid *defReaderCallbacks* argument

**J.19.3.50 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadReleaseLockCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ThreadReleaseLock *threadReleaseLockCallback* )**

Registers the callback for the ThreadReleaseLock event.

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>thread-Release-LockCallback</i>	Function which should be called for all ThreadReleaseLock events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.19.3.51 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-SetThreadTaskCompleteCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-ThreadTaskComplete *threadTaskCompleteCallback* )**

Registers the callback for the ThreadTaskComplete event.

### **Parameters**

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>thread-TaskCompleteCallback</i>	Function which should be called for all ThreadTaskComplete events.

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

J.19.3.52 **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskCreateCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ ThreadTaskCreate *threadTaskCreateCallback* )**

Registers the callback for the ThreadTaskCreate event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTaskCreateCallback</i>	Function which should be called for all ThreadTaskCreate events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

J.19.3.53 **OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskSwitchCallback ( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_ ThreadTaskSwitch *threadTaskSwitchCallback* )**

Registers the callback for the ThreadTaskSwitch event.

### Parameters

<i>glob-alEvtReaderCallbacks</i>	Struct for all callbacks.
<i>threadTaskSwitchCallback</i>	Function which should be called for all ThreadTaskSwitch events.

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

## **J.19 OTF2\_GlobalEvtReaderCallbacks.h File Reference**

---

**J.19.3.54 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadTeamBeginCallback ( OTF2\_GlobalEvtReaderCallbacks  
\* *globalEvtReaderCallbacks*, OTF2\_GlobalEvtReaderCallback\_-  
ThreadTeamBegin *threadTeamBeginCallback* )**

Registers the callback for the ThreadTeamBegin event.

### **Parameters**

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
<i>thread- TeamBegin- Callback</i>	Function which should be called for all ThreadTeamBegin events.

### **Returns**

***OTF2\_SUCCESS*** if successful  
***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid *defReaderCallbacks* argument

**J.19.3.55 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_-  
SetThreadTeamEndCallback ( OTF2\_GlobalEvtReaderCallbacks \*  
globalEvtReaderCallbacks, OTF2\_GlobalEvtReaderCallback\_-  
ThreadTeamEnd *threadTeamEndCallback* )**

Registers the callback for the ThreadTeamEnd event.

### **Parameters**

<i>glob- alEvtRead- erCallbacks</i>	Struct for all callbacks.
<i>threadTea- mEndCall- back</i>	Function which should be called for all ThreadTeamEnd events.

### **Returns**

***OTF2\_SUCCESS*** if successful  
***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid *defReaderCallbacks* argument

## APPENDIX J. FILE DOCUMENTATION

**J.19.3.56 OTF2\_ErrorCode OTF2\_GlobalEvtReaderCallbacks\_SetUnknownCallback  
( OTF2\_GlobalEvtReaderCallbacks \* *globalEvtReaderCallbacks*,  
OTF2\_GlobalEvtReaderCallback\_Unknown *unknownCallback* )**

Registers the callback for unknown events.

### Parameters

<i>glob-alEvtReaderCallbacks</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

## **J.20 OTF2\_GlobalSnapReader.h File Reference**

This is the global snapshot event reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_SnapReader.h>
#include <otf2/OTF2_GlobalSnapReaderCallbacks.h>
```

### Functions

- **OTF2\_ErrorCode OTF2\_GlobalSnapReader\_ReadSnapshots (OTF2\_GlobalSnapReader \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)**

*Reads the given number of records from the global snap event reader.*

- **OTF2\_ErrorCode OTF2\_GlobalSnapReader\_SetCallbacks (OTF2\_GlobalSnapReader \*reader, const OTF2\_GlobalSnapReaderCallbacks \*callbacks, void \*userData)**

*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

## J.20 OTF2\_GlobalSnapReader.h File Reference

---

### J.20.1 Detailed Description

This is the global snapshot event reader.

#### Since

Version 1.2

Used to read from multiple local snap event readers, and provide them in a timely ordered sequence.

### J.20.2 Function Documentation

**J.20.2.1 OTF2\_ErrorCode OTF2\_GlobalSnapReader\_ReadSnapshots (**  
**OTF2\_GlobalSnapReader \* reader, uint64\_t recordsToRead, uint64\_t \***  
**recordsRead )**

Reads the given number of records from the global snap event reader.

#### Parameters

	<i>reader</i>	The records of this reader will be read when the function is issued.
	<i>recordsToRead</i>	This variable tells the reader how much records it has to read.
out	<i>recordsRead</i>	This is a pointer to variable where the amount of actually read records is returned. This may differ to the given recordsToRead if there are no more records left in the trace. In this case the programmer can easily check that the reader has finished his job by checking recordsRead < recordsToRead.

#### Since

Version 1.2

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.20.2.2 OTF2\_ErrorCode OTF2\_GlobalSnapReader\_SetCallbacks ( OTF2\_-**  
**GlobalSnapReader \* reader, const OTF2\_GlobalSnapReaderCallbacks**  
**\* callbacks, void \* userData )**

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function.

## **APPENDIX J. FILE DOCUMENTATION**

---

Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

### **Parameters**

<i>reader</i>	Reader object which reads the snap events from its buffer.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#">OTF2_GlobalSnapReaderCallbacks_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.

## **J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference**

This defines the callbacks for the global snap reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

### **Typedefs**

- **typedef** [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalSnapReaderCallback\\_Enter](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_RegionRef](#) region)
- Callback for the Enter snap record.*
- **typedef** [OTF2\\_CallbackCode](#)(\* [OTF2\\_GlobalSnapReaderCallback\\_MeasurementOnOff](#))([OTF2\\_LocationRef](#) locationID, [OTF2\\_TimeStamp](#) snapTime, void \*userData, [OTF2\\_AttributeList](#) \*attributeList, [OTF2\\_TimeStamp](#) origEventTime, [OTF2\\_MeasurementMode](#) measurementMode)
- Callback for the MeasurementOnOff snap record.*

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_Metric )(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)`

*Callback for the Metric snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_MpiCollectiveBegin )(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime)`

*Callback for the MpiCollectiveBegin snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_MpiCollectiveEnd )(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)`

*Callback for the MpiCollectiveEnd snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_MpiIrecv )(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msLength, uint64_t requestID)`

*Callback for the MpiIrecv snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_MpiIrecvRequest )(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)`

*Callback for the MpiIrecvRequest snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_MpiIsend )(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msLength, uint64_t requestID)`

*Callback for the MpiIsend snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_MpiIsendComplete )(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)`

*Callback for the MpiIsendComplete snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_MpiRecv )(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData,`

---

## APPENDIX J. FILE DOCUMENTATION

`OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

*Callback for the MpiRecv snap record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_MpiSend)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

*Callback for the MpiSend snap record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_OmpAcquireLock)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t lockID, uint32_t acquisitionOrder)`

*Callback for the OmpAcquireLock snap record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_OmpFork)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t numberOfRequestedThreads)`

*Callback for the OmpFork snap record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t taskID)`

*Callback for the OmpTaskCreate snap record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_OmpTaskSwitch)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t taskID)`

*Callback for the OmpTaskSwitch snap record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_ParameterInt)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, int64_t value)`

*Callback for the ParameterInt snap record.*

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_ParameterString)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, OTF2StringRef string)`

*Callback for the ParameterString snap record.*

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_ParameterUnsignedInt)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, uint64_t value)`

*Callback for the ParameterUnsignedInt snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_SnapshotEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t contReadPos)`

*Callback for the SnapshotEnd snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_SnapshotStart)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t numberOfRecords)`

*Callback for the SnapshotStart snap record.*
- `typedef OTF2_CallbackCode(* OTF2_GlobalSnapReaderCallback_Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList)`

*Callback for an unknown snap record.*
- `typedef struct OTF2_GlobalSnapReaderCallbacks_struct OTF2_GlobalSnapReaderCallbacks`

*Opaque struct which holds all snap record callbacks.*

## Functions

- `void OTF2_GlobalSnapReaderCallbacks_Clear (OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks)`

*Clears a struct for the global snap callbacks.*
- `void OTF2_GlobalSnapReaderCallbacks_Delete (OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks)`

*Deallocates a struct for the global snap callbacks.*
- `OTF2_GlobalSnapReaderCallbacks * OTF2_GlobalSnapReaderCallbacks_New (void)`

*Allocates a new struct for the snap callbacks.*
- `OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetEnterCallback (OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_Enter enterCallback)`

*Registers the callback for the Enter snap.*
- `OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMeasurementOnOffCallback (OTF2_GlobalSnapReaderCallbacks *globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_MeasurementOnOff measurementOnOffCallback)`

## **APPENDIX J. FILE DOCUMENTATION**

---

*Registers the callback for the MeasurementOnOff snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMetricCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_Metric](#) metricCallback)

*Registers the callback for the Metric snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiCollectiveBeginCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiCollectiveBegin](#) mpiCollectiveBeginCallback)

*Registers the callback for the MpiCollectiveBegin snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiCollectiveEndCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiCollectiveEnd](#) mpiCollectiveEndCallback)

*Registers the callback for the MpiCollectiveEnd snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiIrecvCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiIrecv](#) mpiIrecvCallback)

*Registers the callback for the MpiIrecv snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiIrecvRequestCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiIrecvRequest](#) mpiIrecvRequestCallback)

*Registers the callback for the MpiIrecvRequest snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiIsendCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiIsend](#) mpiIsendCallback)

*Registers the callback for the MpiIsend snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiIsendCompleteCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiIsendComplete](#) mpiIsendCompleteCallback)

*Registers the callback for the MpiIsendComplete snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiRecvCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiRecv](#) mpiRecvCallback)

*Registers the callback for the MpiRecv snap.*

- [OTF2\\_ErrorCode OTF2\\_GlobalSnapReaderCallbacks\\_SetMpiSendCallback](#) ([OTF2\\_GlobalSnapReaderCallbacks](#) \*globalSnapReaderCallbacks, [OTF2\\_GlobalSnapReaderCallback\\_MpiSend](#) mpiSendCallback)

*Registers the callback for the MpiSend snap.*

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetOmpAcquireLockCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_OmpAcquireLock** ompAcquireLockCallback)  
*Registers the callback for the OmpAcquireLock snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetOmpForkCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_OmpFork** ompForkCallback)  
*Registers the callback for the OmpFork snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetOmpTaskCreateCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_OmpTaskCreate** ompTaskCreateCallback)  
*Registers the callback for the OmpTaskCreate snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetOmpTaskSwitchCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_OmpTaskSwitch** ompTaskSwitchCallback)  
*Registers the callback for the OmpTaskSwitch snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetParameterIntCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_ParameterInt** parameterIntCallback)  
*Registers the callback for the ParameterInt snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetParameterStringCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_ParameterString** parameterStringCallback)  
*Registers the callback for the ParameterString snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetParameterUnsignedIntCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_ParameterUnsignedInt** parameterUnsignedIntCallback)  
*Registers the callback for the ParameterUnsignedInt snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetSnapshotEndCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_SnapshotEnd** snapshotEndCallback)  
*Registers the callback for the SnapshotEnd snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetSnapshotStartCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_SnapshotStart** snapshotStartCallback)  
*Registers the callback for the SnapshotStart snap.*
- **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetUnknownCallback**  
(**OTF2\_GlobalSnapReaderCallbacks** \*globalSnapReaderCallbacks, **OTF2\_GlobalSnapReaderCallback\_Unknown** unknownCallback)  
*Registers the callback for unknown snaps.*

### J.21.1 Detailed Description

This defines the callbacks for the global snap reader.

#### Source Template:

*templates/OTF2\_GlobalSnapReaderCallbacks tmpl.h*

### J.21.2 Typedef Documentation

**J.21.2.1 `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_ - Enter)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_RegionRef region)`**

Callback for the Enter snap record.

This record exists for each *Enter* event where the corresponding *Leave* event did not occur before the snapshot.

#### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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>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](#).

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

J.21.2.2 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-MeasurementOnOff)(OTF2\_LocationRef locationID,  
OTF2\_TimeStamp snapTime, void \*userData, OTF2\_-  
AttributeList \*attributeList, OTF2\_TimeStamp origEventTime,  
OTF2\_MeasurementMode measurementMode)**

Callback for the MeasurementOnOff snap record.

The last occurrence of an *MeasurementOnOff* event of this location, if any.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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>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](#).

J.21.2.3 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-Metric)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime,  
void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp  
origEventTime, OTF2\_MetricRef metric, uint8\_t numberOfMetrics, const  
OTF2\_Type \*typeIDs, const OTF2\_MetricValue \*metricValues)**

Callback for the Metric snap record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

As an exception for metric classes where the metric mode denotes an [OTF2\\_-METRIC\\_VALUE\\_RELATIVE](#) mode the value indicates the accumulation of all previous metric values recorded.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEventTime</i>	The original time this event happened.
<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.
<i>metricValues</i>	List of metric values.

### **Since**

Version 1.2

### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.21.2.4 `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_-MpiCollectiveBegin)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime)`**

Callback for the MpiCollectiveBegin snap record.

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

### **Parameters**

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEventTime</i>	The original time this event happened.

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.21.2.5 typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_ -
MpiCollectiveEnd)(OTF2_LocationRef locationID, OTF2_TimeStamp
snapTime, void *userData, OTF2_AttributeList *attributeList,
OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp,
OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t
sizeReceived)
```

Callback for the *MpiCollectiveEnd* snap record.

Indicates that this location completed a collective operation locally but not all of the participating locations completed the operation yet. The corresponding *MpiCollectiveBeginSaps* record is still in the snapshot though.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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>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 <i>communicator</i> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeRe- ceived</i>	Size of the received message.

### Since

Version 1.2

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.21.2.6 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-  
MpiIrecv)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime,  
void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp  
origEventTime, uint32\_t sender, OTF2\_CommRef communicator, uint32\_t  
msgTag, uint64\_t msgLength, uint64\_t requestID)**

Callback for the MpiIrecv snap record.

This record exists for each *MpiIrecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *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 <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent- Time</i>	The original time this event happened.
<i>sender</i>	MPI rank of sender in <i>communicator</i> .
<i>communi- cator</i>	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_- COMM</i> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### **Since**

Version 1.2

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

J.21.2.7 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-  
MpIrecvRequest)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
snapTime, void \*userData, OTF2\_AttributeList \*attributeList,  
OTF2\_TimeStamp origEventTime, uint64\_t requestID)**

Callback for the MpIrecvRequest 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 <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent- Time</i>	The original time this event happened.
<i>requestID</i>	ID of the requested receive

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.21.2.8 **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, uint64\_t requestID)**

Callback for the MpIsend snap record.

This record exists for each *MpiIsend* event where an corresponding *MpiIsendComplete* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIsendComplete* did occurred (the *MpiIsendCompleteSnap* record exists in the snapshot) but the matching receive message event

## APPENDIX J. FILE DOCUMENTATION

---

did not occur on the remote location before the snapshot. (This could either be an *MpiRecv* or an *MpiIrecv* event.)

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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 <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
<i>requestID</i>	ID of the related request

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.21.2.9** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_-MpiIsendComplete)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)`

Callback for the *MpiIsendComplete* snap record.

This record exists for each *MpiIsend* event where the corresponding *MpiIsendComplete* event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an *MpiRecv* or an *MpiIrecv* event.) .

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEventTime</i>	The original time this event happened.
<i>requestID</i>	ID of the related request

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.21.2.10 typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_-
    MpRecv)(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 MpRecv snap record.

This record exists for each [MpRecv](#) 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>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>origEventTime</i>	The original time this event happened.
<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 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.21.2.11 typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_-
    MpISend)(OTF2_LocationRef locationID, OTF2_TimeStamp snapTime,
    void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp
    origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t
    msgTag, uint64_t msgLength)
```

Callback for the MpISend snap record.

This record exists for each *MpISend* event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event. Note that it may so, that a previous *MpiIsend* with the same envelope than this one is neither completed nor canceled yet, thus the matching receive may already occurred, but the matching couldn't be done yet.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterGlobalSnapCallbacks</i> or <i>OTF2_GlobalSnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEvent- Time</i>	The original time this event happened.
<i>receiver</i>	MPI rank of receiver in <i>communicator</i> .
<i>communi- cator</i>	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

J.21.2.12 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-  
OmpAcquireLock)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
snapTime, void \*userData, OTF2\_AttributeList \*attributeList,  
OTF2\_TimeStamp origEventTime, uint32\_t lockID, uint32\_t acquisitionOrder)**

Callback for the OmpAcquireLock snap record.

This record exists for each *OmpAcquireLock* event where the corresponding *OmpReleaseLock* did not occurred before this snapshot yet.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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>lockID</i>	ID of the lock.
<i>acquisi- tionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

J.21.2.13 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-  
OmpFork)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
snapTime, void \*userData, OTF2\_AttributeList \*attributeList,  
OTF2\_TimeStamp origEventTime, uint32\_t numberOfRequestedThreads)**

Callback for the OmpFork snap record.

This record exists for each *OmpFork* event where the corresponding *OmpJoin* did not occurred before this snapshot.

### Parameters

<i>locationID</i>	The location where this snap happened.
-------------------	--

<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>origEventTime</i>	The original time this event happened.
<i>numberOfRequestedThreads</i>	Requested size of the team.

**Since**

Version 1.2

**Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.21.2.14 typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_-
OmpTaskCreate)(OTF2_LocationRef locationID, OTF2_TimeStamp
snapTime, void *userData, OTF2_AttributeList *attributeList,
OTF2_TimeStamp origEventTime, uint64_t taskID)
```

Callback for the OmpTaskCreate snap record.

This record exists for each *OmpTaskCreate* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

**Parameters**

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.
<i>origEventTime</i>	The original time this event happened.
<i>taskID</i>	Identifier of the newly created task instance.

**Since**

Version 1.2

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.21.2.15 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-  
OmpTaskSwitch)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
snapTime, void \*userData, OTF2\_AttributeList \*attributeList,  
OTF2\_TimeStamp origEventTime, uint64\_t taskID)**

Callback for the OmpTaskSwitch snap record.

This record exists for each *OmpTaskSwitch* event where the corresponding *Omp-TaskComplete* 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 <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>taskID</i>	Identifier of the now active task instance.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

J.21.2.16 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-  
ParameterInt)(OTF2\_LocationRef locationID, OTF2\_TimeStamp  
snapTime, void \*userData, OTF2\_AttributeList \*attributeList,  
OTF2\_TimeStamp origEventTime, OTF2\_ParameterRef parameter,  
int64\_t value)**

Callback for the ParameterInt snap record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

## **APPENDIX J. FILE DOCUMENTATION**

### **Parameters**

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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*.

J.21.2.17 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-ParameterString)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, OTF2\_ParameterRef parameter, OTF2StringRef 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.

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

<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](#).

**J.21.2.18** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback_-ParameterUnsignedInt)(OTF2_LocationRef locationID,  
OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList  
*attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef  
parameter, uint64_t value)`

Callback for the ParameterUnsignedInt snap record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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](#).

J.21.2.19 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-SnapshotEnd)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t contReadPos)**

Callback for the SnapshotEnd snap record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use [OTF2\\_EvtReader\\_Seek](#) with *contReadPos* as the position.

### Parameters

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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](#).

J.21.2.20 **typedef OTF2\_CallbackCode( \* OTF2\_GlobalSnapReaderCallback\_-SnapshotStart)(OTF2\_LocationRef locationID, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t numberOfRecords)**

Callback for the SnapshotStart snap record.

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one *SnapshotStart* record and closes with one *SnapshotEnd* record. All snapshot records inbetween are ordered by the *origEventTime*, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

## **J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference**

---

### **Parameters**

<i>locationID</i>	The location where this snap happened.
<i>time</i>	The time of this snapshot.
<i>userData</i>	User data as set by <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="#">Snap- shotEnd</a> record.

### **Since**

Version 1.2

### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.21.2.21** `typedef OTF2_CallbackCode( * OTF2_GlobalSnapReaderCallback _-  
Unknown)(OTF2_LocationRef locationID, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown snap record.

### **Parameters**

<i>locationID</i>	The location where this snap happened.
<i>snapTime</i>	The time of this snapshot.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterGlobalSnapCallbacks</a> or <a href="#">OTF2_GlobalSnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this snap.

### **Since**

Version 1.2

### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.21.2.22** `typedef struct OTF2_GlobalSnapReaderCallbacks_struct  
OTF2_GlobalSnapReaderCallbacks`

Opaque struct which holds all snap record callbacks.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.2

#### **J.21.3 Function Documentation**

**J.21.3.1 void OTF2\_GlobalSnapReaderCallbacks\_Clear ( OTF2\_-  
GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*  
)**

Clears a struct for the global snap callbacks.

#### **Parameters**

<i>global-SnapReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_GlobalSnapReaderCallbacks_New</a> .
-----------------------------------	---

### **Since**

Version 1.2

**J.21.3.2 void OTF2\_GlobalSnapReaderCallbacks\_Delete ( OTF2\_-  
GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*  
)**

Deallocates a struct for the global snap callbacks.

#### **Parameters**

<i>global-SnapReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_GlobalSnapReaderCallbacks_New</a> .
-----------------------------------	---

### **Since**

Version 1.2

**J.21.3.3 OTF2\_GlobalSnapReaderCallbacks\* OTF2\_GlobalSnapReaderCallbacks\_-  
New ( void )**

Allocates a new struct for the snap callbacks.

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

A newly allocated struct of type [OTF2\\_GlobalSnapReaderCallbacks](#).

**J.21.3.4 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetEnterCallback**  
( **OTF2\_GlobalSnapReaderCallbacks \* globalSnapReaderCallbacks,**  
**OTF2\_GlobalSnapReaderCallback\_enterCallback enterCallback** )

Registers the callback for the Enter snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter snaps.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid `defReaderCallbacks` argument

**J.21.3.5 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetMeasurementOnOffCallback**  
( **OTF2\_GlobalSnapReaderCallbacks \* globalSnapReaderCallbacks, OTF2\_GlobalSnapReaderCallback\_measurementOnOff measurementOnOffCallback** )

Registers the callback for the MeasurementOnOff snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
-----------------------------------	---------------------------

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>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

**J.21.3.6 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetMetricCallback ( OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*, OTF2\_GlobalSnapReaderCallback\_Metric *metricCallback* )**

Registers the callback for the Metric snap.

### **Parameters**

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric snaps.

### **Since**

Version 1.2

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

```
J.21.3.7 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_-
    SetMpiCollectiveBeginCallback ( OTF2_GlobalSnapReaderCallbacks
        * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_-
            MpiCollectiveBegin mpiCollectiveBeginCallback
    )
```

Registers the callback for the MpiCollectiveBegin snap.

### Parameters

<i>global-</i> <i>SnapRead-</i> <i>erCallbacks</i>	Struct for all callbacks.
<i>mpiCollec-</i> <i>tiveBegin-</i> <i>Callback</i>	Function which should be called for all MpiCollectiveBegin snaps.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful  
***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

```
J.21.3.8 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_-
    SetMpiCollectiveEndCallback ( OTF2_GlobalSnapReaderCallbacks *
        globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_-
            MpiCollectiveEnd mpiCollectiveEndCallback )
```

Registers the callback for the MpiCollectiveEnd snap.

### Parameters

<i>global-</i> <i>SnapRead-</i> <i>erCallbacks</i>	Struct for all callbacks.
<i>mpiCollec-</i> <i>tiveEnd-</i> <i>Callback</i>	Function which should be called for all MpiCollectiveEnd snaps.

## APPENDIX J. FILE DOCUMENTATION

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.21.3.9 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetMpilrecvCallback**  
( `OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks,`  
`OTF2_GlobalSnapReaderCallback_MpiIrecv mpilrecvCallback` )

Registers the callback for the MpiIrecv snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpilrecvCallback</i>	Function which should be called for all MpiIrecv snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.21.3.10 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-**  
**SetMpilrecvRequestCallback** ( `OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_-`  
`MpiIrecvRequest mpilrecvRequestCallback` )

Registers the callback for the MpiIrecvRequest snap.

### Parameters

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvRequestCallback</i>	Function which should be called for all MpiIrecvRequest snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.21.3.11 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetMpilsendCallback ( OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*, OTF2\_GlobalSnapReaderCallback\_MpiIsend *mpilsendCallback* )**

Registers the callback for the MpiIsend snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIsendCallback</i>	Function which should be called for all MpiIsend snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

```
J.21.3.12 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_-  
    SetMpiSendCompleteCallback ( OTF2_GlobalSnapReaderCallbacks  
        * globalSnapReaderCallbacks, OTF2_GlobalSnapReaderCallback_-  
        MpiSendComplete mpiSendCompleteCallback  
    )
```

Registers the callback for the MpiSendComplete snap.

#### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSendComplete-Callback</i>	Function which should be called for all MpiSendComplete snaps.

#### Since

Version 1.2

#### Returns

***OTF2\_SUCCESS*** if successful  
***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

```
J.21.3.13 OTF2_ErrorCode OTF2_GlobalSnapReaderCallbacks_SetMpiRecvCallback  
    ( OTF2_GlobalSnapReaderCallbacks * globalSnapReaderCallbacks,  
      OTF2_GlobalSnapReaderCallback_MpiRecv mpiRecvCallback )
```

Registers the callback for the MpiRecv snap.

#### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecv-Callback</i>	Function which should be called for all MpiRecv snaps.

#### Since

Version 1.2

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.21.3.14 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetMpiSendCallback ( OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*, OTF2\_GlobalSnapReaderCallback\_MpiSend *mpiSendCallback* )**

Registers the callback for the MpiSend snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSend-Callback</i>	Function which should be called for all MpiSend snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.21.3.15 OTF2\_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.

## APPENDIX J. FILE DOCUMENTATION

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.21.3.16 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetOmpForkCallback ( OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*, OTF2\_GlobalSnapReaderCallback *ompForkCallback* )**

Registers the callback for the OmpFork snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>ompFork-Callback</i>	Function which should be called for all OmpFork snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.21.3.17 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-SetOmpTaskCreateCallback ( OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*, OTF2\_GlobalSnapReaderCallback\_-OmpTaskCreate *ompTaskCreateCallback* )**

Registers the callback for the OmpTaskCreate snap.

### Parameters

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskCreate-Callback</i>	Function which should be called for all OmpTaskCreate snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.21.3.18 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-SetOmpTaskSwitchCallback( OTF2\_GlobalSnapReaderCallbacks \*  
globalSnapReaderCallbacks, OTF2\_GlobalSnapReaderCallback\_-  
OmpTaskSwitch *ompTaskSwitchCallback* )**

Registers the callback for the OmpTaskSwitch snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>omp-TaskSwitch-Callback</i>	Function which should be called for all OmpTaskSwitch snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

**J.21.3.19 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-  
SetParameterIntCallback ( OTF2\_GlobalSnapReaderCallbacks \*  
globalSnapReaderCallbacks, OTF2\_GlobalSnapReaderCallback\_-  
ParameterInt parameterIntCallback )**

Registers the callback for the ParameterInt snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>parameter-IntCallback</i>	Function which should be called for all ParameterInt snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid defReaderCallbacks argument

**J.21.3.20 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-  
SetParameterStringCallback ( OTF2\_GlobalSnapReaderCallbacks \*  
globalSnapReaderCallbacks, OTF2\_GlobalSnapReaderCallback\_-  
ParameterString parameterStringCallback )**

Registers the callback for the ParameterString snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>parameter-StringCallback</i>	Function which should be called for all ParameterString snaps.

### Since

Version 1.2

## J.21 OTF2\_GlobalSnapReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.21.3.21 **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-  
SetParameterUnsignedIntCallback ( OTF2\_GlobalSnapReaderCallbacks  
\* *globalSnapReaderCallbacks*, OTF2\_GlobalSnapReaderCallback\_-  
ParameterUnsignedInt *parameterUnsignedIntCallback*  
)**

Registers the callback for the ParameterUnsignedInt snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.21.3.22 **OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-  
SetSnapshotEndCallback ( OTF2\_GlobalSnapReaderCallbacks \*  
*globalSnapReaderCallbacks*, OTF2\_GlobalSnapReaderCallback\_-  
SnapshotEnd *snapshotEndCallback* )**

Registers the callback for the SnapshotEnd snap.

### Parameters

## APPENDIX J. FILE DOCUMENTATION

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>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

**J.21.3.23 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_-  
SetSnapshotStartCallback ( OTF2\_GlobalSnapReaderCallbacks \*  
globalSnapReaderCallbacks, OTF2\_GlobalSnapReaderCallback\_-  
SnapshotStart snapshotStartCallback )**

Registers the callback for the SnapshotStart snap.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>snapshotStartCallback</i>	Function which should be called for all SnapshotStart snaps.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

## J.22 OTF2\_IdMap.h File Reference

---

**J.21.3.24 OTF2\_ErrorCode OTF2\_GlobalSnapReaderCallbacks\_SetUnknownCallback**  
( OTF2\_GlobalSnapReaderCallbacks \* *globalSnapReaderCallbacks*,  
OTF2\_GlobalSnapReaderCallback\_Unknown *unknownCallback* )

Registers the callback for unknown snaps.

### Parameters

<i>global-SnapReaderCallbacks</i>	Struct for all callbacks.
<i>unknown-Callback</i>	Function which should be called for all unknown snaps.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.22 OTF2\_IdMap.h File Reference

Identifier mapping data structure, based on Scalasca's `epk_idmap.h`.

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <otf2/OTF2_ErrorCodes.h>
```

### Typedefs

- `typedef struct OTF2_IdMap_struct OTF2_IdMap`
- `typedef void(* OTF2_IdMap_TraverseCallback )(uint64_t localId, uint64_t globalId, void *userData)`

*Function prototype for use in OTF2\_IdMap\_Traverse.*

- `typedef uint8_t OTF2_IdMapMode`

### Enumerations

- enum `OTF2_IdMapMode_enum` {  
    `OTF2_ID_MAP_DENSE`,  
    `OTF2_ID_MAP_SPARSE` }

### Functions

- `OTF2_ErrorCode OTF2_IdMap_AddIdPair (OTF2_IdMap *instance, uint64_t localId, uint64_t globalId)`
- `OTF2_ErrorCode OTF2_IdMap_Clear (OTF2_IdMap *instance)`
- `OTF2_IdMap * OTF2_IdMap_Create (OTF2_IdMapMode mode, uint64_t capacity)`
- `OTF2_IdMap * OTF2_IdMap_CreateFromUint32Array (uint64_t length, const uint32_t *mappings, bool optimizeSize)`
- `OTF2_IdMap * OTF2_IdMap_CreateFromUint64Array (uint64_t length, const uint64_t *mappings, bool optimizeSize)`
- `void OTF2_IdMap_Free (OTF2_IdMap *instance)`
- `OTF2_ErrorCode OTF2_IdMap_GetGlobalId (const OTF2_IdMap *instance, uint64_t localId, uint64_t *globalId)`
- `OTF2_ErrorCode OTF2_IdMap_GetMode (const OTF2_IdMap *instance, OTF2_IdMapMode *mode)`
- `OTF2_ErrorCode OTF2_IdMap.GetSize (const OTF2_IdMap *instance, uint64_t *size)`
- `OTF2_ErrorCode OTF2_IdMap_Traverse (const OTF2_IdMap *instance, OTF2_IdMap_TraverseCallback callback, void *userData)`

#### J.22.1 Detailed Description

Identifier mapping data structure, based on Scalasca's epk\_idmap.h.

#### Maintainer:

Christian Rössel <[c.roessel@fz-juelich.de](mailto:c.roessel@fz-juelich.de)>

This file provides type definitions and function prototypes for an identifier mapping data structure which is used to store mapping tables for converting local into global identifiers.

This mapping data structure can operate in two different modes (see `OTF2_IdMapMode`): A dense mapping can be used if the local identifiers are consecutively enumerated from 0 to N-1. In this case, only the global identifier are stored in the table at the

## J.22 OTF2\_IdMap.h File Reference

---

corresponding entry, leading to compact storage and fast look-up. By contrast, if the local identifiers can consist of arbitrary numbers, a sparse mapping is necessary. Here, (localId, globalId) tuples are stored, which requires a more complicated look-up procedure.

### J.22.2 Typedef Documentation

#### J.22.2.1 `typedef struct OTF2_IdMap_struct OTF2_IdMap`

Opaque data structure representing an ID mapping table.

#### J.22.2.2 `typedef uint8_t OTF2_IdMapMode`

Wrapper around enum OTF2\_IdMapMode\_enum, so that it is guaranteed that it is a uint8\_t

### J.22.3 Enumeration Type Documentation

#### J.22.3.1 `enum OTF2_IdMapMode_enum`

Enumeration type defining the two different modes of an identifier mapping table.

**Enumerator:**

`OTF2_ID_MAP_DENSE` Dense mapping table

`OTF2_ID_MAP_SPARSE` Sparse mapping table

### J.22.4 Function Documentation

#### J.22.4.1 `OTF2_ErrorCode OTF2_IdMap_AddIdPair( OTF2_IdMap * instance, uint64_t localId, uint64_t globalId )`

Adds the given mapping from *localId* to *globalId* to the mapping table *instance*. If the current capacity does not suffice, the data structure is automatically resized.

**Note**

If the mapping table operates in dense mapping mode, the parameter *localId* has to correspond to the next entry in the mapping table.

**Parameters**

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>instance</i>	Object to add the mapping to.
<i>localId</i>	Local identifier.
<i>globalId</i>	Global identifier.

### **Returns**

OTF2\_SUCCESS, or error code.

#### **J.22.4.2 OTF2\_ErrorCode OTF2\_IdMap\_Clear ( OTF2\_IdMap \* *instance* )**

Removes all entries in the given mapping table *instance*. It can be used, e.g., to reuse an mapping table object for new input data.

### **Parameters**

<i>instance</i>	Object to remove entries from.
-----------------	--------------------------------

### **Returns**

OTF2\_SUCCESS, or error code.

#### **J.22.4.3 OTF2\_IdMap\* OTF2\_IdMap\_Create ( OTF2\_IdMapMode *mode*, uint64\_t *capacity* )**

Creates and returns a new instance of OTF2\_IdMap with the given *mode* and initial *capacity*. If the memory allocation request can not be fulfilled, NULL is returned.

### **Parameters**

<i>mode</i>	Mapping mode.
<i>capacity</i>	Initial capacity.

### **Returns**

Pointer to new instance or NULL if memory request couldn't be fulfilled.

#### **J.22.4.4 OTF2\_IdMap\* OTF2\_IdMap\_CreateFromUint32Array ( uint64\_t *length*, const uint32\_t \* *mappings*, bool *optimizeSize* )**

Creates and returns a new instance of OTF2\_IdMap from the array given by *mappings*.

## J.22 OTF2\_IdMap.h File Reference

---

Same as *OTF2\_IdMap\_CreateFromUint64Array*, except 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>optimizeSize</i>	Creates a SPARSE mapping, if the number of non-identities is less than half the array length.

### Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.

#### J.22.4.5 **OTF2\_IdMap\* OTF2\_IdMap\_CreateFromUint64Array ( uint64\_t *length*, const uint64\_t\* *mappings*, bool *optimizeSize* )**

Creates and returns a new instance of OTF2\_IdMap from the array given by *mappings*.

This creates always a DENSE mapping if *optimizeSize* is false. If it is true, it creates a SPARSE mapping, if the number of non-identitiy entries in the *mappings* array (ie. *mapping[ i ] != i*) is less than half the *length*.

Returns NULL when *optimizeSize* is true and the number of non-identitiy entries equals zero, ie. the given map is the identity map.

### Parameters

<i>length</i>	Number of elements in the <i>mappings</i> array.
<i>mappings</i>	Array with a dense mapping.
<i>optimizeSize</i>	Creates a SPARSE mapping, if the number of non-identities is less than half the array length.

### Returns

Pointer to new instance or NULL if memory request couldn't be fulfilled.

#### J.22.4.6 **void OTF2\_IdMap\_Free ( OTF2\_IdMap\* *instance* )**

Destroys the given *instance* of OTF2\_IdMap and releases the allocated memory.

### Parameters

<i>instance</i>	Object to be freed
-----------------	--------------------

## APPENDIX J. FILE DOCUMENTATION

### **J.22.4.7 OTF2\_ErrorCode OTF2\_IdMap\_GetGlobalId ( const OTF2\_IdMap \* instance, uint64\_t localId, uint64\_t \* globalId )**

Maps the given *localId* to the global id and store it in the storage provided by *globalId*.

If the given *localId* is not in the mapping, sets *globalId* to the *localId*.

#### **Parameters**

	<i>instance</i>	Object to add the mapping to.
	<i>localId</i>	Local identifier.
out	<i>globalId</i>	Global identifier.

#### **Returns**

OTF2\_SUCCESS, or error code.

### **J.22.4.8 OTF2\_ErrorCode OTF2\_IdMap\_GetMode ( const OTF2\_IdMap \* instance, OTF2\_IdMapMode \* mode )**

Returns the identifier mapping mode (dense/sparse) used for the given mapping table *instance*.

#### **Parameters**

	<i>instance</i>	Queried object.
out	<i>mode</i>	Identifier mapping mode.

#### **Returns**

OTF2\_SUCCESS, or error code.

### **J.22.4.9 OTF2\_ErrorCode OTF2\_IdMap.GetSize ( const OTF2\_IdMap \* instance, uint64\_t \* size )**

Returns the actual number of entries stored in the given OTF2\_IdMap *instance*.

#### **Parameters**

	<i>instance</i>	Queried object.
out	<i>size</i>	Number of entries.

## J.23 OTF2\_Marker.h File Reference

---

### Returns

OTF2\_SUCCESS, or error code.

**J.22.4.10 OTF2\_ErrorCode OTF2\_IdMap\_Traverse ( const OTF2\_IdMap \* instance, OTF2\_IdMap\_TraverseCallback callback, void \* userData )**

Calls for each mapping pair the callback *callback*.

### Parameters

<i>instance</i>	Object to add the mapping to.
<i>callback</i>	Callback function which is called for each mapping pair.
<i>userData</i>	Data which is passed to the <i>callback</i> function.

### Returns

OTF2\_SUCCESS, or error code.

## J.23 OTF2\_Marker.h File Reference

This file provides types and enums for markers.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
```

### Defines

- #define **OTF2\_UNDEFINED\_MARKER**(( **OTF2\_MarkerRef**)OTF2\_UNDEFINED\_-  
UINT32 )

*The invalid value for a reference to a Marker definition.*

### TypeDefs

- **typedef uint32\_t OTF2\_MarkerRef**  
*Type used to indicate a reference to a Marker definition.*
- **typedef uint8\_t OTF2\_MarkerScope**  
*Wrapper for enum OTF2\_MarkerScope\_enum.*
- **typedef uint8\_t OTF2\_MarkerSeverity**  
*Wrapper for enum OTF2\_MarkerSeverity\_enum.*

### **Enumerations**

- enum `OTF2_MarkerScope_enum` {  
    `OTF2_MARKER_SCOPE_GLOBAL`,  
    `OTF2_MARKER_SCOPE_LOCATION`,  
    `OTF2_MARKER_SCOPE_LOCATION_GROUP`,  
    `OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE`,  
    `OTF2_MARKER_SCOPE_GROUP`,  
    `OTF2_MARKER_SCOPE_COMM` }
- enum `OTF2_MarkerSeverity_enum` {  
    `OTF2_SEVERITY_NONE`,  
    `OTF2_SEVERITY_LOW`,  
    `OTF2_SEVERITY_MEDIUM`,  
    `OTF2_SEVERITY_HIGH` }

#### **J.23.1 Detailed Description**

This file provides types and enums for markers.

#### **J.23.2 Enumeration Type Documentation**

##### **J.23.2.1 enum OTF2\_MarkerScope\_enum**

A user marker does have a scope of its validity.

###### **Enumerator:**

**`OTF2_MARKER_SCOPE_GLOBAL`** The user marker has a global scope (could also be NONE).

**`OTF2_MARKER_SCOPE_LOCATION`** The user marker has a scope of a location.

**`OTF2_MARKER_SCOPE_LOCATION_GROUP`** The user marker has a scope of a location group.

**`OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE`** The user marker has a scope of a system tree.

**`OTF2_MARKER_SCOPE_GROUP`** The user marker has a scope of a group.

**`OTF2_MARKER_SCOPE_COMM`** The user marker has a scope of a communicator.

## J.24 OTF2\_MarkerReader.h File Reference

---

### J.23.2.2 enum OTF2\_MarkerSeverity\_enum

A list of possible severities of user markers.

#### Enumerator:

**OTF2\_SEVERITY\_NONE** The marker does not have a severity.

**OTF2\_SEVERITY\_LOW** The marker has a low severity.

**OTF2\_SEVERITY\_MEDIUM** The marker has a medium severity.

**OTF2\_SEVERITY\_HIGH** The marker has a high severity.

## J.24 OTF2\_MarkerReader.h File Reference

This file provides all routines that read marker records.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Marker.h>
#include <otf2/OTF2_MarkerReaderCallbacks.h>
```

### Functions

- **OTF2\_ErrorCode OTF2\_MarkerReader\_ReadMarkers (OTF2\_MarkerReader \*reader, uint64\_t recordsToRead, uint64\_t \*recordsRead)**

*After callback registration, the markers could be read with the following function. The user of this function tells the system how many markers it is able to handle (recordsToRead) and the function returns how many markers where in the stream (recordsRead). It should usually be the case that both values are the same. If this is not the case, then there where less records than requested in the stream.*

- **OTF2\_ErrorCode OTF2\_MarkerReader\_SetCallbacks (OTF2\_MarkerReader \*reader, const OTF2\_MarkerReaderCallbacks \*callbacks, void \*userData)**

*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### J.24.1 Detailed Description

This file provides all routines that read marker records.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **J.24.2 Function Documentation**

**J.24.2.1 OTF2\_ErrorCode OTF2\_MarkerReader\_ReadMarkers (**  
**OTF2\_MarkerReader \* reader, uint64\_t recordsToRead, uint64\_t \***  
**recordsRead )**

After callback registration, the markers could be read with the following function. The user of this function tells the system how many markers it is able to handle (recordsToRead) and the function returns how many markers where in the stream (recordsRead). It should usually be the case that both values are the same. If this is not the case, then there where less records than requested in the stream.

#### **Parameters**

<i>reader</i>	Reader Object.
<i>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.

**J.24.2.2 OTF2\_ErrorCode OTF2\_MarkerReader\_SetCallbacks (**  
**OTF2\_MarkerReader \* reader, const OTF2\_MarkerReaderCallbacks**  
**\* callbacks, void \* userData )**

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

#### **Parameters**

<i>reader</i>	This given reader object will be setted up with new callback functions.
<i>callbacks</i>	Struct which holds a function pointer for each record type. <a href="#">OTF2_MarkerReaderCallbacks_New</a> .
<i>userData</i>	Data passed as argument <i>userData</i> to the record callbacks.

## J.25 OTF2\_MarkerReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

`OTF2_SUCCESS` if successful, an error code if an error occurs.

## J.25 OTF2\_MarkerReaderCallbacks.h File Reference

This defines the callbacks for the marker reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_IdMap.h>
#include <otf2/OTF2_Marker.h>
```

### Typedefs

- `typedef OTF2_CallbackCode(* OTF2_MarkerReaderCallback_DefMarker )(void *userData, OTF2_MarkerRef self, const char *markerGroup, const char *markerCategory, OTF2_MarkerSeverity severity)`  
*Function pointer definition for the callback which is triggered by a Marker definition record.*
- `typedef OTF2_CallbackCode(* OTF2_MarkerReaderCallback_Marker )(void *userData, OTF2_TimeStamp timestamp, OTF2_TimeStamp duration, OTF2_MarkerRef marker, OTF2_MarkerScope scope, uint64_t scopeRef, const char *text)`  
*Function pointer definition for the callback which is triggered by a Marker record.*
- `typedef OTF2_CallbackCode(* OTF2_MarkerReaderCallback_Unknown )(void *userData)`  
*Function pointer definition for the callback which is triggered for an unknown marker.*
- `typedef struct OTF2_MarkerReaderCallbacks_struct OTF2_MarkerReaderCallbacks`

*Opaque struct which holds all definition record callbacks.*

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Functions**

- void [OTF2\\_MarkerReaderCallbacks\\_Clear](#) (OTF2\_MarkerReaderCallbacks \*markerReaderCallbacks)  
*Clears a struct for the marker callbacks.*
- void [OTF2\\_MarkerReaderCallbacks\\_Delete](#) (OTF2\_MarkerReaderCallbacks \*markerReaderCallbacks)  
*Deallocates a struct for the marker callbacks.*
- [OTF2\\_MarkerReaderCallbacks \\* OTF2\\_MarkerReaderCallbacks\\_New](#) (void)  
*Allocates a new struct for the marker callbacks.*
- [OTF2\\_ErrorCode OTF2\\_MarkerReaderCallbacks\\_SetDefMarkerCallback](#) (OTF2\_MarkerReaderCallbacks \*markerReaderCallbacks, [OTF2\\_MarkerReaderCallback\\_DefMarker](#) defMarkerCallback)  
*Registers the callback for the Marker definition.*
- [OTF2\\_ErrorCode OTF2\\_MarkerReaderCallbacks\\_SetMarkerCallback](#) (OTF2\_MarkerReaderCallbacks \*markerReaderCallbacks, [OTF2\\_MarkerReaderCallback\\_Marker](#) markerCallback)  
*Registers the callback for the Marker record.*
- [OTF2\\_ErrorCode OTF2\\_MarkerReaderCallbacks\\_SetUnknownCallback](#) (OTF2\_MarkerReaderCallbacks \*markerReaderCallbacks, [OTF2\\_MarkerReaderCallback\\_Unknown](#) unknownCallback)  
*Registers the callback for an unknown marker.*

#### **J.25.1 Detailed Description**

This defines the callbacks for the marker reader.

#### **J.25.2 Typedef Documentation**

**J.25.2.1 [typedef OTF2\\_CallbackCode\(\\* OTF2\\_MarkerReaderCallback\\_DefMarker\)\(void \\*userData, OTF2\\_MarkerRef self, const char \\*markerGroup, const char \\*markerCategory, OTF2\\_MarkerSeverity severity\)](#)**

Function pointer definition for the callback which is triggered by a Marker definition record.

#### **Parameters**

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterMarkerCallbacks</a> or <a href="#">OTF2_MarkerReader_SetCallbacks</a> .
<i>self</i>	Reference to this marker defintion.

## J.25 OTF2\_MarkerReaderCallbacks.h File Reference

---

<i>markerGroup</i>	Group name, e.g., "MUST", ...
<i>markerCategory</i>	Category, e.g., "Argument type error", ... The tuple (markerGroup, markerCategory) must be unique over all marker definitions.
<i>severity</i>	The severity for this marker category.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.25.2.2 `typedef OTF2_CallbackCode( * OTF2_MarkerReaderCallback_Marker)(void *userData, OTF2_TimeStamp timestamp, OTF2_TimeStamp duration, OTF2_MarkerRef marker, OTF2_MarkerScope scope, uint64_t scopeRef, const char *text)`**

Function pointer definition for the callback which is triggered by a Marker record.

### Parameters

<i>userData</i>	User data as set by <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 defintion.
<i>scope</i>	The type of scope of this marker instance.
<i>scopeRef</i>	The reference to an element of the scope of this marker. Depends on scope.
<i>text</i>	A textual description for this marker.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## APPENDIX J. FILE DOCUMENTATION

### **J.25.2.3 `typedef OTF2_CallbackCode( * OTF2_MarkerReaderCallback_Unknown)(void *userData)`**

Function pointer definition for the callback which is triggered for an unknown marker.

#### **Parameters**

<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterMarkerCallbacks</a> or <a href="#">OTF2_MarkerReader_SetCallbacks</a> .
-----------------	---

#### **Since**

Version 1.2

#### **Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

## **J.25.3 Function Documentation**

### **J.25.3.1 `void OTF2_MarkerReaderCallbacks_Clear ( OTF2_MarkerReaderCallbacks * markerReaderCallbacks )`**

Clears a struct for the marker callbacks.

#### **Since**

Version 1.2

#### **Parameters**

<i>markerReaderCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_MarkerReaderCallbacks_New</a> .
------------------------------	---

### **J.25.3.2 `void OTF2_MarkerReaderCallbacks_Delete ( OTF2_MarkerReaderCallbacks * markerReaderCallbacks )`**

Deallocates a struct for the marker callbacks.

#### **Since**

Version 1.2

## J.25 OTF2\_MarkerReaderCallbacks.h File Reference

---

### Parameters

<i>marker- Reader- Callbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_MarkerReaderCallbacks_New</a> .
--	---

**J.25.3.3 OTF2\_MarkerReaderCallbacks\* OTF2\_MarkerReaderCallbacks\_New ( void )**

Allocates a new struct for the marker callbacks.

### Since

Version 1.2

### Returns

A newly allocated struct of type [OTF2\\_MarkerReaderCallbacks](#).

**J.25.3.4 OTF2\_ErrorCode OTF2\_MarkerReaderCallbacks\_SetDefMarkerCallback ( OTF2\_MarkerReaderCallbacks \* markerReaderCallbacks, OTF2\_MarkerReaderCallback\_DefMarker defMarkerCallback )**

Registers the callback for the Marker definition.

### Parameters

<i>marker- Reader- Callbacks</i>	Struct for all callbacks.
<i>defMarker- Callback</i>	Function which should be called for all Marker definitions.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful

[OTF2\\_ERROR\\_INVALID\\_ARGUMENT](#) for an invalid *defReaderCallbacks* argument

## APPENDIX J. FILE DOCUMENTATION

**J.25.3.5 OTF2\_ErrorCode OTF2\_MarkerReaderCallbacks\_SetMarkerCallback**  
( **OTF2\_MarkerReaderCallbacks \* markerReaderCallbacks,**  
**OTF2\_MarkerReaderCallback\_Marker markerCallback** )

Registers the callback for the Marker record.

### Parameters

<i>marker- Reader- Callbacks</i>	Struct for all callbacks.
<i>marker- Callback</i>	Function which should be called for all Marker records.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful  
**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.25.3.6 OTF2\_ErrorCode OTF2\_MarkerReaderCallbacks\_SetUnknownCallback**  
( **OTF2\_MarkerReaderCallbacks \* markerReaderCallbacks,**  
**OTF2\_MarkerReaderCallback\_Unknown unknownCallback** )

Registers the callback for an unknown marker.

### Parameters

<i>marker- Reader- Callbacks</i>	Struct for all callbacks.
<i>unknown- Callback</i>	Function which should be called for all unknown definitions.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

## **J.26 OTF2\_MarkerWriter.h File Reference**

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

### **J.26 OTF2\_MarkerWriter.h File Reference**

This file provides all routines that write marker records.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_Marker.h>
```

#### **Typedefs**

- `typedef struct OTF2_MarkerWriter_struct OTF2_MarkerWriter`

*Handle definition for the external marker writer.*

#### **Functions**

- `OTF2_ErrorCode OTF2_MarkerWriter_WriteDefMarker (OTF2_MarkerWriter *writerHandle, OTF2_MarkerRef self, const char *markerGroup, const char *markerCategory, OTF2_MarkerSeverity severity)`

*Write a marker definition.*

- `OTF2_ErrorCode OTF2_MarkerWriter_WriteMarker (OTF2_MarkerWriter *writerHandle, uint64_t time, uint64_t duration, OTF2_MarkerRef marker, OTF2_MarkerScope scope, uint64_t scopeRef, const char *text)`

*Write a marker record.*

#### **J.26.1 Detailed Description**

This file provides all routines that write marker records.

## J.26.2 Function Documentation

**J.26.2.1 OTF2\_ErrorCode OTF2\_MarkerWriter\_WriteDefMarker (**  
**OTF2\_MarkerWriter \* writerHandle, OTF2\_MarkerRef self, const**  
**char \* markerGroup, const char \* markerCategory, OTF2\_MarkerSeverity**  
**severity )**

Write a marker definition.

### Parameters

<i>writerHandle</i>	Marker writer handle.
<i>self</i>	Reference to this marker definition.
<i>markerGroup</i>	Group name e.g. "MUST".
<i>markerCategory</i>	Category name e.g "Argument type error". The tuple (markerGroup, markerCategory) must be unique over all marker definitions.
<i>severity</i>	The severity for this marker category.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.26.2.2 OTF2\_ErrorCode OTF2\_MarkerWriter\_WriteMarker ( OTF2\_MarkerWriter**  
 $\ast$  *writerHandle*, uint64\_t *time*, uint64\_t *duration*, OTF2\_MarkerRef *marker*,  
**OTF2\_MarkerScope scope, uint64\_t scopeRef, const char \* text )**

Write a marker record.

### Parameters

<i>writerHandle</i>	Marker writer handle.
<i>time</i>	Time of the marker.
<i>duration</i>	A possible duration of this marker. May be 0.
<i>marker</i>	Reference to a marker definition.
<i>scope</i>	The type of scope of this marker instance: <i>OTF2_MARKER_SCOPE_GLOBAL</i> , <i>OTF2_MARKER_SCOPE_LOCATION</i> , <i>OTF2_MARKER_SCOPE_LOCATION_GROUP</i> , <i>OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE</i> , <i>OTF2_MARKER_SCOPE_GROUP</i> , or <i>OTF2_MARKER_SCOPE_COMM</i> .

## J.27 OTF2\_Reader.h File Reference

---

<i>scopeRef</i>	The reference to an element of the scope of this marker. Depends on scope.
<i>text</i>	A textual description for this marker.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.27 OTF2\_Reader.h File Reference

Reading interface for OTF2 archives.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Archive.h>
```

### Typedefs

- `typedef struct OTF2_Reader_struct OTF2_Reader`

*Keeps all necessary information for the reader.*

### Functions

- `OTF2_ErrorCode OTF2_Reader_Close (OTF2_Reader *reader)`  
*Close a reader handle.*
- `OTF2_ErrorCode OTF2_Reader_CloseDefReader (OTF2_Reader *reader, OTF2_DefReader *defReader)`  
*Close a local definition reader.*
- `OTF2_ErrorCode OTF2_Reader_CloseEvtReader (OTF2_Reader *reader, OTF2_EvtReader *evtReader)`  
*Close a local event reader.*
- `OTF2_ErrorCode OTF2_Reader_CloseGlobalDefReader (OTF2_Reader *reader, OTF2_GlobalDefReader *globalDefReader)`  
*Closes the global definition reader.*
- `OTF2_ErrorCode OTF2_Reader_CloseGlobalEvtReader (OTF2_Reader *reader, OTF2_GlobalEvtReader *globalEvtReader)`

## **APPENDIX J. FILE DOCUMENTATION**

---

*Closes the global event reader.*

- **OTF2\_ErrorCode OTF2\_Reader\_CloseGlobalSnapReader (OTF2\_Reader \*reader, OTF2\_GlobalSnapReader \*globalSnapReader)**

*Closes the global snapshot reader.*

- **OTF2\_ErrorCode OTF2\_Reader\_CloseMarkerReader (OTF2\_Reader \*reader, OTF2\_MarkerReader \*markerReader)**

*Closes the marker reader.*

- **OTF2\_ErrorCode OTF2\_Reader\_CloseMarkerWriter (OTF2\_Reader \*reader, OTF2\_MarkerWriter \*markerWriter)**

*Closes the marker writer.*

- **OTF2\_ErrorCode OTF2\_Reader\_CloseSnapReader (OTF2\_Reader \*reader, OTF2\_SnapReader \*snapReader)**

*Close a local snapshot reader.*

- **OTF2\_ErrorCode OTF2\_Reader\_CloseThumbReader (OTF2\_Reader \*reader, OTF2\_ThumbReader \*thumbReader)**

*Close an opened thumbnail reader.*

- **OTF2\_ErrorCode OTF2\_Reader\_GetBoolProperty (OTF2\_Reader \*reader, const char \*name, bool \*value)**

*Get the value of the named trace file property as boolean.*

- **OTF2\_ErrorCode OTF2\_Reader\_GetChunkSize (OTF2\_Reader \*reader, uint64\_t \*chunkSizeEvents, uint64\_t \*chunkSizeDefinitions)**

*Get event and definition chunk sizes.*

- **OTF2\_ErrorCode OTF2\_Reader\_GetCompression (OTF2\_Reader \*reader, OTF2\_Compression \*compression)**

*Get copression mode.*

- **OTF2\_ErrorCode OTF2\_Reader\_GetCreator (OTF2\_Reader \*reader, char \*\*creator)**

*Get creator name.*

- **OTF2\_DefReader \* OTF2\_Reader\_GetDefReader (OTF2\_Reader \*reader, OTF2\_LocationRef location)**

*Get a local definition reader.*

- **OTF2\_ErrorCode OTF2\_Reader\_GetDescription (OTF2\_Reader \*reader, char \*\*description)**

*Get description.*

- **OTF2\_EvtReader \* OTF2\_Reader\_GetEvtReader (OTF2\_Reader \*reader, OTF2\_LocationRef location)**

*Get a local event reader.*

- **OTF2\_ErrorCode OTF2\_Reader\_GetFileSubstrate (OTF2\_Reader \*reader, OTF2\_FileSubstrate \*substrate)**

*Get file substrate information.*

## J.27 OTF2\_Reader.h File Reference

---

- **OTF2\_GlobalDefReader \* OTF2\_Reader\_GetGlobalDefReader (OTF2\_Reader \*reader)**  
*Get a global definition reader.*
- **OTF2\_GlobalEvtReader \* OTF2\_Reader\_GetGlobalEvtReader (OTF2\_Reader \*reader)**  
*Get a global event reader.*
- **OTF2\_GlobalSnapReader \* OTF2\_Reader\_GetGlobalSnapReader (OTF2\_Reader \*reader)**  
*Get a global snap reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_GetMachineName (OTF2\_Reader \*reader, char \*\*machineName)**  
*Get machine name.*
- **OTF2\_MarkerReader \* OTF2\_Reader\_GetMarkerReader (OTF2\_Reader \*reader)**  
*Get a marker reader.*
- **OTF2\_MarkerWriter \* OTF2\_Reader\_GetMarkerWriter (OTF2\_Reader \*reader)**  
*Get a marker writer.*
- **OTF2\_ErrorCode OTF2\_Reader\_GetNumberOfGlobalDefinitions (OTF2\_Reader \*reader, uint64\_t \*numberOfDefinitions)**  
*Get number of global definitions.*
- **OTF2\_ErrorCode OTF2\_Reader\_GetNumberOfLocations (OTF2\_Reader \*reader, uint64\_t \*numberOfLocations)**  
*Get number of locations.*
- **OTF2\_ErrorCode OTF2\_Reader\_GetNumberOfSnapshots (OTF2\_Reader \*reader, uint32\_t \*number)**  
*Get number of snapshots.*
- **OTF2\_ErrorCode OTF2\_Reader\_GetNumberOfThumbnails (OTF2\_Reader \*reader, uint32\_t \*number)**  
*Get number of thumbs.*
- **OTF2\_ErrorCode OTF2\_ReaderGetProperty (OTF2\_Reader \*reader, const char \*name, char \*\*value)**  
*Get the value of the named trace file property.*
- **OTF2\_ErrorCode OTF2\_ReaderGetPropertyNames (OTF2\_Reader \*reader, uint32\_t \*numberOfProperties, char \*\*\*names)**  
*Get the names of all trace file properties.*
- **OTF2\_SnapReader \* OTF2\_Reader\_GetSnapReader (OTF2\_Reader \*reader, OTF2\_LocationRef location)**  
*Get a local snapshot reader.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ThumbReader \* OTF2\_Reader\_GetThumbReader** (**OTF2\_Reader** \*reader, **uint32\_t** number)  
*Get a thumb reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_GetTraceId** (**OTF2\_Reader** \*reader, **uint64\_t** \*id)  
*Get the identifier of the trace file.*
- **OTF2\_ErrorCode OTF2\_Reader\_GetVersion** (**OTF2\_Reader** \*reader, **uint8\_t** \*major, **uint8\_t** \*minor, **uint8\_t** \*bugfix)  
*Get OTF2 version.*
- **OTF2\_ErrorCode OTF2\_Reader\_HasGlobalEvent** (**OTF2\_Reader** \*reader, **OTF2\_GlobalEvtReader** \*evtReader, **int** \*flag)  
*Has the global event reader at least one more event to deliver.*
- **OTF2\_Reader \* OTF2\_Reader\_Open** (**const char** \*anchorFilePath)  
*Create a new reader handle.*
- **OTF2\_ErrorCode OTF2\_Reader\_ReadAllGlobalDefinitions** (**OTF2\_Reader** \*reader, **OTF2\_GlobalDefReader** \*defReader, **uint64\_t** \*definitionsRead)  
*Read all definitions via a global definition reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_ReadAllGlobalEvents** (**OTF2\_Reader** \*reader, **OTF2\_GlobalEvtReader** \*evtReader, **uint64\_t** \*eventsRead)  
*Read all events via a global event reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_ReadAllGlobalSnapshots** (**OTF2\_Reader** \*reader, **OTF2\_GlobalSnapReader** \*snapReader, **uint64\_t** \*recordsRead)  
*Read all records via a global snapshot reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_ReadAllLocalDefinitions** (**OTF2\_Reader** \*reader, **OTF2\_DefReader** \*defReader, **uint64\_t** \*definitionsRead)  
*Read all definitions via a local definition reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_ReadAllLocalEvents** (**OTF2\_Reader** \*reader, **OTF2\_EvtReader** \*evtReader, **uint64\_t** \*eventsRead)  
*Read all events via a local event reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_ReadAllLocalSnapshots** (**OTF2\_Reader** \*reader, **OTF2\_SnapReader** \*snapReader, **uint64\_t** \*recordsRead)  
*Read all records via a local snapshot reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_ReadAllMarkers** (**OTF2\_Reader** \*reader, **OTF2\_MarkerReader** \*markerReader, **uint64\_t** \*markersRead)  
*Read all markers via a marker reader.*
- **OTF2\_ErrorCode OTF2\_Reader\_ReadGlobalDefinitions** (**OTF2\_Reader** \*reader, **OTF2\_GlobalDefReader** \*defReader, **uint64\_t** definitionsToRead, **uint64\_t** \*definitionsRead)  
*Read a given number of definitions via a global definition reader.*

## J.27 OTF2\_Reader.h File Reference

---

- `OTF2_ErrorCode OTF2_Reader_ReadGlobalEvent (OTF2_Reader *reader, OTF2_GlobalEvtReader *evtReader)`

*Read an event via a global event reader.*
- `OTF2_ErrorCode OTF2_Reader_ReadGlobalEvents (OTF2_Reader *reader, OTF2_GlobalEvtReader *evtReader, uint64_t eventsToRead, uint64_t *eventsRead)`

*Read a given number of events via a global event reader.*
- `OTF2_ErrorCode OTF2_Reader_ReadGlobalSnapshots (OTF2_Reader *reader, OTF2_GlobalSnapReader *snapReader, uint64_t recordsToRead, uint64_t *recordsRead)`

*Read a given number of records via a global snapshot reader.*
- `OTF2_ErrorCode OTF2_Reader_ReadLocalDefinitions (OTF2_Reader *reader, OTF2_DefReader *defReader, uint64_t definitionsToRead, uint64_t *definitionsRead)`

*Read a given number of definitions via a local definition reader.*
- `OTF2_ErrorCode OTF2_Reader_ReadLocalEvents (OTF2_Reader *reader, OTF2_EvtReader *evtReader, uint64_t eventsToRead, uint64_t *eventsRead)`

*Read a given number of events via a local event reader.*
- `OTF2_ErrorCode OTF2_Reader_ReadLocalEventsBackward (OTF2_Reader *reader, OTF2_EvtReader *evtReader, uint64_t eventsToRead, uint64_t *eventsRead)`

*Read a given number of events via a local event reader backwards.*
- `OTF2_ErrorCode OTF2_Reader_ReadLocalSnapshots (OTF2_Reader *reader, OTF2_SnapReader *snapReader, uint64_t recordsToRead, uint64_t *recordsRead)`

*Read a given number of records via a local snapshot reader.*
- `OTF2_ErrorCode OTF2_Reader_ReadMarkers (OTF2_Reader *reader, OTF2_MarkerReader *markerReader, uint64_t markersToRead, uint64_t *markersRead)`

*Read a given number of markers via a marker reader.*
- `OTF2_ErrorCode OTF2_Reader_RegisterDefCallbacks (OTF2_Reader *reader, OTF2_DefReader *defReader, const OTF2_DefReaderCallbacks *callbacks, void *userData)`

*Register local definition reader callbacks.*
- `OTF2_ErrorCode OTF2_Reader_RegisterEvtCallbacks (OTF2_Reader *reader, OTF2_EvtReader *evtReader, const OTF2_EvtReaderCallbacks *callbacks, void *userData)`

*Register event reader callbacks.*

## **APPENDIX J. FILE DOCUMENTATION**

---

- **OTF2\_ErrorCode OTF2\_Reader\_RegisterGlobalDefCallbacks** (**OTF2\_Reader** \*reader, **OTF2\_GlobalDefReader** \*defReader, const **OTF2\_GlobalDefReaderCallbacks** \*callbacks, void \*userData)  
*Register global definition reader callbacks.*
- **OTF2\_ErrorCode OTF2\_Reader\_RegisterGlobalEvtCallbacks** (**OTF2\_Reader** \*reader, **OTF2\_GlobalEvtReader** \*evtReader, const **OTF2\_GlobalEvtReaderCallbacks** \*callbacks, void \*userData)  
*Register global event reader callbacks.*
- **OTF2\_ErrorCode OTF2\_Reader\_RegisterGlobalSnapCallbacks** (**OTF2\_Reader** \*reader, **OTF2\_GlobalSnapReader** \*evtReader, const **OTF2\_GlobalSnapReaderCallbacks** \*callbacks, void \*userData)  
*Register global event reader callbacks.*
- **OTF2\_ErrorCode OTF2\_Reader\_RegisterMarkerCallbacks** (**OTF2\_Reader** \*reader, **OTF2\_MarkerReader** \*markerReader, const **OTF2\_MarkerReaderCallbacks** \*callbacks, void \*userData)  
*Register marker reader callbacks.*
- **OTF2\_ErrorCode OTF2\_Reader\_RegisterSnapCallbacks** (**OTF2\_Reader** \*reader, **OTF2\_SnapReader** \*snapReader, const **OTF2\_SnapReaderCallbacks** \*callbacks, void \*userData)  
*Register snapshot event reader callbacks.*
- **OTF2\_ErrorCode OTF2\_Reader\_SetFileSionCallbacks** (**OTF2\_Reader** \*reader, const **OTF2\_FileSionCallbacks** \*fileSionCallbacks, void \*fileSionData)  
*Register SION callbacks to the reader.*

### **J.27.1 Detailed Description**

Reading interface for OTF2 archives.

#### **Maintainer:**

Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

#### **Authors**

Dominic Eschweiler <[d.eschweiler@fz-juelich.de](mailto:d.eschweiler@fz-juelich.de)>, Michael Wagner <[michael.wagner@zih.tu-dresden.de](mailto:michael.wagner@zih.tu-dresden.de)>

### **J.27.2 Function Documentation**

#### **J.27.2.1 OTF2\_ErrorCode OTF2\_Reader\_Close ( OTF2\_Reader \* reader )**

Close a reader handle.

## J.27 OTF2\_Reader.h File Reference

---

Closes a reader handle and releases all associated handles. Does nothing if NULL is provided.

### Parameters

<i>reader</i>	Reader handle.
---------------	----------------

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.27.2.2 OTF2\_ErrorCode OTF2\_Reader\_CloseDefReader ( **OTF2\_Reader \* reader,** **OTF2\_DefReader \* defReader )**

Close a local definition reader.

### Parameters

<i>reader</i>	Valid reader handle.
<i>defReader</i>	Definition reader to be closed.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.27.2.3 OTF2\_ErrorCode OTF2\_Reader\_CloseEvtReader ( **OTF2\_Reader \* reader,** **OTF2\_EvtReader \* evtReader )**

Close a local event reader.

### Parameters

<i>reader</i>	Valid reader handle.
<i>evtReader</i>	Event reader to be closed.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.27.2.4 OTF2\_ErrorCode OTF2\_Reader\_CloseGlobalDefReader ( **OTF2\_Reader \* reader,** **OTF2\_GlobalDefReader \* globalDefReader )**

Closes the global definition reader.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

<i>reader</i>	Valid reader handle.
<i>globalDef- Reader</i>	The global definition reader.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.27.2.5 OTF2\_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>glob- alEvtReader</i>	The global event reader.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.27.2.6 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

## J.27 OTF2\_Reader.h File Reference

---

**J.27.2.7 OTF2\_ErrorCode OTF2\_Reader\_CloseMarkerReader ( OTF2\_Reader \*  
reader, OTF2\_MarkerReader \* markerReader )**

Closes the marker reader.

### Parameters

<i>reader</i>	Valid reader handle.
<i>marker- Reader</i>	The marker reader.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.8 OTF2\_ErrorCode OTF2\_Reader\_CloseMarkerWriter ( OTF2\_Reader \*  
reader, OTF2\_MarkerWriter \* markerWriter )**

Closes the marker writer.

### Parameters

<i>reader</i>	Valid reader handle.
<i>marker- Writer</i>	The marker writer.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.9 OTF2\_ErrorCode OTF2\_Reader\_CloseSnapReader ( OTF2\_Reader \*  
reader, OTF2\_SnapReader \* snapReader )**

Close a local snapshot reader.

### Parameters

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>reader</i>	Valid reader handle.
<i>snapReader</i>	snapshot reader to be closed.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### **Since**

Version 1.2

**J.27.2.10 OTF2\_ErrorCode OTF2\_Reader\_CloseThumbReader ( OTF2\_Reader \*  
                  *reader*, OTF2\_ThumbReader \* *thumbReader* )**

Close an opened thumbnail reader.

### **Parameters**

<i>reader</i>	Reader handle.
<i>thum- bReader</i>	Thumbnail reader handle to be closed.

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.11 OTF2\_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.
<i>out</i>	<i>value</i>	Returned boolean value of the property.

## J.27 OTF2\_Reader.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_PROPERTY\_NOT\_FOUND* if the named property was not found

*OTF2\_ERROR\_PROPERTY\_VALUE\_INVALID* if the value could not be interpreted as an boolean value

### J.27.2.12 OTF2\_ErrorCode OTF2\_Reader\_GetChunkSize ( OTF2\_Reader \* *reader*, uint64\_t \* *chunkSizeEvents*, uint64\_t \* *chunkSizeDefinitions* )

Get event and definition chunk sizes.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>chunk-SizeEvents</i>	Returned size of event chunks
out	<i>chunk-SizeDefinitions</i>	Returned size of definition chunks.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.27.2.13 OTF2\_ErrorCode OTF2\_Reader\_GetCompression ( OTF2\_Reader \* *reader*, OTF2\_Compression \* *compression* )

Get compression 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.

## **APPENDIX J. FILE DOCUMENTATION**

---

**J.27.2.14 OTF2\_ErrorCode OTF2\_Reader\_GetCreator ( OTF2\_Reader \* *reader*,  
char \*\* *creator* )**

Get creator name.

### **Parameters**

	<i>reader</i>	Reader handle.
out	<i>creator</i>	Returned creator. Allocated with <i>malloc</i> .

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.15 OTF2\_DefReader\* OTF2\_Reader\_GetDefReader ( OTF2\_Reader \* *reader*,  
OTF2\_LocationRef *location* )**

Get a local definition reader.

### **Parameters**

<i>reader</i>	Valid reader handle.
<i>location</i>	Location ID for the requested local reader.

### **Returns**

Returns a handle to the local definition reader if successful, NULL otherwise.

**J.27.2.16 OTF2\_ErrorCode OTF2\_Reader\_GetDescription ( OTF2\_Reader \* *reader*,  
char \*\* *description* )**

Get description.

### **Parameters**

	<i>reader</i>	Reader handle.
out	<i>description</i>	Returned description. Allocated with <i>malloc</i> .

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.27 OTF2\_Reader.h File Reference

---

### J.27.2.17 OTF2\_EvtReader\* OTF2\_Reader\_GetEvtReader( OTF2\_Reader \* reader, OTF2\_LocationRef location )

Get a local event reader.

#### Parameters

<i>reader</i>	Valid reader handle.
<i>location</i>	Location ID for the requested local reader.

#### Returns

Returns a handle to the local event reader if successful, NULL otherwise.

### J.27.2.18 OTF2\_ErrorCode OTF2\_Reader\_GetFileSubstrate( OTF2\_Reader \* reader, OTF2\_FileSubstrate \* substrate )

Get file substrate information.

#### Parameters

	<i>reader</i>	Reader handle.
out	<i>substrate</i>	Returned file substrate.

#### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.27.2.19 OTF2\_GlobalDefReader\* OTF2\_Reader\_GetGlobalDefReader( OTF2\_Reader \* reader )

Get a global definition reader.

#### Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

#### Returns

Returns a handle to the global definition reader if successful, NULL otherwise.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **J.27.2.20 OTF2\_GlobalEvtReader\* OTF2\_Reader\_GetGlobalEvtReader ( OTF2\_Reader \* reader )**

Get a global event reader.

#### **Parameters**

<i>reader</i>	Valid reader handle.
---------------	----------------------

#### **Returns**

Returns a handle to the global event reader if successful, NULL otherwise.

### **J.27.2.21 OTF2\_GlobalSnapReader\* OTF2\_Reader\_GetGlobalSnapReader ( OTF2\_Reader \* reader )**

Get a global snap reader.

#### **Parameters**

<i>reader</i>	Valid reader handle.
---------------	----------------------

#### **Returns**

Returns a handle to the global snap reader if successful, NULL otherwise.

#### **Since**

Version 1.2

### **J.27.2.22 OTF2\_ErrorCode OTF2\_Reader\_GetMachineName ( OTF2\_Reader \* reader, char \*\* machineName )**

Get machine name.

#### **Parameters**

	<i>reader</i>	Reader handle.
<i>out</i>	<i>machine-Name</i>	Returned machine name. Allocated with <i>malloc</i> .

#### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.27 OTF2\_Reader.h File Reference

---

**J.27.2.23 OTF2\_MarkerReader\* OTF2\_Reader\_GetMarkerReader ( OTF2\_Reader \* *reader* )**

Get a marker reader.

### Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

### Since

Version 1.2

### Returns

Returns a handle to the marker reader if successful, NULL otherwise.

**J.27.2.24 OTF2\_MarkerWriter\* OTF2\_Reader\_GetMarkerWriter ( OTF2\_Reader \* *reader* )**

Get a marker writer.

### Parameters

<i>reader</i>	Valid reader handle.
---------------	----------------------

### Since

Version 1.2

### Returns

Returns a handle to the marker writer if successful, NULL otherwise.

**J.27.2.25 OTF2\_ErrorCode OTF2\_Reader\_GetNumberOfGlobalDefinitions ( OTF2\_Reader \* *reader*, uint64\_t \* *numberOfDefinitions* )**

Get number of global definitions.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>num- berOfDefi- nitions</i>	Returned number of global definitions.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.26 OTF2\_ErrorCode OTF2\_Reader\_GetNumberOfLocations ( OTF2\_Reader \*  
                  reader, uint64\_t \* numberOfLocations )**

Get number of locations.

### **Parameters**

	<i>reader</i>	Reader handle.
out	<i>number-</i> <i>berOfLoca-</i> <i>tions</i>	Returned number of locations.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.27 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

**J.27.2.28 OTF2\_ErrorCode OTF2\_Reader\_GetNumberOfThumbnails ( OTF2\_Reader  
                  \* reader, uint32\_t \* number )**

Get number of thumbs.

## J.27 OTF2\_Reader.h File Reference

---

### Parameters

	<i>reader</i>	Reader handle.
out	<i>number</i>	Returned number of thumbs.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### Since

Version 1.2

### J.27.2.29 OTF2\_ErrorCode OTF2\_ReaderGetProperty ( **OTF2\_Reader** \* *reader*, **const char** \* *name*, **char** \*\* *value* )

Get the value of the named trace file property.

### Parameters

	<i>reader</i>	Reader handle.
	<i>name</i>	Name of the property.
out	<i>value</i>	Returned value of the property. Allocated with <i>malloc</i> .

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_PROPERTY\_NOT\_FOUND* if the named property was not found

### J.27.2.30 OTF2\_ErrorCode OTF2\_ReaderGetPropertyNames ( **OTF2\_Reader** \* *reader*, **uint32\_t** \* *numberOfProperties*, **char** \*\*\* *names* )

Get the names of all trace file properties.

### Parameters

	<i>reader</i>	Reader handle.
out	<i>numberOfProperties</i>	Returned number of trace file properties.
out	<i>names</i>	Returned list of property names. Allocated with <i>malloc</i> . To release memory, just pass <i>*names</i> to <i>free</i> .

## APPENDIX J. FILE DOCUMENTATION

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.27.2.31 **OTF2\_SnapReader\* OTF2\_Reader\_GetSnapReader ( OTF2\_Reader \* reader, OTF2\_LocationRef location )**

Get a local snapshot reader.

### Parameters

<i>reader</i>	Valid reader handle.
<i>location</i>	Location ID for the requested local reader.

### Returns

Returns a handle to the local event reader if successful, NULL otherwise.

### Since

Version 1.2

#### J.27.2.32 **OTF2\_ThumbReader\* OTF2\_Reader\_GetThumbReader ( OTF2\_Reader \* reader, uint32\_t number )**

Get a thumb reader.

### Parameters

<i>reader</i>	Reader handle.
<i>number</i>	Thumbnail number.

### Since

Version 1.2

### Returns

Returns a global definition writer handle if successful, NULL if an error occurs.

## J.27 OTF2\_Reader.h File Reference

---

**J.27.2.33 OTF2\_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.

**J.27.2.34 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.

**J.27.2.35 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 event to read. To 0 if not. Otherwise the value is undefined.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### **J.27.2.36 OTF2\_Reader\* OTF2\_Reader\_Open ( const char \* anchorFilePath )**

Create a new reader handle.

Creates a new reader handle, opens an according archive handle, and calls a routine to register all neccessary function pointers.

### **Parameters**

<i>anchor- FilePath</i>	Path to the anchor file e.g. 'trace.otf2'. This can be a relative as well as an absolute path.
-----------------------------	--

### **Returns**

Returns a handle to the reader if successful, NULL otherwise.

#### **J.27.2.37 OTF2\_ErrorCode OTF2\_Reader\_ReadAllGlobalDefinitions ( OTF2\_Reader \* reader, OTF2\_GlobalDefReader \* defReader, uint64\_t \* definitionsRead )**

Read all definitions via a global definition reader.

### **Parameters**

	<i>reader</i>	Reader handle.
	<i>defReader</i>	Global definition reader handle.
<i>out</i>	<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.

#### **J.27.2.38 OTF2\_ErrorCode OTF2\_Reader\_ReadAllGlobalEvents ( OTF2\_Reader \* reader, OTF2\_GlobalEvtReader \* evtReader, uint64\_t \* eventsRead )**

Read all events via a global event reader.

### **Parameters**

## J.27 OTF2\_Reader.h File Reference

---

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Global event reader handle.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.39 OTF2\_ErrorCode OTF2\_Reader\_ReadAllGlobalSnapshots ( OTF2\_Reader \* *reader*, OTF2\_GlobalSnapReader \* *snapReader*, uint64\_t \* *recordsRead* )**

Read all records via a global snapshot reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Global snapshot reader handle.
out	<i>recordsRead</i>	Return pointer to the number of records

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### Since

Version 1.2

**J.27.2.40 OTF2\_ErrorCode OTF2\_Reader\_ReadAllLocalDefinitions ( OTF2\_Reader \* *reader*, OTF2\_DefReader \* *defReader*, uint64\_t \* *definitionsRead* )**

Read all definitions via a local definition reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>defReader</i>	Local definition reader handle.
out	<i>definitionsRead</i>	Return pointer to the number of definitions actually read.

## APPENDIX J. FILE DOCUMENTATION

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK* if an user supplied call-back returned OTF2\_CALLBACK\_INTERRUPT

*OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE* if an duplicate mapping table definition was read

*otherwise* the error code

### **J.27.2.41 OTF2\_ErrorCode OTF2\_Reader\_ReadAllLocalEvents ( OTF2\_Reader \* reader, OTF2\_EvtReader \* evtReader, uint64\_t \* eventsRead )**

Read all events via a local event reader.

#### Parameters

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Local event reader handle.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### **J.27.2.42 OTF2\_ErrorCode OTF2\_Reader\_ReadAllLocalSnapshots ( OTF2\_Reader \* reader, OTF2\_SnapReader \* snapReader, uint64\_t \* recordsRead )**

Read all records via a local snapshot reader.

#### Parameters

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Local snapshot reader handle.
out	<i>recordsRead</i>	Return pointer to the number of records

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### Since

Version 1.2

## J.27 OTF2\_Reader.h File Reference

---

**J.27.2.43 OTF2\_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>markerReader</i>	Marker reader handle.
out	<i>markersRead</i>	Return pointer to the number of markers actually read.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.44 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>definitionsToRead</i>	Number definitions to be read.
out	<i>definitionsRead</i>	Return pointer to the number of definitions actually read.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.45 OTF2\_ErrorCode OTF2\_Reader\_ReadGlobalEvent ( OTF2\_Reader \*  
reader, OTF2\_GlobalEvtReader \* evtReader )**

Read an event via a global event reader.

## **APPENDIX J. FILE DOCUMENTATION**

### **Parameters**

<i>reader</i>	Reader handle.
<i>evtReader</i>	Global event reader handle.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.46 OTF2\_ErrorCode OTF2\_Reader\_ReadGlobalEvents ( OTF2\_Reader \*  
reader, OTF2\_GlobalEvtReader \* evtReader, uint64\_t eventsToRead,  
uint64\_t \* eventsRead )**

Read a given number of events via a global event reader.

### **Parameters**

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Global event reader handle.
	<i>eventsToRead</i>	Number events to be read.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.47 OTF2\_ErrorCode OTF2\_Reader\_ReadGlobalSnapshots ( OTF2\_Reader \*  
reader, OTF2\_GlobalSnapReader \* snapReader, uint64\_t recordsToRead,  
uint64\_t \* recordsRead )**

Read a given number of records via a global snapshot reader.

### **Parameters**

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Global snapshot reader handle.
	<i>recordsToRead</i>	Number records to be read.
out	<i>recordsRead</i>	Return pointer to the number of records actually read.

## J.27 OTF2\_Reader.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### Since

Version 1.2

**J.27.2.48 OTF2\_ErrorCode OTF2\_Reader\_ReadLocalDefinitions ( OTF2\_Reader \*  
reader, OTF2\_DefReader \* defReader, uint64\_t definitionsToRead, uint64\_t \*  
definitionsRead )**

Read a given number of definitions via a local definition reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>defReader</i>	Local definition reader handle.
	<i>definitionsToRead</i>	Number definitions to be read.
out	<i>definitionsRead</i>	Return pointer to the number of definitions actually read.

### Returns

*OTF2\_SUCCESS* if successful

*OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK* if an user supplied call-back returned OTF2\_CALLBACK\_INTERRUPT

*OTF2\_ERROR\_DUPLICATE\_MAPPING\_TABLE* if an duplicate mapping table definition was read

*otherwise* the error code

**J.27.2.49 OTF2\_ErrorCode OTF2\_Reader\_ReadLocalEvents ( OTF2\_Reader \*  
reader, OTF2\_EvtReader \* evtReader, uint64\_t eventsToRead, uint64\_t \*  
eventsRead )**

Read a given number of events via a local event reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Local event reader handle.

## **APPENDIX J. FILE DOCUMENTATION**

---

	Number events to be read.
<i>eventsToRead</i>	Return pointer to the number of events actually read.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.50 OTF2\_ErrorCode OTF2\_Reader\_ReadLocalEventsBackward (**  
**OTF2\_Reader \* reader, OTF2\_EvtReader \* evtReader, uint64\_t**  
**eventsToRead, uint64\_t \* eventsRead )**

Read a given number of events via a local event reader backwards.

### **Parameters**

	<i>reader</i>	Reader handle.
	<i>evtReader</i>	Local event reader handle.
	<i>eventsToRead</i>	Number events to be read.
out	<i>eventsRead</i>	Return pointer to the number of events actually read.

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.51 OTF2\_ErrorCode OTF2\_Reader\_ReadLocalSnapshots ( OTF2\_Reader \***  
**reader, OTF2\_SnapReader \* snapReader, uint64\_t recordsToRead, uint64\_t**  
**\* recordsRead )**

Read a given number of records via a local snapshot reader.

### **Parameters**

	<i>reader</i>	Reader handle.
	<i>snapReader</i>	Local snapshot reader handle.
	<i>recordsToRead</i>	Number records to be read.
	<i>recordsRead</i>	Return pointer to the number of records actually read.

## J.27 OTF2\_Reader.h File Reference

---

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### Since

Version 1.2

**J.27.2.52 OTF2\_ErrorCode OTF2\_Reader\_ReadMarkers ( *OTF2\_Reader \* reader,*  
*OTF2\_MarkerReader \* markerReader, uint64\_t markersToRead, uint64\_t \**  
*markersRead )***

Read a given number of markers via a marker reader.

### Parameters

	<i>reader</i>	Reader handle.
	<i>marker- Reader</i>	Marker reader handle.
	<i>marker- sToRead</i>	Number markers to be read.
out	<i>marker- sRead</i>	Return pointer to the number of markers actually read.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.53 OTF2\_ErrorCode OTF2\_Reader\_RegisterDefCallbacks ( *OTF2\_Reader \* reader, OTF2\_DefReader \* defReader, const OTF2\_DefReaderCallbacks \* callbacks, void \* userData )***

Register local definition reader callbacks.

### Parameters

<i>reader</i>	OTF2_Reader handle.
<i>defReader</i>	Local definition reader handle.
<i>callbacks</i>	Callbacks for the local definition readers.
<i>userData</i>	Addition user data.

## APPENDIX J. FILE DOCUMENTATION

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.54 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.

**J.27.2.55 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.
<i>defReader</i>	Global definition reader handle.
<i>callbacks</i>	Callbacks for the global definition readers.
<i>userData</i>	Addition user data.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.27 OTF2\_Reader.h File Reference

---

**J.27.2.56 OTF2\_ErrorCode OTF2\_Reader\_RegisterGlobalEvtCallbacks (**  
    **OTF2\_Reader \* reader, OTF2\_GlobalEvtReader \* evtReader, const**  
    **OTF2\_GlobalEvtReaderCallbacks \* callbacks, void \* userData )**

Register global event reader callbacks.

### Parameters

<i>reader</i>	OTF2_Reader handle.
<i>evtReader</i>	Global event reader handle.
<i>callbacks</i>	Callbacks for the global event reader.
<i>userData</i>	Addition user data.

### Returns

Returns *OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.57 OTF2\_ErrorCode OTF2\_Reader\_RegisterGlobalSnapCallbacks (**  
    **OTF2\_Reader \* reader, OTF2\_GlobalSnapReader \* evtReader, const**  
    **OTF2\_GlobalSnapReaderCallbacks \* callbacks, void \* userData )**

Register global event reader callbacks.

### Parameters

<i>reader</i>	OTF2_Reader handle.
<i>evtReader</i>	Global event reader handle.
<i>callbacks</i>	Callbacks for the global event reader.
<i>userData</i>	Addition user data.

### Since

Version 1.2

### Returns

Returns *OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.58 OTF2\_ErrorCode OTF2\_Reader\_RegisterMarkerCallbacks (**  
    **OTF2\_Reader \* reader, OTF2\_MarkerReader \* markerReader, const**  
    **OTF2\_MarkerReaderCallbacks \* callbacks, void \* userData )**

Register marker reader callbacks.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Parameters**

<i>reader</i>	OTF2_Reader handle.
<i>marker- Reader</i>	Marker reader handle.
<i>callbacks</i>	Callbacks for the marker reader.
<i>userData</i>	Addition user data.

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.59 OTF2\_ErrorCode OTF2\_Reader\_RegisterSnapCallbacks (**  
**OTF2\_Reader \* *reader*, OTF2\_SnapReader \* *snapReader*, const**  
**OTF2\_SnapReaderCallbacks \* *callbacks*, void \* *userData* )**

Register snapshot event reader callbacks.

### **Parameters**

<i>reader</i>	OTF2_Reader handle.
<i>snapReader</i>	Local snap reader handle.
<i>callbacks</i>	Callbacks for the event readers.
<i>userData</i>	Addition user data.

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.27.2.60 OTF2\_ErrorCode OTF2\_Reader\_SetFileSionCallbacks ( OTF2\_Reader \***  
***reader*, const OTF2\_FileSionCallbacks \* *fileSionCallbacks*, void \***  
***fileSionData* )**

Register SION callbacks to the reader.

It suffice to provide a function for *OTF2\_FileSionGetRank*. The neccessary information for the rank mapping can be extracted from the global group definition

## J.28 OTF2\_SnapReader.h File Reference

---

of type *OTF2\_GROUP\_TYPE\_MPI\_LOCATIONS* or by the *locationGroupId* attribute of the Location definitions.

### Parameters

<i>reader</i>	Reader handle.
<i>fileSion-Callbacks</i>	Struct holding the callbacks.
<i>fileSion-Data</i>	Pointer passed to the callbacks by the caller.

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.28 OTF2\_SnapReader.h File Reference

This is the local snap reader, which reads snapshot events from one location.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_Definitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_SnapReaderCallbacks.h>
```

### Functions

- **OTF2\_ErrorCode OTF2\_SnapReader\_GetLocationID** (const *OTF2\_SnapReader* \**reader*, *OTF2\_LocationRef* \**location*)  
*Return the location ID of the reading related location.*
- **OTF2\_ErrorCode OTF2\_SnapReader\_ReadSnapshots** (*OTF2\_SnapReader* \**reader*, *uint64\_t* *recordsToRead*, *uint64\_t* \**recordsRead*)  
*After callback registration, the local events could be read with the following function. Readn reads recordsToRead records. The reader indicates that it reached the end of the trace by just reading less records than requested.*
- **OTF2\_ErrorCode OTF2\_SnapReader\_Seek** (*OTF2\_SnapReader* \**reader*, *uint64\_t* *req\_time*, *bool* \**found*)  
*Seek jumps to start of latest snapshot that was made before a given time 'req\_time'.*

- **OTF2\_ErrorCode OTF2\_SnapReader\_SetCallbacks (OTF2\_SnapReader \*reader, const OTF2\_SnapReaderCallbacks \*callbacks, void \*userData)**  
*Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.*

### J.28.1 Detailed Description

This is the local snap reader, which reads snapshot events from one location.

### J.28.2 Function Documentation

#### J.28.2.1 OTF2\_ErrorCode OTF2\_SnapReader\_GetLocationID ( const OTF2\_SnapReader \* reader, OTF2\_LocationRef \* location )

Return the location ID of the reading related location.

##### Parameters

	<i>reader</i>	Reader object which reads the snapshot events from its buffer.
out	<i>location</i>	ID of the location.

##### Since

Version 1.2

##### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

#### J.28.2.2 OTF2\_ErrorCode OTF2\_SnapReader\_ReadSnapshots ( OTF2\_SnapReader \* reader, uint64\_t recordsToRead, uint64\_t \* recordsRead )

After callback registration, the local events could be read with the following function. Readn reads *recordsToRead* records. The reader indicates that it reached the end of the trace by just reading less records than requested.

##### Parameters

	<i>reader</i>	Reader object which reads the events from its buffer.
	<i>recordsToRead</i>	How many records can be read next.

## J.28 OTF2\_SnapReader.h File Reference

---

out	<i>record-sRead</i>	Return how many records where really read.
-----	---------------------	--

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

### J.28.2.3 OTF2\_ErrorCode OTF2\_SnapReader\_Seek ( OTF2\_SnapReader \* *reader*, uint64\_t *req\_time*, bool \* *found* )

Seek jumps to start of latest snapshot that was made before a given time '*req\_time*'.

### Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
<i>req_time</i>	Requested time (see above)
<i>found</i>	returns if a matching snapshot was found

### Since

Version 1.2

### Returns

OTF2\_Error\_Code with !=OTF2\_SUCCESS if there was an error.

### J.28.2.4 OTF2\_ErrorCode OTF2\_SnapReader\_SetCallbacks ( OTF2\_SnapReader \* *reader*, const OTF2\_SnapReaderCallbacks \* *callbacks*, void \* *userData* )

Sets the callback functions for the given reader object. Everytime when OTF2 reads a record, a callback function is called and the records data is passed to this function. Therefore the programmer needs to set function pointers at the "callbacks" struct for the record type he wants to read.

These callbacks are ignored, if the events are read by an global event reader.

### Parameters

<i>reader</i>	Reader object which reads the events from its buffer.
---------------	---

## **APPENDIX J. FILE DOCUMENTATION**

---

<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.

## **J.29 OTF2\_SnapReaderCallbacks.h File Reference**

This defines the callbacks for the snap reader.

```
#include <stdint.h>
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_GeneralDefinitions.h>
#include <otf2/OTF2_AttributeList.h>
#include <otf2/OTF2_Events.h>
```

### **Typedefs**

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_Enter )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_RegionRef region)`

*Callback for the Enter snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MeasurementOnOff )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_MeasurementMode measurementMode)`

*Callback for the MeasurementOnOff snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_Metric )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)`

*Callback for the Metric snap event record.*

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime)`  
*Callback for the MpiCollectiveBegin snap event record.*
- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t sizeReceived)`  
*Callback for the MpiCollectiveEnd snap event record.*
- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiIrecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`  
*Callback for the MpiIrecv snap event record.*
- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)`  
*Callback for the MpiIrecvRequest snap event record.*
- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiIsend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength, uint64_t requestID)`  
*Callback for the MpiIsend snap event record.*
- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t requestID)`  
*Callback for the MpiIsendComplete snap event record.*
- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiRecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`  
*Callback for the MpiRecv snap event record.*
- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_MpiSend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t msgTag, uint64_t msgLength)`

---

## APPENDIX J. FILE DOCUMENTATION

*Callback for the MpiSend snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_OmpAcquireLock )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t lockID, uint32_t acquisitionOrder)`

*Callback for the OmpAcquireLock snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_OmpFork )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t num- berOfRequestedThreads)`

*Callback for the OmpFork snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_OmpTaskCreate )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t taskID)`

*Callback for the OmpTaskCreate snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_OmpTaskSwitch )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint64_t taskID)`

*Callback for the OmpTaskSwitch snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_ParameterInt )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, int64_t value)`

*Callback for the ParameterInt snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_ParameterString )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, OTF2StringRef string)`

*Callback for the ParameterString snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_ParameterUnsignedInt )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, uint64_t value)`

*Callback for the ParameterUnsignedInt snap event record.*

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_SnapshotEnd )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t contReadPos)`

*Callback for the SnapshotEnd snap event record.*

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_SnapshotStart )(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, uint64_t numberOfRecords)`  
*Callback for the SnapshotStart snap event record.*
- `typedef OTF2_CallbackCode(* OTF2_SnapReaderCallback_Unknown )(OTF2_- LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_- AttributeList *attributeList)`  
*Callback for an unknown snap event record.*
- `typedef struct OTF2_SnapReaderCallbacks_struct OTF2_SnapReaderCallbacks`

*Opaque struct which holds all snap event record callbacks.*

### Functions

- `void OTF2_SnapReaderCallbacks_Clear (OTF2_SnapReaderCallbacks *snapReaderCallbacks)`  
*Clears a struct for the snap event callbacks.*
- `void OTF2_SnapReaderCallbacks_Delete (OTF2_SnapReaderCallbacks *snapReaderCallbacks)`  
*Deallocates a struct for the snap event callbacks.*
- `OTF2_SnapReaderCallbacks * OTF2_SnapReaderCallbacks_New (void)`  
*Allocates a new struct for the snap event callbacks.*
- `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetEnterCallback (OTF2_- SnapReaderCallbacks *snapReaderCallbacks, OTF2_SnapReaderCallback_- Enter enterCallback)`  
*Registers the callback for the Enter snap event.*
- `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMeasurementOnOffCallback (OTF2_SnapReaderCallbacks *snapReaderCallbacks, OTF2_SnapReaderCallback_- MeasurementOnOff measurementOnOffCallback)`  
*Registers the callback for the MeasurementOnOff snap event.*
- `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMetricCallback (OTF2_- SnapReaderCallbacks *snapReaderCallbacks, OTF2_SnapReaderCallback_- Metric metricCallback)`  
*Registers the callback for the Metric snap event.*
- `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiCollectiveBeginCallback (OTF2_SnapReaderCallbacks *snapReaderCallbacks, OTF2_SnapReaderCallback_- MpiCollectiveBegin mpiCollectiveBeginCallback)`  
*Registers the callback for the MpiCollectiveBegin snap event.*
- `OTF2_ErrorCode OTF2_SnapReaderCallbacks_SetMpiCollectiveEndCallback (OTF2_SnapReaderCallbacks *snapReaderCallbacks, OTF2_SnapReaderCallback_- MpiCollectiveEnd mpiCollectiveEndCallback)`  
*Registers the callback for the MpiCollectiveEnd snap event.*

## **APPENDIX J. FILE DOCUMENTATION**

---

*Registers the callback for the MpiCollectiveEnd snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiIrecvCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_MpiIrecv mpiIrecvCallback)**

*Registers the callback for the MpiIrecv snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiIrecvRequestCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_MpiIrecvRequest mpiIrecvRequestCallback)**

*Registers the callback for the MpiIrecvRequest snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiIsendCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_MpiIsend mpiIsendCallback)**

*Registers the callback for the MpiIsend snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiIsendCompleteCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_MpiIsendComplete mpiIsendCompleteCallback)**

*Registers the callback for the MpiIsendComplete snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiRecvCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_MpiRecv mpiRecvCallback)**

*Registers the callback for the MpiRecv snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiSendCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_MpiSend mpiSendCallback)**

*Registers the callback for the MpiSend snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetOmpAcquireLockCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_OmpAcquireLock ompAcquireLockCallback)**

*Registers the callback for the OmpAcquireLock snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetOmpForkCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_OmpFork ompForkCallback)**

*Registers the callback for the OmpFork snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetOmpTaskCreateCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_OmpTaskCreate ompTaskCreateCallback)**

*Registers the callback for the OmpTaskCreate snap event.*

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetOmpTaskSwitchCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_OmpTaskSwitch ompTaskSwitchCallback)**

*Registers the callback for the OmpTaskSwitch snap event.*

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetParameterIntCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_ParameterInt parameterIntCallback)**  
*Registers the callback for the ParameterInt snap event.*
- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetParameterStringCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_ParameterString parameterStringCallback)**  
*Registers the callback for the ParameterString snap event.*
- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetParameterUnsignedIntCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_ParameterUnsignedInt parameterUnsignedIntCallback)**  
*Registers the callback for the ParameterUnsignedInt snap event.*
- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetSnapshotEndCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_SnapshotEnd snapshotEndCallback)**  
*Registers the callback for the SnapshotEnd snap event.*
- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetSnapshotStartCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_SnapshotStart snapshotStartCallback)**  
*Registers the callback for the SnapshotStart snap event.*
- **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetUnknownCallback (OTF2\_SnapReaderCallbacks \*snapReaderCallbacks, OTF2\_SnapReaderCallback\_Unknown unknownCallback)**  
*Registers the callback for the Unknown snap event.*

### J.29.1 Detailed Description

This defines the callbacks for the snap reader.

#### Source Template:

*templates/OTF2\_SnapReaderCallbacks.tpl.h*

### J.29.2 Typedef Documentation

#### J.29.2.1 **typedef OTF2\_CallbackCode( \* OTF2\_SnapReaderCallback\_Enter)(OTF2\_LocationRef location, OTF2\_TimeStamp snapTime, void \*userData, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp origEventTime, OTF2\_RegionRef region)**

Callback for the Enter snap event record.

## **APPENDIX J. FILE DOCUMENTATION**

---

This record exists for each *Enter* event where the corresponding *Leave* event did not occur before the snapshot.

### **Parameters**

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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.
<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](#).

```
J.29.2.2 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
MeasurementOnOff)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_MeasurementMode  
measurementMode)
```

Callback for the *MeasurementOnOff* snap event record.

The last occurrence of an *MeasurementOnOff* event of this location, if any.

### **Parameters**

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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.
<i>measurementMode</i>	Is the measurement turned on ( <a href="#">OTF2_MEASUREMENT_ON</a> ) or off ( <a href="#">OTF2_MEASUREMENT_OFF</a> )?

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

```
J.29.2.3 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_  
Metric)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void  
*userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, OTF2_MetricRef metric, uint8_t numberOfMetrics, const  
OTF2_Type *typeIDs, const OTF2_MetricValue *metricValues)
```

Callback for the Metric snap event record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

As an exception for metric classes where the metric mode denotes 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>origEventTime</i>	The original time this event happened.
<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.
<i>metricValues</i>	List of metric values.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.29.2.4** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
MpiCollectiveBegin)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime)`

Callback for the *MpiCollectiveBegin* snap event record.

Indicates that this location started a collective operation but not all of the participating locations completed the operation yet, including this location.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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.

### Since

Version 1.2

### Returns

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.29.2.5** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
MpiCollectiveEnd)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, OTF2_CollectiveOp collectiveOp,  
OTF2_CommRef communicator, uint32_t root, uint64_t sizeSent, uint64_t  
sizeReceived)`

Callback for the *MpiCollectiveEnd* snap event record.

Indicates that this location completed a collective operation locally but not all of the participating locations completed the operation yet. The corresponding *MpiCollectiveBeginSaps* record is still in the snapshot though.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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.
<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 <i>communicator</i> .
<i>sizeSent</i>	Size of the sent message.
<i>sizeReceived</i>	Size of the received message.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.29.2.6 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
    MpiIrecv)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,  
    void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
    origEventTime, uint32_t sender, OTF2_CommRef communicator, uint32_t  
    msgTag, uint64_t msgLength, uint64_t requestId)
```

Callback for the *MpiIrecv* snap event record.

This record exists for each *MpiIrecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiSendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

### Parameters

<i>location</i>	The location where this snap event happened.
-----------------	--

## APPENDIX J. FILE DOCUMENTATION

---

<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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.
<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
<i>requestID</i>	ID of the related request

**Since**

Version 1.2

**Returns**

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.29.2.7 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -
MpiIrecvRequest)(OTF2_LocationRef location, OTF2_TimeStamp
snapTime, void *userData, OTF2_AttributeList *attributeList,
OTF2_TimeStamp origEventTime, uint64_t requestID)
```

Callback for the *MpiIrecvRequest* snap event record.

This record exists for each *MpiIrecvRequest* event where an corresponding *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 <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

<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*.

```
J.29.2.8 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-
    MpiIsend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,
    void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp
    origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t
    msgTag, uint64_t msgLength, uint64_t requestID)
```

Callback for the *MpiIsend* snap event record.

This record exists for each *MpiIsend* event where an corresponding *MpiIsendComplete* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIsendComplete* did occurred (the *MpiIsendCompleteSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. (This could either be an*MpiRecv* or an *MpiIrecv* event.)

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent-Time</i>	The original time this event happened.
<i>receiver</i>	MPI rank of receiver in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

**Since**

Version 1.2

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.29.2.9** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
MpiIsendComplete)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t requestID)`

Callback for the `MpiIsendComplete` snap event record.

This record exists for each `MpiIsend` event where the corresponding `MpiIsendComplete` event occurred, but where the matching receive message event did not occur on the remote location before the snapshot. (This could either be an `MpiRecv` or an `MpiIrecv` event.) .

**Parameters**

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <code>OTF2_Reader_RegisterSnapCallbacks</code> or <code>OTF2_- SnapReader_SetCallbacks</code> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent- Time</i>	The original time this event happened.
<i>requestID</i>	ID of the related request

**Since**

Version 1.2

**Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

```
J.29.2.10 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -
    MpRecv)(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 MpRecv snap event record.

This record exists for each *MpRecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpSend* 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>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.
<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 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.29.2.11** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
MpiSend)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp  
origEventTime, uint32_t receiver, OTF2_CommRef communicator, uint32_t  
msgTag, uint64_t msgLength)`

Callback for the *MpiSend* snap event record.

This record exists for each *MpiSend* event where the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event. Note that it may so, that a previous *MpiIsend* with the same envelope than this one is neither completed nor 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 happened.
<i>receiver</i>	MPI rank of receiver 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\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.29.2.12** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
OmpAcquireLock)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint32_t lockID, uint32_t acquisitionOrder)`

Callback for the *OmpAcquireLock* snap event record.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

This record exists for each *OmpAcquireLock* event where the corresponding *OmpReleaseLock* did not occurred before this snapshot yet.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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.
<i>lockID</i>	ID of the lock.
<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.29.2.13** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-OmpFork)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, uint32_t numberOfRequestedThreads)`

Callback for the OmpFork snap event record.

This record exists for each *OmpFork* event where the corresponding *OmpJoin* did not occurred before this snapshot.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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.

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>num- berOfRe- quest- edThreads</i>	Requested size of the team.
---	-----------------------------

### **Since**

Version 1.2

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

**J.29.2.14** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
OmpTaskCreate)(OTF2_LocationRef location, OTF2_TimeStamp  
snapTime, void *userData, OTF2_AttributeList *attributeList,  
OTF2_TimeStamp origEventTime, uint64_t taskID)`

Callback for the OmpTaskCreate snap event record.

This record exists for each *OmpTaskCreate* event where the corresponding *Omp-TaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

### **Parameters**

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <i>OTF2_Reader_RegisterSnapCallbacks</i> or <i>OTF2_- SnapReader_SetCallbacks</i> .
<i>attributeList</i>	Additional attributes for this event.
<i>origEvent- Time</i>	The original time this event happened.
<i>taskID</i>	Identifier of the newly created task instance.

### **Since**

Version 1.2

### **Returns**

*OTF2\_CALLBACK\_SUCCESS* or *OTF2\_CALLBACK\_INTERRUPT*.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

```
J.29.2.15 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -
OmpTaskSwitch)(OTF2_LocationRef location, OTF2_TimeStamp
snapTime, void *userData, OTF2_AttributeList *attributeList,
OTF2_TimeStamp origEventTime, uint64_t taskID)
```

Callback for the OmpTaskSwitch snap event record.

This record exists for each *OmpTaskSwitch* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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>taskID</i>	Identifier of the now active task instance.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

```
J.29.2.16 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_ -
ParameterInt)(OTF2_LocationRef location, OTF2_TimeStamp
snapTime, void *userData, OTF2_AttributeList *attributeList,
OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter,
int64_t value)
```

Callback for the ParameterInt snap event record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>location</i>	The location where this snap event happened.
-----------------	--

## APPENDIX J. FILE DOCUMENTATION

---

<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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](#).

**J.29.2.17** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-ParameterString)(OTF2_LocationRef location, OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList *attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, OTF2StringRef 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.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

<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](#).

```
J.29.2.18 typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback_-  
          ParameterUnsignedInt)(OTF2_LocationRef location,  
          OTF2_TimeStamp snapTime, void *userData, OTF2_AttributeList  
          *attributeList, OTF2_TimeStamp origEventTime, OTF2_ParameterRef  
          parameter, uint64_t value)
```

Callback for the ParameterUnsignedInt snap event record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <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](#).

## APPENDIX J. FILE DOCUMENTATION

J.29.2.19 **typedef OTF2\_CallbackCode( \* OTF2\_SnapReaderCallback\_-  
SnapshotEnd)(OTF2\_LocationRef location, OTF2\_TimeStamp  
snapTime, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t  
contReadPos)**

Callback for the SnapshotEnd snap event record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use [OTF2\\_EvtReader\\_Seek](#) with *contReadPos* as the position.

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapshotTime</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](#).

J.29.2.20 **typedef OTF2\_CallbackCode( \* OTF2\_SnapReaderCallback\_-  
SnapshotStart)(OTF2\_LocationRef location, OTF2\_TimeStamp  
snapTime, void \*userData, OTF2\_AttributeList \*attributeList, uint64\_t  
numberOfRecords)**

Callback for the SnapshotStart snap event record.

This record marks the start of a snapshot.

A snapshot consists of an timestamp and a set of snapshot records. All these snapshot records have the same snapshot time. A snapshot starts with one [SnapshotStart](#) record and closes with one [SnapshotEnd](#) record. All snapshot records inbetween are ordered by the *origEventTime*, which are also less than the snapshot timestamp. Ie. The timestamp of the next event read from the event stream is greater or equal to the snapshot time.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

### Parameters

<i>location</i>	The location where this snap event happened.
<i>snapTime</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</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="#">Snap- shotEnd</a> record.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.29.2.21** `typedef OTF2_CallbackCode( * OTF2_SnapReaderCallback _ -  
Unknown)(OTF2_LocationRef location, OTF2_TimeStamp snapTime,  
void *userData, OTF2_AttributeList *attributeList)`

Callback for an unknown snap event record.

### Parameters

<i>location</i>	The location where this event happened.
<i>time</i>	Snapshot time.
<i>userData</i>	User data as set by <a href="#">OTF2_Reader_RegisterSnapCallbacks</a> or <a href="#">OTF2_SnapReader_SetCallbacks</a> .
<i>attributeList</i>	Additional attributes for this event.

### Since

Version 1.2

### Returns

[OTF2\\_CALLBACK\\_SUCCESS](#) or [OTF2\\_CALLBACK\\_INTERRUPT](#).

**J.29.2.22** `typedef struct OTF2_SnapReaderCallbacks_struct  
OTF2_SnapReaderCallbacks`

Opaque struct which holds all snap event record callbacks.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.2

#### **J.29.3 Function Documentation**

**J.29.3.1 void OTF2\_SnapReaderCallbacks\_Clear ( OTF2\_SnapReaderCallbacks \* snapReaderCallbacks )**

Clears a struct for the samp event callbacks.

### **Parameters**

<i>snapRead-</i> <i>erCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_SnapReaderCallbacks_New</a> .
--	---

### **Since**

Version 1.2

**J.29.3.2 void OTF2\_SnapReaderCallbacks\_Delete ( OTF2\_SnapReaderCallbacks \* snapReaderCallbacks )**

Deallocates a struct for the snap event callbacks.

### **Parameters**

<i>snapRead-</i> <i>erCallbacks</i>	Handle to a struct previously allocated with <a href="#">OTF2_SnapReaderCallbacks_New</a> .
--	---

### **Since**

Version 1.2

**J.29.3.3 OTF2\_SnapReaderCallbacks\* OTF2\_SnapReaderCallbacks\_New ( void )**

Allocates a new struct for the snap event callbacks.

### **Since**

Version 1.2

### **Returns**

A newly allocated struct of type [OTF2\\_SnapReaderCallbacks](#).

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

J.29.3.4 **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetEnterCallback**  
( **OTF2\_SnapReaderCallbacks \* snapReaderCallbacks,**  
**OTF2\_SnapReaderCallback\_Enter enterCallback** )

Registers the callback for the Enter snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>enterCallback</i>	Function which should be called for all Enter events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

J.29.3.5 **OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMeasurementOnOffCallback**  
( **OTF2\_SnapReaderCallbacks \* snapReaderCallbacks,**  
**OTF2\_SnapReaderCallback\_MeasurementOnOff**  
**measurementOnOffCallback** )

Registers the callback for the MeasurementOnOff snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>measurementOnOffCallback</i>	Function which should be called for all MeasurementOnOff events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

## APPENDIX J. FILE DOCUMENTATION

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.29.3.6 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMetricCallback  
( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_Metric *metricCallback* )**

Registers the callback for the Metric snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>metricCallback</i>	Function which should be called for all Metric events.

### Since

Version 1.2

### Returns

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.29.3.7 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiCollectiveBeginCallback  
( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_MpiCollectiveBegin  
*mpiCollectiveBeginCallback* )**

Registers the callback for the MpiCollectiveBegin snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiCollectiveBeginCallback</i>	Function which should be called for all MpiCollectiveBegin events.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.29.3.8 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiCollectiveEndCallback**  
( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`  
`OTF2_SnapReaderCallback_MpiCollectiveEnd`  
`mpiCollectiveEndCallback` )

Registers the callback for the MpiCollectiveEnd snap event.

### Parameters

<code>snapReaderCallbacks</code>	Struct for all callbacks.
<code>mpiCollectiveEndCallback</code>	Function which should be called for all MpiCollectiveEnd events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.29.3.9 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpilrecvCallback**  
( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`  
`OTF2_SnapReaderCallback_MpiIrecv mpilrecvCallback` )

Registers the callback for the Mpilrecv snap event.

### Parameters

---

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiIrecvCallback</i>	Function which should be called for all MpiIrecv events.

### **Since**

Version 1.2

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.29.3.10 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiRecvRequestCallback**  
( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`  
`OTF2_SnapReaderCallback_MpiRecvRequest`  
`mpirecvRequestCallback` )

Registers the callback for the MpiRecvRequest snap event.

### **Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpirecvRequestCallback</i>	Function which should be called for all MpiRecvRequest events.

### **Since**

Version 1.2

### **Returns**

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

**J.29.3.11 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpilsendCallback**  
( **OTF2\_SnapReaderCallbacks \* snapReaderCallbacks,**  
**OTF2\_SnapReaderCallback\_MpiSend mpilsendCallback** )

Registers the callback for the MpiSend snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpilsendCallback</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

**J.29.3.12 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpilsendCompleteCallback**  
( **OTF2\_SnapReaderCallbacks \* snapReaderCallbacks,**  
**OTF2\_SnapReaderCallback\_MpiSendComplete**  
**mpilsendCompleteCallback** )

Registers the callback for the MpiSendComplete snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpilsendCompleteCallback</i>	Function which should be called for all MpiSendComplete events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

## **APPENDIX J. FILE DOCUMENTATION**

---

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.29.3.13 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiRecvCallback  
( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_MpiRecv *mpiRecvCallback* )**

Registers the callback for the MpiRecv snap event.

### **Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiRecvCallback</i>	Function which should be called for all MpiRecv events.

### **Since**

Version 1.2

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.29.3.14 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetMpiSendCallback  
( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_MpiSend *mpiSendCallback* )**

Registers the callback for the MpiSend snap event.

### **Parameters**

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>mpiSendCallback</i>	Function which should be called for all MpiSend events.

### **Since**

Version 1.2

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.29.3.15 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetOmpAcquireLockCallback**  
( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`  
`OTF2_SnapReaderCallback_OmpAcquireLock`  
`ompAcquireLockCallback` )

Registers the callback for the OmpAcquireLock snap event.

### Parameters

<code>snapReaderCallbacks</code>	Struct for all callbacks.
<code>ompAcquireLockCallback</code>	Function which should be called for all OmpAcquireLock events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

**J.29.3.16 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetOmpForkCallback**  
( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`  
`OTF2_SnapReaderCallback_OmpFork ompForkCallback` )

Registers the callback for the OmpFork snap event.

### Parameters

<code>snapReaderCallbacks</code>	Struct for all callbacks.
<code>ompForkCallback</code>	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

**J.29.3.17 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetOmpTaskCreateCallback**  
( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`  
`OTF2_SnapReaderCallback_OmpTaskCreate ompTaskCreateCallback`  
)

Registers the callback for the OmpTaskCreate snap event.

**Parameters**

<code>snapReaderCallbacks</code>	Struct for all callbacks.
<code>ompTaskCreateCallback</code>	Function which should be called for all OmpTaskCreate events.

**Since**

Version 1.2

**Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.29.3.18 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetOmpTaskSwitchCallback**  
( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`  
`OTF2_SnapReaderCallback_OmpTaskSwitch ompTaskSwitchCallback`  
)

Registers the callback for the OmpTaskSwitch snap event.

**Parameters**

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>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

**J.29.3.19 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

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

## APPENDIX J. FILE DOCUMENTATION

**J.29.3.20 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetParameterStringCallback**  
( OTF2\_SnapReaderCallbacks \* *snapReaderCallbacks*,  
OTF2\_SnapReaderCallback\_ParameterString *parameterStringCallback*  
)

Registers the callback for the ParameterString snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterStringCallback</i>	Function which should be called for all ParameterString events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid *defReaderCallbacks* argument

**J.29.3.21 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_-**  
**SetParameterUnsignedIntCallback** ( OTF2\_SnapReaderCallbacks  
\* *snapReaderCallbacks*, OTF2\_SnapReaderCallback\_-  
ParameterUnsignedInt *parameterUnsignedIntCallback*  
)

Registers the callback for the ParameterUnsignedInt snap event.

### Parameters

<i>snapReaderCallbacks</i>	Struct for all callbacks.
<i>parameterUnsignedIntCallback</i>	Function which should be called for all ParameterUnsignedInt events.

## J.29 OTF2\_SnapReaderCallbacks.h File Reference

---

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

### J.29.3.22 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetSnapshotEndCallback

( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`

`OTF2_SnapReaderCallback_SnapshotEnd snapshotEndCallback )`

Registers the callback for the SnapshotEnd snap event.

### Parameters

<code>snapReaderCallbacks</code>	Struct for all callbacks.
<code>snapshotEndCallback</code>	Function which should be called for all SnapshotEnd events.

### Since

Version 1.2

### Returns

**OTF2\_SUCCESS** if successful

**OTF2\_ERROR\_INVALID\_ARGUMENT** for an invalid `defReaderCallbacks` argument

### J.29.3.23 OTF2\_ErrorCode OTF2\_SnapReaderCallbacks\_SetSnapshotStartCallback

( `OTF2_SnapReaderCallbacks * snapReaderCallbacks,`

`OTF2_SnapReaderCallback_SnapshotStart snapshotStartCallback )`

Registers the callback for the SnapshotStart snap event.

### Parameters

<code>snapReaderCallbacks</code>	Struct for all callbacks.
----------------------------------	---------------------------

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>snapshot-StartCallback</i>	Function which should be called for all SnapshotStart events.
-------------------------------	---

### **Since**

Version 1.2

### **Returns**

***OTF2\_SUCCESS*** if successful

***OTF2\_ERROR\_INVALID\_ARGUMENT*** for an invalid `defReaderCallbacks` argument

**J.29.3.24 OTF2\_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

## **J.30 OTF2\_SnapWriter.h File Reference**

This lowest user-visible layer provides write routines to write snapshot records for a single location.

```
#include <stdint.h>
```

## J.30 OTF2\_SnapWriter.h File Reference

---

```
#include <otf2/OTF2_ErrorCodes.h>
#include <otf2/OTF2_Events.h>
#include <otf2/OTF2_AttributeList.h>
```

### Typedefs

- **typedef struct OTF2\_SnapWriter\_struct OTF2\_SnapWriter**

*Keeps all necessary information about the snap writer. See OTF2\_SnapWriter\_struct for detailed information.*

### Functions

- **OTF2\_ErrorCode OTF2\_SnapWriter\_Enter (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, OTF2\_RegionRef region)**

*Records an Enter snapshot record.*

- **OTF2\_ErrorCode OTF2\_SnapWriter\_GetLocationID (const OTF2\_SnapWriter \*writer, OTF2\_LocationRef \*locationID)**

*Function to get the location ID of a snap writer object.*

- **OTF2\_ErrorCode OTF2\_SnapWriter\_MeasurementOnOff (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, OTF2\_MeasurementMode measurementMode)**

*Records an MeasurementOnOff snapshot record.*

- **OTF2\_ErrorCode OTF2\_SnapWriter\_Metric (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, OTF2\_MetricRef metric, uint8\_t numberOfMetrics, const OTF2\_Type \*typeIDs, const OTF2\_MetricValue \*metricValues)**

*Records an Metric snapshot record.*

- **OTF2\_ErrorCode OTF2\_SnapWriter\_MpiCollectiveBegin (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime)**

*Records an MpiCollectiveBegin snapshot record.*

- **OTF2\_ErrorCode OTF2\_SnapWriter\_MpiCollectiveEnd (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, OTF2\_CollectiveOp collectiveOp, OTF2\_CommRef communicator, uint32\_t root, uint64\_t sizeSent, uint64\_t sizeReceived)**

*Records an MpiCollectiveEnd snapshot record.*

---

## **APPENDIX J. FILE DOCUMENTATION**

- **OTF2\_ErrorCode OTF2\_SnapWriter\_MpiIrecv (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint32\_t sender, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID)**

*Records an MpiIrecv snapshot record.*
- **OTF2\_ErrorCode OTF2\_SnapWriter\_MpiIrecvRequest (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint64\_t requestID)**

*Records an MpiIrecvRequest snapshot record.*
- **OTF2\_ErrorCode OTF2\_SnapWriter\_MpiIsend (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID)**

*Records an MpiIsend snapshot record.*
- **OTF2\_ErrorCode OTF2\_SnapWriter\_MpiIsendComplete (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint64\_t requestID)**

*Records an MpiIsendComplete snapshot record.*
- **OTF2\_ErrorCode OTF2\_SnapWriter\_MpiRecv (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint32\_t sender, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)**

*Records an MpiRecv snapshot record.*
- **OTF2\_ErrorCode OTF2\_SnapWriter\_MpiSend (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint32\_t receiver, OTF2\_CommRef communicator, uint32\_t msgTag, uint64\_t msgLength)**

*Records an MpiSend snapshot record.*
- **OTF2\_ErrorCode OTF2\_SnapWriter\_OmpAcquireLock (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint32\_t lockID, uint32\_t acquisitionOrder)**

*Records an OmpAcquireLock snapshot record.*
- **OTF2\_ErrorCode OTF2\_SnapWriter\_OmpFork (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint32\_t numberOfRequestedThreads)**

*Records an OmpFork snapshot record.*
- **OTF2\_ErrorCode OTF2\_SnapWriter\_OmpTaskCreate (OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint64\_t taskID)**

*Records an OmpTaskCreate snapshot record.*

## J.30 OTF2\_SnapWriter.h File Reference

---

- `OTF2_ErrorCode OTF2_SnapWriter_OmpTaskSwitch (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, uint64_t taskID)`

*Records an OmpTaskSwitch snapshot record.*
- `OTF2_ErrorCode OTF2_SnapWriter_ParameterInt (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, int64_t value)`

*Records an ParameterInt snapshot record.*
- `OTF2_ErrorCode OTF2_SnapWriter_ParameterString (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, OTF2StringRef string)`

*Records an ParameterString snapshot record.*
- `OTF2_ErrorCode OTF2_SnapWriter_ParameterUnsignedInt (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime, OTF2_ParameterRef parameter, uint64_t value)`

*Records an ParameterUnsignedInt snapshot record.*
- `OTF2_ErrorCode OTF2_SnapWriter_SnapshotEnd (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, uint64_t contReadPos)`

*Records an SnapshotEnd snapshot record.*
- `OTF2_ErrorCode OTF2_SnapWriter_SnapshotStart (OTF2_SnapWriter *writer, OTF2_AttributeList *attributeList, OTF2_TimeStamp snapTime, uint64_t numberOfRecords)`

*Records an SnapshotStart snapshot record.*

### J.30.1 Detailed Description

This lowest user-visible layer provides write routines to write snapshot records for a single location.

#### Source Template:

*templates/OTF2\_SnapWriter.tmpl.h*

### J.30.2 Typedef Documentation

#### J.30.2.1 `typedef struct OTF2_SnapWriter_struct OTF2_SnapWriter`

Keeps all necessary information about the snap writer. See `OTF2_SnapWriter_struct` for detailed information.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.2

#### **J.30.3 Function Documentation**

**J.30.3.1 OTF2\_ErrorCode OTF2\_SnapWriter\_Enter ( OTF2\_SnapWriter \* writer,  
OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp snapTime,  
OTF2\_TimeStamp origEventTime, OTF2\_RegionRef region )**

Records an Enter snapshot record.

This record exists for each *Enter* event where the corresponding *Leave* event did not occur before the snapshot.

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happened.
<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.

**J.30.3.2 OTF2\_ErrorCode OTF2\_SnapWriter\_GetLocationID ( const  
OTF2\_SnapWriter \* writer, OTF2\_LocationRef \* locationID )**

Function to get the location ID of a snap writer object.

### **Parameters**

<i>writer</i>	Snap writer object of interest
<i>locationID</i>	Pointer to a variable where the ID is returned in

## J.30 OTF2\_SnapWriter.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.3 OTF2\_ErrorCode OTF2\_SnapWriter\_MeasurementOnOff (**  
*OTF2\_SnapWriter \* writer, OTF2\_AttributeList \* attributeList,*  
*OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime,*  
***OTF2\_MeasurementMode measurementMode )***

Records an MeasurementOnOff snapshot record.

The last occurrence of an *MeasurementOnOff* event of this location, if any.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent- Time</i>	The original time this event happened.
<i>measure- mentMode</i>	Is the measurement turned on ( <i>OTF2_MEASUREMENT_ON</i> ) or off ( <i>OTF2_MEASUREMENT_OFF</i> )?

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.4 OTF2\_ErrorCode OTF2\_SnapWriter\_Metric ( OTF2\_SnapWriter \* writer,**  
*OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp snapTime,*  
*OTF2\_TimeStamp origEventTime, OTF2\_MetricRef metric, uint8\_t*  
*numberOfMetrics, const OTF2\_Type \* typeIDs, const OTF2\_MetricValue \**  
***metricValues )***

Records an Metric snapshot record.

This record exists for each referenced metric class or metric instance event this location recorded metrics before and provides the last known recorded metric values.

## **APPENDIX J. FILE DOCUMENTATION**

---

As an exception for metric classes where the metric mode denotes an [\*OTF2\\_METRIC\\_VALUE\\_RELATIVE\*](#) mode the value indicates the accumulation of all previous metric values recorded.

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEventTime</i>	The original time this event happened.
<i>metric</i>	Could be a metric class or a metric instance. References a <a href="#"><i>MetricClass</i></a> , or a <a href="#"><i>MetricInstance</i></a> definition and will be mapped to the global definition if a mapping table of type <a href="#"><i>OTF2_MAPPING_METRIC</i></a> is available.
<i>numberOfMetrics</i>	Number of metrics with in the set.
<i>typeIDs</i>	List of metric types.
<i>metricValues</i>	List of metric values.

### **Since**

Version 1.2

### **Returns**

[\*OTF2\\_SUCCESS\*](#) if successful, an error code if an error occurs.

**J.30.3.5 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>origEventTime</i>	The original time this event happened.

## J.30 OTF2\_SnapWriter.h File Reference

---

### Since

Version 1.2

### Returns

`OTF2_SUCCESS` if successful, an error code if an error occurs.

```
J.30.3.6 OTF2_ErrorCode OTF2_SnapWriter_MpiCollectiveEnd (
    OTF2_SnapWriter * writer, OTF2_AttributeList * attributeList,
    OTF2_TimeStamp snapTime, OTF2_TimeStamp origEventTime,
    OTF2_CollectiveOp collectiveOp, OTF2_CommRef communicator,
    uint32_t root, uint64_t sizeSent, uint64_t sizeReceived )
```

Records an MpiCollectiveEnd snapshot record.

Indicates that this location completed a collective operation locally but not all of the participating locations completed the operation yet. The corresponding *MpiCollectiveBeginSaps* record is still in the snapshot though.

### Parameters

<code>writer</code>	Writer object.
<code>attributeList</code>	Generic attributes for the record.
<code>snapTime</code>	Snapshot time.
<code>origEvent-Time</code>	The original time this event happened.
<code>collectiveOp</code>	Determines which collective operation it is.
<code>communicator</code>	Communicator References a <code>Comm</code> definition and will be mapped to the global definition if a mapping table of type <code>OTF2_MAPPING_COMM</code> is available.
<code>root</code>	MPI rank of root in <code>communicator</code> .
<code>sizeSent</code>	Size of the sent message.
<code>sizeReceived</code>	Size of the received message.

### Since

Version 1.2

### Returns

`OTF2_SUCCESS` if successful, an error code if an error occurs.

## **APPENDIX J. FILE DOCUMENTATION**

---

**J.30.3.7 OTF2\_ErrorCode OTF2\_SnapWriter\_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, uint64\_t requestID )**

Records an MpiIrecv snapshot record.

This record exists for each *MpiIrecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *MpiIsendComplete* event. Or an *MpiIrecvRequest* occurred before this event but the corresponding *MpiIrecv* event did not occurred before this snapshot. In this case the message matching couldn't performed yet, because the envelope of the ongoing *MpiIrecvRequest* is not yet known.

### **Parameters**

<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>sender</i>	MPI rank of sender in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.8 OTF2\_ErrorCode OTF2\_SnapWriter\_MpiRecvRequest ( 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

## J.30 OTF2\_SnapWriter.h File Reference

---

in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. This could either be an *MpiRecv* or an *MpiIrecv* event.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.
<i>requestID</i>	ID of the requested receive

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.9 OTF2\_ErrorCode OTF2\_SnapWriter\_Mpilsend ( OTF2\_SnapWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp snapTime,  
OTF2\_TimeStamp origEventTime, uint32\_t receiver, OTF2\_CommRef  
communicator, uint32\_t msgTag, uint64\_t msgLength, uint64\_t requestID )**

Records an MpiIsend snapshot record.

This record exists for each *MpiIsend* event where an corresponding *MpiIsendComplete* or *MpiRequestCancelled* event did not occur on this location before the snapshot. Or the corresponding *MpiIsendComplete* did occurred (the *MpiIsendCompleteSnap* record exists in the snapshot) but the matching receive message event did not occur on the remote location before the snapshot. (This could either be an *MpiRecv* or an *MpiIrecv* event.)

### Parameters

<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>receiver</i>	MPI rank of receiver in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.

## **APPENDIX J. FILE DOCUMENTATION**

---

<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length
<i>requestID</i>	ID of the related request

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.10 OTF2\_ErrorCode OTF2\_SnapWriter\_MpiSendComplete (**  
**OTF2\_SnapWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint64\_t**  
**requestID )**

Records an *MpiSendComplete* snapshot record.

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.) .

### **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>requestID</i>	ID of the related request

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## J.30 OTF2\_SnapWriter.h File Reference

---

**J.30.3.11 OTF2\_ErrorCode OTF2\_SnapWriter\_MpiRecv ( OTF2\_SnapWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp snapTime,  
OTF2\_TimeStamp origEventTime, uint32\_t sender, OTF2\_CommRef  
communicator, uint32\_t msgTag, uint64\_t msgLength )**

Records an MpiRecv snapshot record.

This record exists for each *MpiRecv* event where the matching send message event did not occur on the remote location before the snapshot. This could either be an *MpiSend* or an *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>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>sender</i>	MPI rank of sender in <i>communicator</i> .
<i>communicator</i>	Communicator ID. References a <i>Comm</i> definition and will be mapped to the global definition if a mapping table of type <i>OTF2_MAPPING_COMM</i> is available.
<i>msgTag</i>	Message tag
<i>msgLength</i>	Message length

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.12 OTF2\_ErrorCode OTF2\_SnapWriter\_MpiSend ( OTF2\_SnapWriter \*  
writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp snapTime,  
OTF2\_TimeStamp origEventTime, uint32\_t receiver, OTF2\_CommRef  
communicator, uint32\_t msgTag, uint64\_t msgLength )**

Records an MpiSend snapshot record.

This record exists for each *MpiSend* event where the matching receive message event did not occur on the remote location before the snapshot. This could either

## APPENDIX J. FILE DOCUMENTATION

---

be an *MpiRecv* or an *MpiIrecv* event. Note that it may so, that a previous *MpiIsend* with the same envelope than this one is neither completed nor 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 happened.
<i>receiver</i>	MPI rank of receiver 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 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.30.3.13 OTF2\_ErrorCode OTF2\_SnapWriter\_OmpAcquireLock (**  
**OTF2\_SnapWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint32\_t**  
**lockID, uint32\_t acquisitionOrder )**

Records an OmpAcquireLock snapshot record.

This record exists for each *OmpAcquireLock* event where the corresponding *OmpReleaseLock* did not occurred before this snapshot yet.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.
<i>lockID</i>	ID of the lock.

## J.30 OTF2\_SnapWriter.h File Reference

---

<i>acquisitionOrder</i>	A monotonically increasing number to determine the order of lock acquisitions (with unsynchronized clocks this is otherwise not possible). Corresponding acquire-release events have same number.
-------------------------	---

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.14 *OTF2\_ErrorCode OTF2\_SnapWriter\_OmpFork( OTF2\_SnapWriter \*writer, OTF2\_AttributeList \*attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, uint32\_t numberOfRequestedThreads )***

Records an OmpFork snapshot record.

This record exists for each *OmpFork* event where the corresponding *OmpJoin* did not occurred before this snapshot.

### Parameters

<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>numberOfRequestedThreads</i>	Requested size of the team.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

## APPENDIX J. FILE DOCUMENTATION

**J.30.3.15 OTF2\_ErrorCode OTF2\_SnapWriter\_OmpTaskCreate ( OTF2\_SnapWriter  
\* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp  
snapTime, OTF2\_TimeStamp origEventTime, uint64\_t taskID )**

Records an OmpTaskCreate snapshot record.

This record exists for each *OmpTaskCreate* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.
<i>taskID</i>	Identifier of the newly created task instance.

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.16 OTF2\_ErrorCode OTF2\_SnapWriter\_OmpTaskSwitch ( OTF2\_SnapWriter  
\* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp  
snapTime, OTF2\_TimeStamp origEventTime, uint64\_t taskID )**

Records an OmpTaskSwitch snapshot record.

This record exists for each *OmpTaskSwitch* event where the corresponding *OmpTaskComplete* event did not occurred before this snapshot. Neither on this location nor on any other location in the current thread team.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.
<i>taskID</i>	Identifier of the now active task instance.

## J.30 OTF2\_SnapWriter.h File Reference

---

### Since

Version 1.2

### Returns

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.30.3.17 OTF2\_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.

**J.30.3.18 OTF2\_ErrorCode OTF2\_SnapWriter\_ParameterString ( OTF2\_SnapWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime, OTF2\_ParameterRef parameter, OTF2StringRef string )**

Records an ParameterString snapshot record.

## **APPENDIX J. FILE DOCUMENTATION**

---

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.
<i>parameter</i>	Parameter ID. References a <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.

**J.30.3.19 OTF2\_ErrorCode OTF2\_SnapWriter\_ParameterUnsignedInt (**  
**OTF2\_SnapWriter \* writer, OTF2\_AttributeList \* attributeList,**  
**OTF2\_TimeStamp snapTime, OTF2\_TimeStamp origEventTime,**  
**OTF2\_ParameterRef parameter, uint64\_t value )**

Records a ParameterUnsignedInt snapshot record.

This record must be included in the snapshot until the leave event for the enter event occurs which has the greatest timestamp less or equal the timestamp of this record.

### **Parameters**

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>origEvent-Time</i>	The original time this event happened.

## J.30 OTF2\_SnapWriter.h File Reference

---

<i>parameter</i>	Parameter ID. References a <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.

**J.30.3.20 OTF2\_ErrorCode OTF2\_SnapWriter\_SnapshotEnd ( OTF2\_SnapWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp snapTime, uint64\_t contReadPos )**

Records an SnapshotEnd snapshot record.

This record marks the end of a snapshot. It contains the position to continue reading in the event trace for this location. Use [OTF2\\_EvtReader\\_Seek](#) with *contReadPos* as the position.

### Parameters

<i>writer</i>	Writer object.
<i>attributeList</i>	Generic attributes for the record.
<i>snapTime</i>	Snapshot time.
<i>contReadPos</i>	Position to continue reading in the event trace.

### Since

Version 1.2

### Returns

[OTF2\\_SUCCESS](#) if successful, an error code if an error occurs.

**J.30.3.21 OTF2\_ErrorCode OTF2\_SnapWriter\_SnapshotStart ( OTF2\_SnapWriter \* writer, OTF2\_AttributeList \* attributeList, OTF2\_TimeStamp snapTime, uint64\_t numberOfRecords )**

Records an SnapshotStart snapshot record.

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.

## J.31 OTF2\_Thumbnail.h File Reference

This lowest user-visible layer provides write routines to read and write thumbnail data.

```
#include <stdint.h>
#include <otf2/OTF2_GeneralDefinitions.h>
```

### Typedefs

- **typedef struct OTF2\_ThumbReader\_struct OTF2\_ThumbReader**  
*Keeps all necessary information about the event reader. See OTF2\_ThumbReader\_struct for detailed information.*
- **typedef struct OTF2\_ThumbWriter\_struct OTF2\_ThumbWriter**  
*Keeps all necessary information about the thumb writer. See OTF2\_ThumbWriter\_struct for detailed information.*

## J.31 OTF2\_Thumbnail.h File Reference

---

### Functions

- `OTF2_ErrorCode OTF2_ThumbReader_GetHeader (OTF2_ThumbReader *reader, char **const name, char **const description, OTF2_ThumbnailType *type, uint32_t *numberOfSamples, uint32_t *numberOfMetrics, uint64_t **refsToDefs)`

*Reads a thumbnail header.*

- `OTF2_ErrorCode OTF2_ThumbReader_ReadSample (OTF2_ThumbReader *reader, uint64_t *baseline, uint32_t numberOfMetrics, uint64_t *metricSamples)`

*Reads a thumbnail sample.*

- `OTF2_ErrorCode OTF2_ThumbWriter_WriteSample (OTF2_ThumbWriter *writer, uint64_t baseline, uint32_t numberOfMetrics, const uint64_t *metricSamples)`

*Writes a thumbnail sample.*

#### J.31.1 Detailed Description

This lowest user-visible layer provides write routines to read and write thumbnail data.

#### J.31.2 Function Documentation

##### J.31.2.1 `OTF2_ErrorCode OTF2_ThumbReader_GetHeader ( OTF2_ThumbReader * reader, char **const name, char **const description, OTF2_ThumbnailType * type, uint32_t * numberOfSamples, uint32_t * numberOfMetrics, uint64_t ** refsToDefs )`

Reads a thumbnail header.

A thumbnail header contains some meta information for a thumbnail.

#### Parameters

<code>reader</code>	Reader object.
<code>name</code>	Name of thumbnail.
<code>description</code>	Description of thumbnail.
<code>type</code>	Type of thumbnail.
<code>numberOfSamples</code>	Number of samples.
<code>numberOfMetrics</code>	Number of metrics.

## **APPENDIX J. FILE DOCUMENTATION**

---

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.31.2.2 OTF2\_ErrorCode OTF2\_ThumbReader\_ReadSample (**  
**OTF2\_ThumbReader \* reader, uint64\_t \* baseline, uint32\_t**  
**numberOfMetrics, uint64\_t \* metricSamples )**

Reads a thumbnail sample.

### **Parameters**

<i>reader</i>	Reader object.
<i>baseline</i>	Baseline for this sample. If zero, the baseline is the sum of all metric values in this sample.
<i>numberOfMetrics</i>	Number of metric sample values.
<i>metricSamples</i>	Metric sample values.

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

**J.31.2.3 OTF2\_ErrorCode OTF2\_ThumbWriter\_WriteSample (**  
**OTF2\_ThumbWriter \* writer, uint64\_t baseline, uint32\_t numberOfMetrics,**  
**const uint64\_t \* metricSamples )**

Writes a thumbnail sample.

### **Parameters**

<i>writer</i>	Writer object.
<i>baseline</i>	Baseline for this sample. If zero, the baseline is the sum of all metric values in this sample.
<i>numberOfMetrics</i>	Number of metric sample values.

## **J.31 OTF2\_Thumbnail.h File Reference**

---

<i>metricSamples</i>	Metric sample values.
----------------------	-----------------------

### **Since**

Version 1.2

### **Returns**

*OTF2\_SUCCESS* if successful, an error code if an error occurs.

# Index

otf2.h, [95](#)  
OTF2\_ABORT  
    OTF2\_ErrorCodes.h, [221](#)  
OTF2\_Archive.h  
    OTF2\_MASTER, [102](#)  
    OTF2\_SLAVE, [102](#)  
OTF2\_BASE\_BINARY  
    OTF2\_Definitions.h, [163](#)  
OTF2\_BASE\_DECIMAL  
    OTF2\_Definitions.h, [163](#)  
OTF2\_CALLBACK\_INTERRUPT  
    OTF2\_GeneralDefinitions.h, [363](#)  
OTF2\_CALLBACK\_SUCCESS  
    OTF2\_GeneralDefinitions.h, [363](#)  
OTF2\_COLLECTIVE\_OP\_ALLOCATE  
    OTF2\_Events.h, [228](#)  
OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE  
    OTF2\_Events.h, [228](#)  
OTF2\_COLLECTIVE\_OP\_CREATE\_HANDLE\_AND\_ALLOCATE  
    OTF2\_Events.h, [228](#)  
OTF2\_COLLECTIVE\_OP\_DEALLOCATE  
    OTF2\_Events.h, [228](#)  
OTF2\_COLLECTIVE\_OP\_DESTROY\_HANDLE  
    OTF2\_Events.h, [228](#)  
OTF2\_COLLECTIVE\_OP\_DESTROY\_HANDLE\_AND\_DEALLOCATE  
    OTF2\_Events.h, [228](#)  
OTF2\_COMPRESSION\_NONE  
    OTF2\_GeneralDefinitions.h, [364](#)  
OTF2\_COMPRESSION\_UNDEFINED  
    OTF2\_GeneralDefinitions.h, [364](#)  
OTF2\_COMPRESSION\_ZLIB  
    OTF2\_GeneralDefinitions.h, [364](#)  
OTF2\_Definitions.h  
OTF2\_BASE\_BINARY, [163](#)  
OTF2\_BASE\_DECIMAL, [163](#)  
OTF2\_GROUP\_FLAG\_GLOBAL\_MEMBERS, [161](#)  
OTF2\_GROUP\_FLAG\_NONE, [161](#)  
OTF2\_GROUP\_TYPE\_COMM\_GROUP, [162](#)  
OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS, [162](#)  
OTF2\_GROUP\_TYPE\_COMM\_SELF, [162](#)  
OTF2\_GROUP\_TYPE\_LOCATIONS, [161](#)  
OTF2\_GROUP\_TYPE\_METRIC, [161](#)  
OTF2\_GROUP\_TYPE\_REGIONS, [161](#)  
OTF2\_GROUP\_TYPE\_UNKNOWN, [161](#)  
OTF2\_LOCATION\_GROUP\_TYPE\_PROCESS, [162](#)  
OTF2\_LOCATION\_GROUP\_TYPE\_UNKNOWN, [162](#)  
OTF2\_LOCATION\_TYPE\_CPU\_THREAD, [162](#)  
OTF2\_LOCATION\_TYPE\_GPU, [163](#)  
OTF2\_LOCATION\_TYPE\_METRIC, [163](#)  
OTF2\_LOCATION\_TYPE\_UNKNOWN, [162](#)  
OTF2\_METRIC\_ABSOLUTE\_LAST, [163](#)  
OTF2\_METRIC\_ABSOLUTE\_NEXT, [163](#)  
OTF2\_METRIC\_ABSOLUTE\_POINT, [163](#)

## INDEX

---

OTF2_METRIC_ACCUMULATED_- LAST, 163	OTF2_PARAMETER_TYPE_UINT64, 166
OTF2_METRIC_ACCUMULATED_- NEXT, 163	OTF2_RECORDER_KIND_ABSTRACT, 166
OTF2_METRIC_ACCUMULATED_- POINT, 163	OTF2_RECORDER_KIND_CPU, 166 OTF2_RECORDER_KIND_GPU, 166
OTF2_METRIC_ACCUMULATED_- START, 163	OTF2_RECORDER_KIND_UNKNOWN, 166
OTF2_METRIC_ASYNCRONOUS, 164	OTF2_REGION_FLAG_DYNAMIC, 167
OTF2_METRIC_RELATIVE_LAST, 163	OTF2_REGION_FLAG_NONE, 167 OTF2_REGION_FLAG_PHASE, 167
OTF2_METRIC_RELATIVE_NEXT, 163	OTF2_REGION_ROLE_ARTIFICIAL, 168
OTF2_METRIC_RELATIVE_POINT, 163	OTF2_REGION_ROLE_ATOMIC, 168
OTF2_METRIC_SYNCHRONOUS, 164	OTF2_REGION_ROLE_BARRIER, 168
OTF2_METRIC_SYNCHRONOUS_- STRICT, 164	OTF2_REGION_ROLE_CODE, 167 OTF2_REGION_ROLE_COLL_ALL2ALL, 168
OTF2_METRIC_TIMING_LAST, 165	OTF2_REGION_ROLE_COLL_ALL2ONE, 168
OTF2_METRIC_TIMING_MASK, 165	OTF2_REGION_ROLE_COLL_ONE2ALL, 168
OTF2_METRIC_TIMING_NEXT, 165	OTF2_REGION_ROLE_COLL_OTHER, 168
OTF2_METRIC_TIMING_POINT, 164	OTF2_REGION_ROLE_CRITICAL, 168
OTF2_METRIC_TIMING_START, 164	OTF2_REGION_ROLE_CRITICAL_- SBLOCK, 168
OTF2_METRIC_TYPE_OTHER, 165	OTF2_REGION_ROLE_DATA_TRANSFER, 168
OTF2_METRIC_TYPE_PAPI, 165	OTF2_REGION_ROLE_FILE_IO, 168 OTF2_REGION_ROLE_FLUSH, 168
OTF2_METRIC_TYPE_RUSAGE, 165	OTF2_REGION_ROLE_FUNCTION, 167
OTF2_METRIC_TYPE_USER, 165	OTF2_REGION_ROLE_IMPLICIT_- BARRIER, 168
OTF2_METRIC_VALUE_ABSOLUTE, 165	OTF2_REGION_ROLE_LOOP, 167
OTF2_METRIC_VALUE_ACCUMULATE, 165	OTF2_REGION_ROLE_MASTER, 168
OTF2_METRIC_VALUE_MASK, 166	OTF2_REGION_ROLE_ORDERED, 168
OTF2_METRIC_VALUE_RELATIVE, 165	
OTF2_PARAMETER_TYPE_INT64, 166	
OTF2_PARAMETER_TYPE_STRING, 166	

---

## INDEX

OTF2\_REGION\_ROLE\_ORDERED\_- OTF2\_SYSTEM\_TREE\_DOMAIN\_-  
SBLOCK, 168 SOCKET, 169  
OTF2\_REGION\_ROLE\_PARALLELOTF2\_DEPRECATED  
167 OTF2\_ErrorCodes.h, 221  
OTF2\_REGION\_ROLE\_POINT2POINTOTF2\_ERROR\_DUPLICATE\_MAPPING\_-  
168 TABLE  
OTF2\_REGION\_ROLE\_RMA, 168 OTF2\_ErrorCodes.h, 224  
OTF2\_REGION\_ROLE\_SECTION, OTF2\_ERROR\_E2BIG  
167 OTF2\_ErrorCodes.h, 221  
OTF2\_REGION\_ROLE\_SECTIONS,OTF2\_ERROR\_EACCES  
167 OTF2\_ErrorCodes.h, 221  
OTF2\_REGION\_ROLE\_SINGLE, 167OTF2\_ERROR\_EADDRNOTAVAIL  
OTF2\_REGION\_ROLE\_SINGLE\_- OTF2\_ErrorCodes.h, 221  
SBLOCK, 167 OTF2\_ERROR\_EAFNOSUPPORT  
OTF2\_REGION\_ROLE\_TASK, 168 OTF2\_ErrorCodes.h, 221  
OTF2\_REGION\_ROLE\_TASK\_CREATEOTF2\_ERROR\_EAGAIN  
168 OTF2\_ErrorCodes.h, 222  
OTF2\_REGION\_ROLE\_TASK\_WAIT, OTF2\_ERROR\_EALREADY  
168 OTF2\_ErrorCodes.h, 222  
OTF2\_REGION\_ROLE\_UNKNOWN, OTF2\_ERROR\_EBADF  
167 OTF2\_ErrorCodes.h, 222  
OTF2\_REGION\_ROLE\_WORKSHARE, OTF2\_ERROR\_EBADMSG  
167 OTF2\_ErrorCodes.h, 222  
OTF2\_REGION\_ROLE\_WRAPPER, OTF2\_ERROR\_EBUSY  
167 OTF2\_ErrorCodes.h, 222  
OTF2\_SCOPE\_GROUP, 164 OTF2\_ERROR\_ECANCELED  
OTF2\_SCOPE\_LOCATION, 164 OTF2\_ErrorCodes.h, 222  
OTF2\_SCOPE\_LOCATION\_GROUP, OTF2\_ErrorCodes.h, 222  
164 OTF2\_ERROR\_ECONNREFUSED  
OTF2\_SCOPE\_SYSTEM\_TREE\_- OTF2\_ErrorCodes.h, 222  
NODE, 164 OTF2\_ERROR\_ECONNRESET  
OTF2\_SYSTEM\_TREE\_DOMAIN\_- OTF2\_ErrorCodes.h, 222  
CACHE, 169 OTF2\_ERROR\_EDEADLK  
OTF2\_SYSTEM\_TREE\_DOMAIN\_- OTF2\_ErrorCodes.h, 222  
CORE, 169 OTF2\_ERROR\_EDESTADDRREQ  
OTF2\_SYSTEM\_TREE\_DOMAIN\_- OTF2\_ErrorCodes.h, 222  
MACHINE, 169 OTF2\_ERROR\_EDOM  
OTF2\_SYSTEM\_TREE\_DOMAIN\_- OTF2\_ErrorCodes.h, 222  
NUMA, 169 OTF2\_ERROR\_EDQUOT  
OTF2\_SYSTEM\_TREE\_DOMAIN\_- OTF2\_ErrorCodes.h, 222  
PU, 169 OTF2\_ERROR\_EEXIST  
OTF2\_SYSTEM\_TREE\_DOMAIN\_- OTF2\_ErrorCodes.h, 222  
SHARED\_MEMORY, 169 OTF2\_ERROR\_EFAULT

---

## INDEX

---

OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EFBIG  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EINPROGRESS  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EINTR  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EINVAL  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EIO  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EISCONN  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EISDIR  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_ELOOP  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EMFILE  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EMLINK  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EMSGSIZE  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_EMULTIHOP  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_ENAMETOOLONG  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_END\_OF\_BUFFER  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_END\_OF\_FUNCTION  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_ENETDOWN  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_ENETRESET  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_ENETUNREACH  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_ENFILE  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_ENOBUFS  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_ENODATA  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ERROR\_ENODEV  
    OTF2\_ErrorCodes.h, 222  
OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOENT  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOEXEC  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOLCK  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOLINK  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOMEM  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOMSG  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOPROTOOPT  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOSPC  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOSR  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOSTR  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOSYS  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOTCONN  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOTDIR  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOTEMPTY  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOTSOCK  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOTSUP  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENOTTY  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ENXIO  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EOPNOTSUPP  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EOVERFLOW  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EPERM  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EPIPE

---

## INDEX

OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EPROTO  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EPROTONOSUPPORT  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EPROTOTYPE  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ERANGE  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EROFS  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ESPIPE  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ESRCH  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ESTALE  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERRORETIME  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ETIMEDOUT  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_ETXTBSY  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ERROR\_EWOULDBLOCK  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_EXDEV  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_FILE\_CAN\_NOT\_OPENOTF2\_ERROR\_PROPERTY\_VALUE\_-  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_FILE\_COMPRESSION\_-  
    NOT\_SUPPORTED  
        OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_FILE\_INTERACTION  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INDEX\_OUT\_OF\_BOUNDSOTF2\_DEPRECATED, 221  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INTEGRITY\_FAULT  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INTERRUPTED\_BY\_-  
    CALLBACK  
        OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INVALID  
    OTF2\_ErrorCodes.h, 221  
OTF2\_ERROR\_INVALID\_ARGUMENT  
    OTF2\_ErrorCodes.h, 223  
OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INVALID\_CALL  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INVALID\_DATA  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INVALID\_FILE\_MODE\_-  
    TRANSITION  
        OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INVALID\_FILENO  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INVALID\_RECORD  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_INVALID\_SIZE\_GIVEN  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_MEM\_ALLOC\_FAILED  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_MEM\_FAULT  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_PROCESSED\_WITH\_-  
    FAULTS  
        OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_PROPERTY\_EXISTS  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_PROPERTY\_NAME\_INVALID  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_PROPERTY\_NOT\_FOUND  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_UNKNOWN\_TYPE  
    OTF2\_ErrorCodes.h, 224  
OTF2\_ERROR\_ABORT, 221  
OTF2\_ERROR\_DUPLICATE\_MAPPING\_-  
    TABLE, 224  
OTF2\_ERROR\_E2BIG, 221  
OTF2\_ERROR\_EACCES, 221  
OTF2\_ERROR\_EADDRNOTAVAIL,  
    221  
OTF2\_ERROR\_EAFNOSUPPORT,  
    221  
OTF2\_ERROR\_EAGAIN, 222

---

## INDEX

---

OTF2\_ERROR\_EALREADY, [222](#)  
OTF2\_ERROR\_EBADF, [222](#)  
OTF2\_ERROR\_EBADMSG, [222](#)  
OTF2\_ERROR\_EBUSY, [222](#)  
OTF2\_ERROR\_ECANCELED, [222](#)  
OTF2\_ERROR\_ECHILD, [222](#)  
OTF2\_ERROR\_ECONNREFUSED, [222](#)  
OTF2\_ERROR\_ECONNRESET, [222](#)  
OTF2\_ERROR\_EDEADLK, [222](#)  
OTF2\_ERROR\_EDESTADDRREQ, [222](#)  
OTF2\_ERROR\_EDOM, [222](#)  
OTF2\_ERROR\_EDQUOT, [222](#)  
OTF2\_ERROR\_EEXIST, [222](#)  
OTF2\_ERROR\_EFAULT, [222](#)  
OTF2\_ERROR\_EFBIG, [222](#)  
OTF2\_ERROR\_EINPROGRESS, [222](#)  
OTF2\_ERROR\_EINTR, [222](#)  
OTF2\_ERROR\_EINVAL, [222](#)  
OTF2\_ERROR\_EIO, [222](#)  
OTF2\_ERROR\_EISCONN, [222](#)  
OTF2\_ERROR\_EISDIR, [222](#)  
OTF2\_ERROR\_ELOOP, [222](#)  
OTF2\_ERROR\_EMFILE, [222](#)  
OTF2\_ERROR\_EMLINK, [222](#)  
OTF2\_ERROR\_EMSGSIZE, [222](#)  
OTF2\_ERROR\_EMULTIHOP, [222](#)  
OTF2\_ERROR\_ENAMETOOLONG, [222](#)  
OTF2\_ERROR\_END\_OF\_BUFFER, [224](#)  
OTF2\_ERROR\_END\_OF\_FUNCTION, [224](#)  
OTF2\_ERROR\_ENETDOWN, [222](#)  
OTF2\_ERROR\_ENETRESET, [222](#)  
OTF2\_ERROR\_ENETUNREACH, [222](#)  
OTF2\_ERROR\_ENFILE, [222](#)  
OTF2\_ERROR\_ENOBUFS, [222](#)  
OTF2\_ERROR\_ENODATA, [222](#)  
OTF2\_ERROR\_ENODEV, [223](#)  
OTF2\_ERROR\_ENOENT, [223](#)  
OTF2\_ERROR\_ENOEXEC, [223](#)  
OTF2\_ERROR\_ENOLCK, [223](#)  
OTF2\_ERROR\_ENOLINK, [223](#)  
OTF2\_ERROR\_ENOMEM, [223](#)  
OTF2\_ERROR\_ENOMSG, [223](#)  
OTF2\_ERROR\_ENOPROTOOPT, [223](#)  
OTF2\_ERROR\_ENOSPC, [223](#)  
OTF2\_ERROR\_ENOSR, [223](#)  
OTF2\_ERROR\_ENOSTR, [223](#)  
OTF2\_ERROR\_ENOSYS, [223](#)  
OTF2\_ERROR\_ENOTCONN, [223](#)  
OTF2\_ERROR\_ENOTDIR, [223](#)  
OTF2\_ERROR\_ENOTEMPTY, [223](#)  
OTF2\_ERROR\_ENOTSOCK, [223](#)  
OTF2\_ERROR\_ENOTSUP, [223](#)  
OTF2\_ERROR\_ENOTTY, [223](#)  
OTF2\_ERROR\_ENXIO, [223](#)  
OTF2\_ERROR\_EOPNOTSUPP, [223](#)  
OTF2\_ERROR\_EOVERFLOW, [223](#)  
OTF2\_ERROR\_EPERM, [223](#)  
OTF2\_ERROR\_EPIPE, [223](#)  
OTF2\_ERROR\_EPROTO, [223](#)  
OTF2\_ERROR\_EPROTONOSUPPORT, [223](#)  
OTF2\_ERROR\_EPROTOTYPE, [223](#)  
OTF2\_ERROR\_ERANGE, [223](#)  
OTF2\_ERROR\_EROFS, [223](#)  
OTF2\_ERROR\_ESPIPE, [223](#)  
OTF2\_ERROR\_ESRCH, [223](#)  
OTF2\_ERROR\_ESTALE, [223](#)  
OTF2\_ERRORETIME, [223](#)  
OTF2\_ERROR\_ETIMEDOUT, [223](#)  
OTF2\_ERROR\_ETIME, [223](#)  
OTF2\_ERROR\_ETIME, [223](#)  
OTF2\_ERROR\_ETIME, [223](#)  
OTF2\_ERROR\_EWOULDLOCK, [224](#)  
OTF2\_ERROR\_EXDEV, [224](#)  
OTF2\_ERROR\_FILE\_CAN\_NOT\_OPEN, [224](#)  
OTF2\_ERROR\_FILE\_COMPRESSION\_NOT\_SUPPORTED, [224](#)  
OTF2\_ERROR\_FILE\_INTERACTION, [224](#)  
OTF2\_ERROR\_INDEX\_OUT\_OF\_BOUNDS, [224](#)

---

## INDEX

OTF2\_ERROR\_INTEGRITY\_FAULT, 224  
OTF2\_ERROR\_INTERRUPTED\_BY\_CALLBACK, 224  
OTF2\_ERROR\_INVALID, 221  
OTF2\_ERROR\_INVALID\_ARGUMENT, 224  
OTF2\_ERROR\_INVALID\_CALL, 224  
OTF2\_ERROR\_INVALID\_DATA, 224  
OTF2\_ERROR\_INVALID\_FILE\_MODE, 224  
OTF2\_ERROR\_INVALID lineno, 224  
OTF2\_ERROR\_INVALID\_RECORD, 224  
OTF2\_ERROR\_INVALID\_SIZE\_GIVEN, 224  
OTF2\_ERROR\_MEM\_ALLOC\_FAILED, 224  
OTF2\_ERROR\_MEM\_FAULT, 224  
OTF2\_ERROR\_PROCESSED\_WITH\_FAULTS, 224  
OTF2\_ERROR\_PROPERTY\_EXISTS, 224  
OTF2\_ERROR\_PROPERTY\_NAME\_INVALID, 224  
OTF2\_ERROR\_PROPERTY\_NOT\_FOUND, 224  
OTF2\_ERROR\_PROPERTY\_VALUE\_INVALID, 224  
OTF2\_ERROR\_UNKNOWN\_TYPE, 224  
OTF2\_SUCCESS, 221  
OTF2\_WARNING, 221  
OTF2\_Events.h  
    OTF2\_COLLECTIVE\_OP\_ALLOCATE, 228  
    OTF2\_COLLECTIVE\_OP\_CREATE HANDLE, 228  
    OTF2\_COLLECTIVE\_OP\_CREATE HANDLE\_AND\_ALLOCATE, 228  
    OTF2\_COLLECTIVE\_OP\_DEALLOCATE, 228  
        OTF2\_COLLECTIVE\_OP\_DESTROY HANDLE, 228  
        OTF2\_COLLECTIVE\_OP\_DESTROY HANDLE\_AND\_DEALLOCATE, 228  
        OTF2\_LOCK\_EXCLUSIVE, 228  
        OTF2\_LOCK\_SHARED, 228  
        OTF2\_MEASUREMENT\_OFF, 229  
        OTF2\_MEASUREMENT\_ON, 229  
        OTF2\_RMA\_SYNC\_LEVEL\_MEMORY\_TRANSITION, 229  
        OTF2\_RMA\_SYNC\_LEVEL\_NONE, 229  
        OTF2\_RMA\_SYNC\_LEVEL\_PROCESS, 229  
        OTF2\_RMA\_SYNC\_TYPE\_MEMORY, 230  
        OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_IN, 230  
        OTF2\_RMA\_SYNC\_TYPE\_NOTIFY\_OUT, 230  
        OTF2\_FILEMODE MODIFY  
        OTF2\_FILEMODE\_READ  
        OTF2\_FILEMODE\_WRITE  
        OTF2\_FILETYPE\_ANCHOR  
        OTF2\_FILETYPE\_EVENTS  
        OTF2\_FILETYPE\_GLOBAL\_DEFS  
        OTF2\_FILETYPE\_LOCAL\_DEFS  
        OTF2\_GENERALDEFINITIONS\_H, 365  
        OTF2\_FILETYPE\_MARKER  
        OTF2\_GENERALDEFINITIONS\_H, 365  
        OTF2\_FILETYPE\_SNAPSHOTS  
        OTF2\_GENERALDEFINITIONS\_H, 365  
        OTF2\_FILETYPE\_THUMBNAIL  
        OTF2\_GENERALDEFINITIONS\_H, 365  
        OTF2\_FLUSH  
        OTF2\_GENERALDEFINITIONS\_H, 365  
        OTF2\_GENERALDEFINITIONS\_H, 365

## INDEX

---

OTF2\_CALLBACK\_INTERRUPT, 363  
OTF2\_CALLBACK\_SUCCESS, 363  
OTF2\_COMPRESSION\_NONE, 364  
OTF2\_COMPRESSION\_UNDEFINED, 364  
OTF2\_COMPRESSION\_ZLIB, 364  
OTF2\_FILEMODE MODIFY, 364  
OTF2\_FILEMODE\_READ, 364  
OTF2\_FILEMODE\_WRITE, 364  
OTF2\_FILETYPE\_ANCHOR, 365  
OTF2\_FILETYPE\_EVENTS, 365  
OTF2\_FILETYPE\_GLOBAL\_DEFS, 365  
OTF2\_FILETYPE\_LOCAL\_DEFS, 365  
OTF2\_FILETYPE\_MARKER, 365  
OTF2\_FILETYPE\_SNAPSHOTS, 365  
OTF2\_FILETYPE\_THUMBNAIL, 365  
OTF2\_FLUSH, 365  
OTF2\_MAPPING\_ATTRIBUTE, 365  
OTF2\_MAPPING\_COMM, 366  
OTF2\_MAPPING\_GROUP, 365  
OTF2\_MAPPING\_LOCATION, 365  
OTF2\_MAPPING\_MAX, 366  
OTF2\_MAPPING\_METRIC, 365  
OTF2\_MAPPING\_PARAMETER, 366  
OTF2\_MAPPING\_REGION, 365  
OTF2\_MAPPING\_RMA\_WIN, 366  
OTF2\_MAPPING\_STRING, 365  
OTF2\_NO\_FLUSH, 365  
OTF2\_PARADIGM\_COMPILER, 366  
OTF2\_PARADIGM\_CUDA, 366  
OTF2\_PARADIGM\_MEASUREMENT SYSTEM, 366  
OTF2\_PARADIGM\_MPI, 366  
OTF2\_PARADIGM\_OPENMP, 366  
OTF2\_PARADIGM\_UNKNOWN, 366  
OTF2\_PARADIGM\_USER, 366  
OTF2\_SUBSTRATE\_NONE, 364  
OTF2\_SUBSTRATE\_POSIX, 364  
OTF2\_SUBSTRATE\_SION, 364  
OTF2\_SUBSTRATE\_UNDEFINED, 364  
OTF2\_THUMBNAIL\_TYPE\_ATTRIBUTES, 367  
OTF2\_THUMBNAIL\_TYPE\_METRIC, 367  
OTF2\_THUMBNAIL\_TYPE\_REGION, 367  
OTF2\_TYPE\_DOUBLE, 367  
OTF2\_TYPE\_FLOAT, 367  
OTF2\_TYPE\_INT16, 367  
OTF2\_TYPE\_INT32, 367  
OTF2\_TYPE\_INT64, 367  
OTF2\_TYPE\_INT8, 367  
OTF2\_TYPE\_NONE, 367  
OTF2\_TYPE\_UINT16, 367  
OTF2\_TYPE\_UINT32, 367  
OTF2\_TYPE\_UINT64, 367  
OTF2\_TYPE\_UINT8, 367  
OTF2\_GROUP\_FLAG\_GLOBAL\_MEMBERS OTF2\_Definitions.h, 161  
OTF2\_GROUP\_FLAG\_NONE OTF2\_Definitions.h, 161  
OTF2\_GROUP\_TYPE\_COMM\_GROUP OTF2\_Definitions.h, 162  
OTF2\_GROUP\_TYPE\_COMM\_LOCATIONS OTF2\_Definitions.h, 162  
OTF2\_GROUP\_TYPE\_COMM\_SELF OTF2\_Definitions.h, 162  
OTF2\_GROUP\_TYPE\_LOCATIONS OTF2\_Definitions.h, 161  
OTF2\_GROUP\_TYPE\_METRIC OTF2\_Definitions.h, 161  
OTF2\_GROUP\_TYPE\_REGIONS OTF2\_Definitions.h, 161  
OTF2\_ID\_MAP\_DENSE OTF2\_IdMap.h, 533  
OTF2\_ID\_MAP\_SPARSE OTF2\_IdMap.h, 533  
OTF2\_IdMap.h OTF2\_IdMap.h  
OTF2\_ID\_MAP\_DENSE, 533  
OTF2\_ID\_MAP\_SPARSE, 533

## INDEX

OTF2_LOCATION_GROUP_TYPE_PROCESS	OTF2_MARKER_SCOPE_LOCATION,
OTF2_Definitions.h, 162	538
OTF2_LOCATION_GROUP_TYPE_UNKNO	OTF2_MARKER_SCOPE_LOCATION_-
OTF2_Definitions.h, 162	GROUP, 538
OTF2_LOCATION_TYPE_CPU_THREAD	OTF2_MARKER_SCOPE_SYSTEM_-
OTF2_Definitions.h, 162	TREE_NODE, 538
OTF2_LOCATION_TYPE_GPU	OTF2_SEVERITY_HIGH, 539
OTF2_Definitions.h, 163	OTF2_SEVERITY_LOW, 539
OTF2_LOCATION_TYPE_METRIC	OTF2_SEVERITY_MEDIUM, 539
OTF2_Definitions.h, 163	OTF2_SEVERITY_NONE, 539
OTF2_LOCATION_TYPE_UNKNOWN	OTF2_MARKER_SCOPE_COMM
OTF2_Definitions.h, 162	OTF2_Marker.h, 538
OTF2_LOCK_EXCLUSIVE	OTF2_MARKER_SCOPE_GLOBAL
OTF2_Events.h, 228	OTF2_Marker.h, 538
OTF2_LOCK_SHARED	OTF2_MARKER_SCOPE_GROUP
OTF2_Events.h, 228	OTF2_Marker.h, 538
OTF2_MAPPING_ATTRIBUTE	OTF2_MARKER_SCOPE_LOCATION
OTF2_GeneralDefinitions.h, 365	OTF2_Marker.h, 538
OTF2_MAPPING_COMM	OTF2_MARKER_SCOPE_LOCATION_-
OTF2_GeneralDefinitions.h, 366	GROUP
OTF2_MAPPING_GROUP	OTF2_Marker.h, 538
OTF2_GeneralDefinitions.h, 365	OTF2_MAPPING_LOCATION
OTF2_MAPPING_LOCATION	OTF2_Marker.h, 538
OTF2_GeneralDefinitions.h, 365	OTF2_MAPPING_MAX
OTF2_MAPPING_MAX	OTF2_Marker.h, 538
OTF2_GeneralDefinitions.h, 366	OTF2_MASTER
OTF2_MAPPING_METRIC	OTF2_Archive.h, 102
OTF2_GeneralDefinitions.h, 365	OTF2_MEASUREMENT_OFF
OTF2_MAPPING_PARAMETER	OTF2_Events.h, 229
OTF2_GeneralDefinitions.h, 366	OTF2_MEASUREMENT_ON
OTF2_MAPPING_REGION	OTF2_Events.h, 229
OTF2_GeneralDefinitions.h, 365	OTF2_METRIC_ABSOLUTE_LAST
OTF2_MAPPING_RMA_WIN	OTF2_Definitions.h, 163
OTF2_GeneralDefinitions.h, 366	OTF2_METRIC_ABSOLUTE_NEXT
OTF2_MAPPING_STRING	OTF2_Definitions.h, 163
OTF2_GeneralDefinitions.h, 365	OTF2_METRIC_ABSOLUTE_POINT
OTF2_Marker.h	OTF2_Definitions.h, 163
OTF2_MARKER_SCOPE_COMM, 538	OTF2_METRIC_ACCUMULATED_LAST
OTF2_MARKER_SCOPE_GLOBAL, 538	OTF2_Definitions.h, 163
OTF2_MARKER_SCOPE_GROUP, 538	OTF2_METRIC_ACCUMULATED_NEXT
OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE, 538	OTF2_Definitions.h, 163
OTF2_MARKER_SCOPE_COMM, 538	OTF2_METRIC_ACCUMULATED_POINT
OTF2_MARKER_SCOPE_GLOBAL, 538	OTF2_Definitions.h, 163
OTF2_MARKER_SCOPE_GROUP, 538	OTF2_METRIC_ACCUMULATED_START
OTF2_MARKER_SCOPE_SYSTEM_TREE_NODE, 538	OTF2_Definitions.h, 163

## INDEX

---

OTF2\_METRIC\_ASYNCNCHRONOUS  
    OTF2\_Definitions.h, 164

OTF2\_METRIC\_RELATIVE\_LAST  
    OTF2\_Definitions.h, 163

OTF2\_METRIC\_RELATIVE\_NEXT  
    OTF2\_Definitions.h, 163

OTF2\_METRIC\_RELATIVE\_POINT  
    OTF2\_Definitions.h, 163

OTF2\_METRIC\_SYNCHRONOUS  
    OTF2\_Definitions.h, 164

OTF2\_METRIC\_SYNCHRONOUS\_STRICT  
    OTF2\_Definitions.h, 164

OTF2\_METRIC\_TIMING\_LAST  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_TIMING\_MASK  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_TIMING\_NEXT  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_TIMING\_POINT  
    OTF2\_Definitions.h, 164

OTF2\_METRIC\_TIMING\_START  
    OTF2\_Definitions.h, 164

OTF2\_METRIC\_TYPE\_OTHER  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_TYPE\_PAPI  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_TYPE\_RUSAGE  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_TYPE\_USER  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_VALUE\_ABSOLUTE  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_VALUE\_ACCUMULATED  
    OTF2\_Definitions.h, 165

OTF2\_METRIC\_VALUE\_MASK  
    OTF2\_Definitions.h, 166

OTF2\_METRIC\_VALUE\_RELATIVE  
    OTF2\_Definitions.h, 165

OTF2\_NO\_FLUSH  
    OTF2\_GeneralDefinitions.h, 365

OTF2\_PARADIGM\_COMPILER  
    OTF2\_GeneralDefinitions.h, 366

OTF2\_PARADIGM\_CUDA  
    OTF2\_GeneralDefinitions.h, 366

OTF2\_PARADIGM\_MEASUREMENT\_SYSTEM  
    OTF2\_GeneralDefinitions.h, 366

OTF2\_PARADIGM\_MPI  
    OTF2\_GeneralDefinitions.h, 366

OTF2\_PARADIGM\_OPENMP  
    OTF2\_GeneralDefinitions.h, 366

OTF2\_PARADIGM\_UNKNOWN  
    OTF2\_GeneralDefinitions.h, 366

OTF2\_PARADIGM\_USER  
    OTF2\_GeneralDefinitions.h, 366

OTF2\_PARAMETER\_TYPE\_INT64  
    OTF2\_Definitions.h, 166

OTF2\_PARAMETER\_TYPE\_STRING  
    OTF2\_Definitions.h, 166

OTF2\_PARAMETER\_TYPE\_UINT64  
    OTF2\_Definitions.h, 166

OTF2\_RECORDER\_KIND\_ABSTRACT  
    OTF2\_Definitions.h, 166

OTF2\_RECORDER\_KIND\_CPU  
    OTF2\_Definitions.h, 166

OTF2\_RECORDER\_KIND\_GPU  
    OTF2\_Definitions.h, 166

OTF2\_RECORDER\_KIND\_UNKNOWN  
    OTF2\_Definitions.h, 166

OTF2\_REGION\_FLAG\_DYNAMIC  
    OTF2\_Definitions.h, 167

OTF2\_REGION\_FLAG\_NONE  
    OTF2\_Definitions.h, 167

OTF2\_REGION\_FLAG\_PHASE  
    OTF2\_Definitions.h, 167

OTF2\_REGION\_ROLE\_ARTIFICIAL  
    OTF2\_Definitions.h, 168

OTF2\_REGION\_ROLE\_ATOMIC  
    OTF2\_Definitions.h, 168

OTF2\_REGION\_ROLE\_BARRIER  
    OTF2\_Definitions.h, 168

OTF2\_REGION\_ROLE\_CODE  
    OTF2\_Definitions.h, 167

OTF2\_REGION\_ROLE\_COLL\_ALL2ALL  
    OTF2\_Definitions.h, 168

OTF2\_REGION\_ROLE\_COLL\_ALL2ONE  
    OTF2\_Definitions.h, 168

OTF2\_REGION\_ROLE\_COLL\_ONE2ALL

## INDEX

OTF2_Definitions.h, 168	OTF2_Definitions.h, 168
OTF2_REGION_ROLE_COLL_OTHER OTF2_Definitions.h, 168	OTF2_REGION_ROLE_UNKNOWN OTF2_Definitions.h, 167
OTF2_REGION_ROLE_CRITICAL OTF2_Definitions.h, 168	OTF2_REGION_ROLE_WORKSHARE OTF2_Definitions.h, 167
OTF2_REGION_ROLE_CRITICAL_SBLOCKER REGION_ROLE_WRAPPER OTF2_Definitions.h, 168	OTF2_REGION_ROLE_WRAPPER OTF2_Definitions.h, 167
OTF2_REGION_ROLE_DATA_TRANSFER OTF2_RMA_SYNC_LEVEL_MEMORY OTF2_Definitions.h, 168	OTF2_Events.h, 229
OTF2_REGION_ROLE_FILE_IO OTF2_Definitions.h, 168	OTF2_RMA_SYNC_LEVEL_NONE OTF2_Events.h, 229
OTF2_REGION_ROLE_FLUSH OTF2_Definitions.h, 168	OTF2_RMA_SYNC_LEVEL_PROCESS OTF2_Events.h, 229
OTF2_REGION_ROLE_FUNCTION OTF2_Definitions.h, 167	OTF2_RMA_SYNC_TYPE_MEMORY OTF2_Events.h, 230
OTF2_REGION_ROLE_IMPLICIT_BAROTTER RMA_SYNC_TYPE_NOTIFY_- OTF2_Definitions.h, 168 IN	OTF2_EVENTS, 230
OTF2_REGION_ROLE_LOOP OTF2_Definitions.h, 167	OTF2_RMA_SYNC_TYPE_NOTIFY_- OUT
OTF2_REGION_ROLE_MASTER OTF2_Definitions.h, 168	OTF2_Events.h, 230
OTF2_REGION_ROLE_ORDERED OTF2_Definitions.h, 168	OTF2_SCOPE_GROUP OTF2_Definitions.h, 164
OTF2_REGION_ROLE_ORDERED_SBLOCKER_SCOPE_LOCATION OTF2_Definitions.h, 168	OTF2_SCOPE_LOCATION OTF2_Definitions.h, 164
OTF2_REGION_ROLE_PARALLEL OTF2_Definitions.h, 167	OTF2_SCOPE_LOCATION_GROUP OTF2_Definitions.h, 164
OTF2_REGION_ROLE_POINT2POINT OTF2_SCOPE_SYSTEM_TREE_NODE OTF2_Definitions.h, 168	OTF2_Definitions.h, 164
OTF2_REGION_ROLE_RMA OTF2_Definitions.h, 168	OTF2_SEVERITY_HIGH OTF2_Marker.h, 539
OTF2_REGION_ROLE_SECTION OTF2_Definitions.h, 167	OTF2_SEVERITY_LOW OTF2_Marker.h, 539
OTF2_REGION_ROLE_SECTIONS OTF2_Definitions.h, 167	OTF2_SEVERITY_MEDIUM OTF2_Marker.h, 539
OTF2_REGION_ROLE_SINGLE OTF2_Definitions.h, 167	OTF2_SEVERITY_NONE OTF2_Marker.h, 539
OTF2_REGION_ROLE_SINGLE_SBLOCKER SLAVE OTF2_Definitions.h, 167	OTF2_Archive.h, 102
OTF2_REGION_ROLE_TASK OTF2_Definitions.h, 168	OTF2_SUBSTRATE_NONE OTF2_GeneralDefinitions.h, 364
OTF2_REGION_ROLE_TASK_CREATEOTF2_SUBSTRATE_POSIX OTF2_Definitions.h, 168	OTF2_GeneralDefinitions.h, 364
OTF2_REGION_ROLE_TASK_WAIT OTF2_Definitions.h, 168	OTF2_SUBSTRATE_SION

## INDEX

---

OTF2_GeneralDefinitions.h, 364	OTF2_TYPE_UINT64
OTF2_SUBSTRATE_UNDEFINED	OTF2_GeneralDefinitions.h, 367
OTF2_GeneralDefinitions.h, 364	OTF2_TYPE_UINT8
OTF2_SUCCESS	OTF2_GeneralDefinitions.h, 367
OTF2_ErrorCodes.h, 221	OTF2_WARNING
OTF2_SYSTEM_TREE_DOMAIN_CACHE	OTF2_ErrorCodes.h, 221
OTF2_Definitions.h, 169	OTF2_Archive
OTF2_SYSTEM_TREE_DOMAIN_CORE	OTF2_Archive.h, 101
OTF2_Definitions.h, 169	OTF2_Archive.h, 95
OTF2_SYSTEM_TREE_DOMAIN_MACHINE	OTF2_Archive, 101
OTF2_Definitions.h, 169	OTF2_Archive_Close, 102
OTF2_SYSTEM_TREE_DOMAIN_NUMA	OTF2_Archive_CloseDefReader, 102
OTF2_Definitions.h, 169	OTF2_Archive_CloseDefWriter, 103
OTF2_SYSTEM_TREE_DOMAIN_PU	OTF2_Archive_CloseEvtReader, 103
OTF2_Definitions.h, 169	OTF2_Archive_CloseEvtWriter, 103
OTF2_SYSTEM_TREE_DOMAIN_SHARED_MEMORY	OTF2_Archive_CloseGlobalDefReader, 104
OTF2_Definitions.h, 169	OTF2_Archive_CloseGlobalEvtReader, 104
OTF2_SYSTEM_TREE_DOMAIN_SOCKET	OTF2_Archive_CloseGlobalSnapReader, 105
OTF2_Definitions.h, 169	OTF2_Archive_CloseMarkerReader, 105
OTF2_THUMBNAIL_TYPE_ATTRIBUTES	OTF2_Archive_CloseMarkerWriter, 105
OTF2_GeneralDefinitions.h, 367	OTF2_Archive_CloseSnapReader, 106
OTF2_THUMBNAIL_TYPE_METRIC	OTF2_Archive_CloseSnapWriter, 106
OTF2_GeneralDefinitions.h, 367	OTF2_Archive_CloseThumbReader, 107
OTF2_THUMBNAIL_TYPE_REGION	OTF2_Archive_GetChunkSize, 107
OTF2_GeneralDefinitions.h, 367	OTF2_Archive_GetCompression, 107
OTF2_TYPE_DOUBLE	OTF2_Archive_GetCreator, 108
OTF2_GeneralDefinitions.h, 367	OTF2_Archive_GetDefReader, 108
OTF2_TYPE_FLOAT	OTF2_Archive_GetDefWriter, 108
OTF2_GeneralDefinitions.h, 367	OTF2_Archive_GetDescription, 109
OTF2_TYPE_INT16	OTF2_Archive_GetEvtReader, 109
OTF2_GeneralDefinitions.h, 367	OTF2_Archive_GetEvtWriter, 109
OTF2_TYPE_INT32	OTF2_Archive_GetFileSubstrate, 110
OTF2_GeneralDefinitions.h, 367	OTF2_Archive_GetGlobalDefReader, 110
OTF2_TYPE_INT64	OTF2_Archive_GetGlobalDefWriter, 110
OTF2_GeneralDefinitions.h, 367	OTF2_Archive_GetGlobalEvtReader, 111
OTF2_TYPE_INT8	
OTF2_GeneralDefinitions.h, 367	
OTF2_TYPE_NONE	
OTF2_GeneralDefinitions.h, 367	
OTF2_TYPE_UINT16	
OTF2_GeneralDefinitions.h, 367	
OTF2_TYPE_UINT32	
OTF2_GeneralDefinitions.h, 367	

## INDEX

---

OTF2\_Archive\_GetGlobalSnapReader, OTF2\_CHUNK\_SIZE\_EVENTS\_-  
111 DEFAULT, 101  
OTF2\_Archive\_GetMachineName, 111 OTF2\_MasterSlaveMode, 102  
OTF2\_Archive\_GetMarkerReader, 112 OTF2\_MasterSlaveMode\_enum, 102  
OTF2\_Archive\_GetMarkerWriter, 112 OTF2\_Archive\_Close  
OTF2\_Archive\_GetMasterSlaveMode, OTF2\_Archive.h, 102  
113 OTF2\_Archive\_CloseDefReader  
OTF2\_Archive\_GetNumberOfGlobalDefinitions, OTF2\_Archive.h, 102  
113 OTF2\_Archive\_CloseDefWriter  
OTF2\_Archive\_GetNumberOfLocations, OTF2\_Archive.h, 103  
113 OTF2\_Archive\_CloseEvtReader  
OTF2\_Archive\_GetNumberOfSnapshots, OTF2\_Archive.h, 103  
114 OTF2\_Archive\_CloseEvtWriter  
OTF2\_Archive\_GetNumberOfThumbnails, OTF2\_Archive.h, 103  
114 OTF2\_Archive\_CloseGlobalDefReader  
OTF2\_ArchiveGetProperty, 114 OTF2\_Archive.h, 104  
OTF2\_ArchiveGetPropertyNames, OTF2\_Archive\_CloseGlobalEvtReader  
115 OTF2\_Archive.h, 104  
OTF2\_Archive\_GetSnapReader, 115 OTF2\_Archive\_CloseGlobalSnapReader  
OTF2\_Archive\_GetSnapWriter, 116 OTF2\_Archive.h, 105  
OTF2\_Archive\_GetThumbReader, 116 OTF2\_Archive\_CloseMarkerReader  
OTF2\_Archive\_GetThumbWriter, 116 OTF2\_Archive.h, 105  
OTF2\_Archive\_GetTraceId, 117 OTF2\_Archive\_CloseMarkerWriter  
OTF2\_Archive\_GetVersion, 117 OTF2\_Archive\_CloseSnapReader  
OTF2\_Archive\_Open, 118 OTF2\_Archive.h, 106  
OTF2\_Archive\_SetBoolProperty, 119 OTF2\_Archive\_CloseSnapWriter  
OTF2\_Archive\_SetCreator, 120 OTF2\_Archive.h, 106  
OTF2\_Archive\_SetDescription, 120 OTF2\_Archive\_CloseThumbReader  
OTF2\_Archive\_SetFileSionCallbacks, OTF2\_Archive.h, 107  
121 OTF2\_Archive\_GetChunkSize  
OTF2\_Archive\_SetFlushCallbacks, OTF2\_Archive.h, 107  
121 OTF2\_Archive\_GetCompression  
OTF2\_Archive\_SetMachineName, 121 OTF2\_Archive.h, 107  
OTF2\_Archive\_SetMasterSlaveModeOTF2\_Archive\_GetCreator  
122 OTF2\_Archive.h, 108  
OTF2\_Archive\_SetMemoryCallbacksOTF2\_Archive\_GetDefReader  
122 OTF2\_Archive.h, 108  
OTF2\_Archive\_SetNumberOfSnapshotOTF2\_Archive\_GetDefWriter  
123 OTF2\_Archive.h, 108  
OTF2\_Archive\_SetProperty, 123 OTF2\_Archive\_GetDescription  
OTF2\_Archive\_Switch FileMode, 124 OTF2\_Archive.h, 109  
OTF2\_CHUNK\_SIZE\_DEFINITIONSOTF2\_Archive\_GetEvtReader  
DEFAULT, 101 OTF2\_Archive.h, 109

## INDEX

---

OTF2\_Archive\_GetEvtWriter  
    OTF2\_Archive.h, 109  
OTF2\_Archive\_GetFileSubstrate  
    OTF2\_Archive.h, 110  
OTF2\_Archive\_GetGlobalDefReader  
    OTF2\_Archive.h, 110  
OTF2\_Archive\_GetGlobalDefWriter  
    OTF2\_Archive.h, 110  
OTF2\_Archive\_GetGlobalEvtReader  
    OTF2\_Archive.h, 111  
OTF2\_Archive\_GetGlobalSnapReader  
    OTF2\_Archive.h, 111  
OTF2\_Archive\_GetMachineName  
    OTF2\_Archive.h, 111  
OTF2\_Archive\_GetMarkerReader  
    OTF2\_Archive.h, 112  
OTF2\_Archive\_GetMarkerWriter  
    OTF2\_Archive.h, 112  
OTF2\_Archive\_GetMasterSlaveMode  
    OTF2\_Archive.h, 113  
OTF2\_Archive\_GetNumberOfGlobalDefs  
    OTF2\_Archive.h, 113  
OTF2\_Archive\_GetNumberOfLocations  
    OTF2\_Archive.h, 113  
OTF2\_Archive\_GetNumberOfSnapshots  
    OTF2\_Archive.h, 114  
OTF2\_Archive\_GetNumberOfThumbnails  
    OTF2\_Archive.h, 114  
OTF2\_ArchiveGetProperty  
    OTF2\_Archive.h, 114  
OTF2\_Archive\_GetPropertyNames  
    OTF2\_Archive.h, 115  
OTF2\_Archive\_GetSnapReader  
    OTF2\_Archive.h, 115  
OTF2\_Archive\_GetSnapWriter  
    OTF2\_Archive.h, 116  
OTF2\_Archive\_GetThumbReader  
    OTF2\_Archive.h, 116  
OTF2\_Archive\_GetThumbWriter  
    OTF2\_Archive.h, 116  
OTF2\_Archive\_GetTraceId  
    OTF2\_Archive.h, 117  
OTF2\_Archive\_GetVersion  
    OTF2\_Archive.h, 117

OTF2\_Archive\_Open  
    OTF2\_Archive.h, 118  
OTF2\_Archive\_SetBoolProperty  
    OTF2\_Archive.h, 119  
OTF2\_Archive\_SetCreator  
    OTF2\_Archive.h, 120  
OTF2\_Archive\_SetDescription  
    OTF2\_Archive.h, 120  
OTF2\_Archive\_SetFileSionCallbacks  
    OTF2\_Archive.h, 121  
OTF2\_Archive\_SetFlushCallbacks  
    OTF2\_Archive.h, 121  
OTF2\_Archive\_SetMachineName  
    OTF2\_Archive.h, 121  
OTF2\_Archive\_SetMasterSlaveMode  
    OTF2\_Archive.h, 122  
OTF2\_Archive\_SetMemoryCallbacks  
    OTF2\_Archive.h, 122  
OTF2\_Archive\_SetNumberOfSnapshots  
    OTF2\_Archive.h, 123  
OTF2\_Archive\_SetProperty  
    OTF2\_Archive.h, 123  
OTF2\_Archive\_Switch FileMode  
    OTF2\_Archive.h, 124  
OTF2\_AttributeList  
    OTF2\_AttributeList.h, 124  
        OTF2\_AttributeList\_AddAttribute, 130  
        OTF2\_AttributeList\_AddAttributeRef,  
            131  
        OTF2\_AttributeList\_AddCommRef,  
            131  
        OTF2\_AttributeList\_AddDouble, 131  
        OTF2\_AttributeList\_AddFloat, 132  
        OTF2\_AttributeList\_AddGroupRef,  
            132  
        OTF2\_AttributeList\_AddInt16, 133  
        OTF2\_AttributeList\_AddInt32, 133  
        OTF2\_AttributeList\_AddInt64, 133  
        OTF2\_AttributeList\_AddInt8, 134  
        OTF2\_AttributeList\_AddLocationRef,  
            134  
        OTF2\_AttributeList\_AddMetricRef,  
            134  
        OTF2\_AttributeList\_AddParameterRef,  
            135

---

## INDEX

OTF2\_AttributeList\_AddRegionRef,  
    135  
OTF2\_AttributeList\_AddRmaWinRef,  
    136  
OTF2\_AttributeList\_AddString, 136  
OTF2\_AttributeList\_AddStringRef,  
    137  
OTF2\_AttributeList\_AddUint16, 137  
OTF2\_AttributeList\_AddUint32, 137  
OTF2\_AttributeList\_AddUint64, 138  
OTF2\_AttributeList\_AddUint8, 138     OTF2\_AttributeList\_AddAttribute  
OTF2\_AttributeList\_Delete, 138         OTF2\_AttributeList.h, 130  
OTF2\_AttributeList\_GetAttributeByID, 139  
OTF2\_AttributeList\_GetAttributeByIOTF2\_AttributeList\_AddCommRef  
    139         OTF2\_AttributeList.h, 131  
OTF2\_AttributeList\_GetAttributeRef, 140  
OTF2\_AttributeList\_GetAttributeRef, OTF2\_AttributeList\_AddDouble  
    140             OTF2\_AttributeList.h, 131  
OTF2\_AttributeList\_GetCommRef, 140  
OTF2\_AttributeList\_GetDouble, 140     OTF2\_AttributeList\_AddFloat  
OTF2\_AttributeList\_GetFloat, 141         OTF2\_AttributeList.h, 132  
OTF2\_AttributeList\_GetGroupRef, 141  
OTF2\_AttributeList\_GetInt16, 142     OTF2\_AttributeList\_AddGroupRef  
OTF2\_AttributeList\_GetInt32, 142  
OTF2\_AttributeList\_GetInt64, 142  
OTF2\_AttributeList\_GetInt8, 143  
OTF2\_AttributeList\_GetLocationRef, 143  
OTF2\_AttributeList\_GetMetricRef, 144  
OTF2\_AttributeList\_GetNumberOfElOTF2\_AttributeList\_AddLocationRef  
    144         OTF2\_AttributeList.h, 134  
OTF2\_AttributeList\_GetParameterReOTF2\_AttributeList\_AddMetricRef  
    144         OTF2\_AttributeList.h, 134  
OTF2\_AttributeList\_GetRegionRef, 145  
OTF2\_AttributeList\_GetRmaWinRef, 145  
OTF2\_AttributeList\_GetString, 146     OTF2\_AttributeList\_AddString  
OTF2\_AttributeList\_GetStringRef, 146     OTF2\_AttributeList.h, 136  
OTF2\_AttributeList\_GetUint16, 147  
OTF2\_AttributeList\_GetUint32, 147     OTF2\_AttributeList.h, 137

## INDEX

---

OTF2\_AttributeList\_AddUInt16  
    OTF2\_AttributeList.h, 137  
OTF2\_AttributeList\_AddUInt32  
    OTF2\_AttributeList.h, 137  
OTF2\_AttributeList\_AddUInt64  
    OTF2\_AttributeList.h, 138  
OTF2\_AttributeList\_AddUInt8  
    OTF2\_AttributeList.h, 138  
OTF2\_AttributeList\_Delete  
    OTF2\_AttributeList.h, 138  
OTF2\_AttributeList\_GetAttributeByID  
    OTF2\_AttributeList.h, 139  
OTF2\_AttributeList\_GetAttributeByIndex  
    OTF2\_AttributeList.h, 139  
OTF2\_AttributeList\_GetAttributeRef  
    OTF2\_AttributeList.h, 140  
OTF2\_AttributeList\_GetCommRef  
    OTF2\_AttributeList.h, 140  
OTF2\_AttributeList\_GetDouble  
    OTF2\_AttributeList.h, 140  
OTF2\_AttributeList\_GetFloat  
    OTF2\_AttributeList.h, 141  
OTF2\_AttributeList\_GetGroupRef  
    OTF2\_AttributeList.h, 141  
OTF2\_AttributeList\_GetInt16  
    OTF2\_AttributeList.h, 142  
OTF2\_AttributeList\_GetInt32  
    OTF2\_AttributeList.h, 142  
OTF2\_AttributeList\_GetInt64  
    OTF2\_AttributeList.h, 142  
OTF2\_AttributeList\_GetInt8  
    OTF2\_AttributeList.h, 143  
OTF2\_AttributeList\_GetLocationRef  
    OTF2\_AttributeList.h, 143  
OTF2\_AttributeList\_GetMetricRef  
    OTF2\_AttributeList.h, 144  
OTF2\_AttributeList\_GetNumberOfElements  
    OTF2\_AttributeList.h, 144  
OTF2\_AttributeList\_GetParameterRef  
    OTF2\_AttributeList.h, 144  
OTF2\_AttributeList\_GetRegionRef  
    OTF2\_AttributeList.h, 145  
OTF2\_AttributeList\_GetRmaWinRef  
    OTF2\_AttributeList.h, 145

OTF2\_AttributeList\_GetString  
    OTF2\_AttributeList.h, 146  
OTF2\_AttributeList\_GetStringRef  
    OTF2\_AttributeList.h, 146  
OTF2\_AttributeList\_GetUInt16  
    OTF2\_AttributeList.h, 147  
OTF2\_AttributeList\_GetUInt32  
    OTF2\_AttributeList.h, 147  
OTF2\_AttributeList\_GetUInt64  
    OTF2\_AttributeList.h, 147  
OTF2\_AttributeList\_GetUInt8  
    OTF2\_AttributeList.h, 148  
OTF2\_AttributeList\_New  
    OTF2\_AttributeList.h, 148  
OTF2\_AttributeList\_PopAttribute  
    OTF2\_AttributeList.h, 148  
OTF2\_AttributeList\_RemoveAllAttributes  
    OTF2\_AttributeList.h, 149  
OTF2\_AttributeList\_RemoveAttribute  
    OTF2\_AttributeList.h, 149  
OTF2\_AttributeList\_TestAttributeByID  
    OTF2\_AttributeList.h, 150  
OTF2\_AttributeValue, 89  
OTF2\_CallbackCode  
    OTF2\_GeneralDefinitions.h, 363  
OTF2\_Callbacks.h, 150  
    OTF2\_FileSionClose, 152  
    OTF2\_FileSionGetRank, 152  
    OTF2\_FileSionOpen, 152  
    OTF2\_MemoryAllocate, 153  
    OTF2\_MemoryFreeAll, 154  
    OTF2\_PostFlushCallback, 154  
    OTF2\_PreFlushCallback, 155  
OTF2\_CHUNK\_SIZE\_DEFINITIONS\_-  
    DEFAULT  
        OTF2\_Archive.h, 101  
OTF2\_CHUNK\_SIZE\_EVENTS\_DEFAULT  
    OTF2\_Archive.h, 101  
OTF2\_CollectiveOp\_enum  
    OTF2\_Events.h, 228  
OTF2\_Compression\_enum  
    OTF2\_GeneralDefinitions.h, 363  
OTF2\_Definitions.h, 155  
OTF2\_GroupFlag\_enum, 161

## INDEX

---

OTF2\_GroupType\_enum, 161  
OTF2\_LocationGroupType\_enum, 160  
OTF2\_LocationType\_enum, 162  
OTF2\_MetricBase\_enum, 163  
OTF2\_MetricMode\_enum, 163  
OTF2\_MetricOccurrence\_enum, 163  
OTF2\_MetricScope\_enum, 164  
OTF2\_MetricTiming\_enum, 164  
OTF2\_MetricType\_enum, 165  
OTF2\_MetricValueProperty\_enum, 165  
OTF2\_ParameterType\_enum, 166  
OTF2RecorderKind\_enum, 166  
OTF2RegionFlag\_enum, 166  
OTF2RegionRole\_enum, 167  
OTF2SystemTreeDomain\_enum, 168  
OTF2\_DefReader.h, 169  
    OTF2\_DefReader\_GetLocationID, 170  
    OTF2\_DefReader\_ReadDefinitions, 170  
        OTF2\_DefReader\_SetCallbacks, 171  
OTF2\_DefReader\_GetLocationID  
    OTF2\_DefReader.h, 170  
OTF2\_DefReader\_ReadDefinitions  
    OTF2\_DefReader.h, 170  
OTF2\_DefReader\_SetCallbacks  
    OTF2\_DefReader.h, 171  
OTF2\_DefReaderCallback\_Attribute  
    OTF2\_DefReaderCallbacks.h, 178  
OTF2\_DefReaderCallback\_Callpath  
    OTF2\_DefReaderCallbacks.h, 178  
OTF2\_DefReaderCallback\_Callsite  
    OTF2\_DefReaderCallbacks.h, 179  
OTF2\_DefReaderCallback\_ClockOffset  
    OTF2\_DefReaderCallbacks.h, 179  
OTF2\_DefReaderCallback\_Comm  
    OTF2\_DefReaderCallbacks.h, 180  
OTF2\_DefReaderCallback\_Group  
    OTF2\_DefReaderCallbacks.h, 180  
OTF2\_DefReaderCallback\_Location  
    OTF2\_DefReaderCallbacks.h, 181  
OTF2\_DefReaderCallback\_LocationGroup  
    OTF2\_DefReaderCallbacks.h, 182  
OTF2\_DefReaderCallback\_MappingTable

OTF2\_DefReaderCallbacks.h, 182  
OTF2\_DefReaderCallback\_MetricClass  
    OTF2\_DefReaderCallbacks.h, 183  
OTF2\_DefReaderCallback\_MetricClassRecorder  
    OTF2\_DefReaderCallbacks.h, 184  
OTF2\_DefReaderCallback\_MetricInstance  
    OTF2\_DefReaderCallbacks.h, 184  
OTF2\_DefReaderCallback\_MetricMember  
    OTF2\_DefReaderCallbacks.h, 185  
OTF2\_DefReaderCallback\_Parameter  
    OTF2\_DefReaderCallbacks.h, 186  
OTF2\_DefReaderCallback\_Region  
    OTF2\_DefReaderCallbacks.h, 187  
OTF2\_DefReaderCallback\_RmaWin  
    OTF2\_DefReaderCallbacks.h, 188  
OTF2\_DefReaderCallback\_String  
    OTF2\_DefReaderCallbacks.h, 188  
OTF2\_DefReaderCallback\_SystemTreeNode  
    OTF2\_DefReaderCallbacks.h, 189  
OTF2\_DefReaderCallback\_SystemTreeNodeDomain  
    OTF2\_DefReaderCallbacks.h, 189  
OTF2\_DefReaderCallback\_SystemTreeNodeProperty  
    OTF2\_DefReaderCallbacks.h, 190  
OTF2\_DefReaderCallback\_Unknown  
    OTF2\_DefReaderCallbacks.h, 191  
OTF2\_DefReaderCallbacks.h, 172  
    OTF2\_DefReaderCallback\_Attribute, 178  
OTF2\_DefReaderCallback\_Callpath, 178  
OTF2\_DefReaderCallback\_Callsite, 179  
OTF2\_DefReaderCallback\_ClockOffset, 179  
OTF2\_DefReaderCallback\_Comm, 180  
OTF2\_DefReaderCallback\_Group, 180  
OTF2\_DefReaderCallback\_Location, 181  
OTF2\_DefReaderCallback\_LocationGroup, 182  
OTF2\_DefReaderCallback\_MappingTable, 182

## INDEX

---

OTF2\_DefReaderCallback\_MetricClass, OTF2\_DefReaderCallbacks\_SetMappingTableCallback,  
183 195  
OTF2\_DefReaderCallback\_MetricClassRecorder, OTF2\_DefReaderCallbacks\_SetMetricClassCallback,  
184 196  
OTF2\_DefReaderCallback\_MetricInstance, OTF2\_DefReaderCallbacks\_SetMetricClassRecorderCallback,  
184 196  
OTF2\_DefReaderCallback\_MetricMember, OTF2\_DefReaderCallbacks\_SetMetricInstanceCallback,  
185 197  
OTF2\_DefReaderCallback\_Parameter, OTF2\_DefReaderCallbacks\_SetMetricMemberCallback,  
186 197  
OTF2\_DefReaderCallback\_Region, OTF2\_DefReaderCallbacks\_SetParameterCallback,  
187 198  
OTF2\_DefReaderCallback\_RmaWin, OTF2\_DefReaderCallbacks\_SetRegionCallback,  
188 198  
OTF2\_DefReaderCallback\_String, 188 OTF2\_DefReaderCallbacks\_SetRmaWinCallback,  
OTF2\_DefReaderCallback\_SystemTreeNode, 189 199  
189 OTF2\_DefReaderCallbacks\_SetStringCallback,  
199  
OTF2\_DefReaderCallback\_SystemTreeNodeDomain, 189 OTF2\_DefReaderCallbacks\_SetSystemTreeNodeCallback,  
199  
OTF2\_DefReaderCallback\_SystemTreeNodeProperty, 190 200  
OTF2\_DefReaderCallbacks\_SetSystemTreeNodeDomainCallback  
190 200  
OTF2\_DefReaderCallback\_Unknown, 191 OTF2\_DefReaderCallbacks\_SetSystemTreeNodePropertyCallback  
191 201  
OTF2\_DefReaderCallbacks\_Clear, 191 OTF2\_DefReaderCallbacks\_SetUnknownCallback,  
OTF2\_DefReaderCallbacks\_Delete, 191 201  
OTF2\_DefReaderCallbacks\_Clear  
OTF2\_DefReaderCallbacks\_New, 191 OTF2\_DefReaderCallbacks.h, 191  
OTF2\_DefReaderCallbacks\_SetAttributeCallback  
192 OTF2\_DefReaderCallbacks\_Delete  
192 OTF2\_DefReaderCallbacks.h, 191  
OTF2\_DefReaderCallbacks\_SetCallpathCallback  
192 OTF2\_DefReaderCallbacks\_New  
192 OTF2\_DefReaderCallbacks.h, 191  
OTF2\_DefReaderCallbacks\_SetCallsiteCallback  
193 OTF2\_DefReaderCallbacks\_SetAttributeCallback  
193 OTF2\_DefReaderCallbacks.h, 192  
OTF2\_DefReaderCallbacks\_SetClockOffsetCallback  
193 OTF2\_DefReaderCallbacks\_SetCallpathCallback  
193 OTF2\_DefReaderCallbacks.h, 192  
OTF2\_DefReaderCallbacks\_SetCommCallback  
193 OTF2\_DefReaderCallbacks\_SetCallsiteCallback  
193 OTF2\_DefReaderCallbacks.h, 193  
OTF2\_DefReaderCallbacks\_SetGroupCallback  
194 OTF2\_DefReaderCallbacks\_SetClockOffsetCallback  
194 OTF2\_DefReaderCallbacks.h, 193  
OTF2\_DefReaderCallbacks\_SetLocationCallback  
194 OTF2\_DefReaderCallbacks\_SetCommCallback  
194 OTF2\_DefReaderCallbacks.h, 193  
OTF2\_DefReaderCallbacks\_SetLocationCallback  
195 OTF2\_DefReaderCallbacks\_SetGroupCallback  
195 OTF2\_DefReaderCallbacks.h, 194

## INDEX

---

OTF2\_DefReaderCallbacks\_SetLocationCallbackOTF2\_DefWriter\_WriteMetricClass,  
OTF2\_DefReaderCallbacks.h, 194 210  
OTF2\_DefReaderCallbacks\_SetLocationGroupOTF2\_DefWriter\_WriteMetricClassRecorder,  
OTF2\_DefReaderCallbacks.h, 195 210  
OTF2\_DefReaderCallbacks\_SetMappingTableOTF2\_DefWriter\_WriteMetricInstance,  
OTF2\_DefReaderCallbacks.h, 195 211  
OTF2\_DefReaderCallbacks\_SetMetricClassCallbackOTF2\_DefWriter\_WriteMetricMember,  
OTF2\_DefReaderCallbacks.h, 196 211  
OTF2\_DefReaderCallbacks\_SetMetricClassRecorderOTF2\_DefWriter\_WriteParameter, 212  
OTF2\_DefReaderCallbacks.h, 196 OTF2\_DefWriter\_WriteRegion, 213  
OTF2\_DefReaderCallbacks\_SetMetricInstanceOTF2\_DefWriter\_WriteRmaWin, 214  
OTF2\_DefReaderCallbacks.h, 197 OTF2\_DefWriter\_WriteString, 214  
OTF2\_DefReaderCallbacks\_SetMetricMemberCallbackOTF2\_DefWriter\_WriteSystemTreeNode,  
OTF2\_DefReaderCallbacks.h, 197 215  
OTF2\_DefReaderCallbacks\_SetParameterCallbackOTF2\_DefWriter\_WriteSystemTreeNodeDomain,  
OTF2\_DefReaderCallbacks.h, 198 215  
OTF2\_DefReaderCallbacks\_SetRegionCallbackOTF2\_DefWriter\_WriteSystemTreeNodeProperty,  
OTF2\_DefReaderCallbacks.h, 198 216  
OTF2\_DefReaderCallbacks\_SetRmaWinCallbackOTF2\_DefWriter\_GetLocationID  
OTF2\_DefReaderCallbacks.h, 199 OTF2\_DefWriter.h, 204  
OTF2\_DefReaderCallbacks\_SetStringCallbackOTF2\_DefWriter\_WriteAttribute  
OTF2\_DefReaderCallbacks.h, 199 OTF2\_DefWriter.h, 205  
OTF2\_DefReaderCallbacks\_SetSystemTreeOTF2\_DefWriter\_WriteCallpath  
OTF2\_DefReaderCallbacks.h, 200 OTF2\_DefWriter.h, 205  
OTF2\_DefReaderCallbacks\_SetSystemTreeOTF2\_DefWriter\_CallSite  
OTF2\_DefReaderCallbacks.h, 200 OTF2\_DefWriter.h, 206  
OTF2\_DefReaderCallbacks\_SetSystemTreeOTF2\_DefWriter\_ClockOffset  
OTF2\_DefReaderCallbacks.h, 201 OTF2\_DefWriter.h, 206  
OTF2\_DefReaderCallbacks\_SetUnknownOTF2\_DefWriter\_WriteComm  
OTF2\_DefReaderCallbacks.h, 201 OTF2\_DefWriter.h, 206  
OTF2\_DefWriter.h, 202 OTF2\_DefWriter\_WriteGroup  
OTF2\_DefWriter\_GetLocationID, 204 OTF2\_DefWriter.h, 207  
OTF2\_DefWriter\_WriteAttribute, 205 OTF2\_DefWriter\_WriteLocation  
OTF2\_DefWriter\_WriteCallpath, 205 OTF2\_DefWriter.h, 208  
OTF2\_DefWriter\_WriteCallsite, 206 OTF2\_DefWriter\_WriteLocationGroup  
OTF2\_DefWriter\_WriteClockOffset, 206 OTF2\_DefWriter.h, 208  
OTF2\_DefWriter\_WriteMappingTable  
OTF2\_DefWriter\_WriteComm, 206 OTF2\_DefWriter.h, 209  
OTF2\_DefWriter\_WriteGroup, 207 OTF2\_DefWriter\_WriteMetricClass  
OTF2\_DefWriter\_WriteLocation, 208 OTF2\_DefWriter.h, 210  
OTF2\_DefWriter\_WriteLocationGroupOTF2\_DefWriter\_WriteMetricClassRecorder  
208 OTF2\_DefWriter.h, 210  
OTF2\_DefWriter\_WriteMappingTableOTF2\_DefWriter\_WriteMetricInstance  
209 OTF2\_DefWriter.h, 211

## INDEX

---

OTF2\_DefWriter\_WriteMetricMember  
    OTF2\_DefWriter.h, 211

OTF2\_DefWriter\_WriteParameter  
    OTF2\_DefWriter.h, 212

OTF2\_DefWriter\_WriteRegion  
    OTF2\_DefWriter.h, 213

OTF2\_DefWriter\_WriteRmaWin  
    OTF2\_DefWriter.h, 214

OTF2\_DefWriter\_WriteString  
    OTF2\_DefWriter.h, 214

OTF2\_DefWriter\_WriteSystemTreeNode  
    OTF2\_DefWriter.h, 215

OTF2\_DefWriter\_WriteSystemTreeNodeDom  
    OTF2\_DefWriter.h, 215

OTF2\_DefWriter\_WriteSystemTreeNodeProperty  
    OTF2\_DefWriter.h, 216

OTF2\_Error\_GetDescription  
    OTF2\_ErrorCodes.h, 225

OTF2\_Error\_GetName  
    OTF2\_ErrorCodes.h, 225

OTF2\_Error\_RegisterCallback  
    OTF2\_ErrorCodes.h, 225

OTF2\_ErrorCallback  
    OTF2\_ErrorCodes.h, 221

OTF2\_ErrorCode  
    OTF2\_ErrorCodes.h, 221

OTF2\_ErrorCodes.h, 217

OTF2\_Error\_GetDescription, 225

OTF2\_Error\_GetName, 225

OTF2\_Error\_RegisterCallback, 225

OTF2\_ErrorCallback, 221

OTF2\_ErrorCode, 221

OTF2\_Events.h, 226

    OTF2\_CollectiveOp\_enum, 228

    OTF2\_LockType\_enum, 228

    OTF2\_MeasurementMode\_enum, 228

    OTF2\_RmaAtomicType\_enum, 229

    OTF2\_RmaSyncLevel\_enum, 229

    OTF2\_RmaSyncType\_enum, 229

OTF2\_EvtReader.h, 230

    OTF2\_EvtReader\_ApplyClockOffsets, 232

    OTF2\_EvtReader\_ApplyMappingTables, 232

OTF2\_EvtReader.h, 232

    OTF2\_EvtReader\_GetLocationID, 232

    OTF2\_EvtReader\_GetPos, 233

    OTF2\_EvtReader\_ReadEvents, 233

    OTF2\_EvtReader\_ReadEventsBackward, 233

    OTF2\_EvtReader\_Seek, 234

    OTF2\_EvtReader\_SetCallbacks, 234

    OTF2\_EvtReader\_TimeStampRewrite, 235

    OTF2\_EvtReader\_ApplyClockOffsets

    OTF2\_EvtReader.h, 232

    OTF2\_EvtReader\_ApplyMappingTables

OTF2\_EvtReader.h, 232

    OTF2\_EvtReader\_GetLocationID

OTF2\_EvtReader.h, 232

    OTF2\_EvtReader\_GetPos

    OTF2\_EvtReader.h, 233

    OTF2\_EvtReader\_ReadEvents

    OTF2\_EvtReader.h, 233

    OTF2\_EvtReader\_ReadEventsBackward

    OTF2\_EvtReader.h, 233

    OTF2\_EvtReader\_Seek

    OTF2\_EvtReader.h, 234

    OTF2\_EvtReader\_SetCallbacks

    OTF2\_EvtReader.h, 234

    OTF2\_EvtReader\_TimeStampRewrite

    OTF2\_EvtReader.h, 235

    OTF2\_EvtReaderCallback\_BufferFlush

    OTF2\_EvtReaderCallbacks.h, 248

    OTF2\_EvtReaderCallback\_Enter

    OTF2\_EvtReaderCallbacks.h, 248

    OTF2\_EvtReaderCallback\_Leave

    OTF2\_EvtReaderCallbacks.h, 249

    OTF2\_EvtReaderCallback\_MeasurementOnOff

    OTF2\_EvtReaderCallbacks.h, 250

    OTF2\_EvtReaderCallback\_Metric

    OTF2\_EvtReaderCallbacks.h, 250

    OTF2\_EvtReaderCallback\_MpiCollectiveBegin

    OTF2\_EvtReaderCallbacks.h, 251

    OTF2\_EvtReaderCallback\_MpiCollectiveEnd

    OTF2\_EvtReaderCallbacks.h, 252

    OTF2\_EvtReaderCallback\_MpiIrecv

    OTF2\_EvtReaderCallbacks.h, 253

    OTF2\_EvtReaderCallback\_MpiIrecvRequest

---

## INDEX

OTF2\_EvtReaderCallbacks.h, 254  
OTF2\_EvtReaderCallback\_MpiIsend  
    OTF2\_EvtReaderCallbacks.h, 254  
OTF2\_EvtReaderCallback\_MpiIsendComplete  
    OTF2\_EvtReaderCallbacks.h, 255  
OTF2\_EvtReaderCallback\_MpiRecv  
    OTF2\_EvtReaderCallbacks.h, 256  
OTF2\_EvtReaderCallback\_MpiRequestCancel  
    OTF2\_EvtReaderCallbacks.h, 257  
OTF2\_EvtReaderCallback\_MpiRequestTest  
    OTF2\_EvtReaderCallbacks.h, 258  
OTF2\_EvtReaderCallback\_MpiSend  
    OTF2\_EvtReaderCallbacks.h, 259  
OTF2\_EvtReaderCallback\_OmpAcquireLock  
    OTF2\_EvtReaderCallbacks.h, 259  
OTF2\_EvtReaderCallback\_OmpFork  
    OTF2\_EvtReaderCallbacks.h, 259  
OTF2\_EvtReaderCallback\_OmpJoin  
    OTF2\_EvtReaderCallbacks.h, 260  
OTF2\_EvtReaderCallback\_OmpReleaseLock  
    OTF2\_EvtReaderCallbacks.h, 261  
OTF2\_EvtReaderCallback\_OmpTaskCompletion  
    OTF2\_EvtReaderCallbacks.h, 262  
OTF2\_EvtReaderCallback\_OmpTaskCreate  
    OTF2\_EvtReaderCallbacks.h, 262  
OTF2\_EvtReaderCallback\_OmpTaskSwitch  
    OTF2\_EvtReaderCallbacks.h, 263  
OTF2\_EvtReaderCallback\_ParameterInt  
    OTF2\_EvtReaderCallbacks.h, 264  
OTF2\_EvtReaderCallback\_ParameterString  
    OTF2\_EvtReaderCallbacks.h, 264  
OTF2\_EvtReaderCallback\_ParameterUnion  
    OTF2\_EvtReaderCallbacks.h, 265  
OTF2\_EvtReaderCallback\_RmaAcquireLock  
    OTF2\_EvtReaderCallbacks.h, 266  
OTF2\_EvtReaderCallback\_RmaAtomic  
    OTF2\_EvtReaderCallbacks.h, 267  
OTF2\_EvtReaderCallback\_RmaCollective  
    OTF2\_EvtReaderCallbacks.h, 268  
OTF2\_EvtReaderCallback\_RmaCollectiveSync  
    OTF2\_EvtReaderCallbacks.h, 268  
OTF2\_EvtReaderCallback\_RmaGet  
    OTF2\_EvtReaderCallbacks.h, 269  
OTF2\_EvtReaderCallback\_RmaGroupSync  
    OTF2\_EvtReaderCallbacks.h, 270  
OTF2\_EvtReaderCallback\_RmaOpCompleteBlocking  
    OTF2\_EvtReaderCallbacks.h, 271  
OTF2\_EvtReaderCallback\_RmaOpCompleteNonBlocking  
    OTF2\_EvtReaderCallbacks.h, 271  
OTF2\_EvtReaderCallback\_RmaOpCompleteRemote  
    OTF2\_EvtReaderCallbacks.h, 272  
OTF2\_EvtReaderCallback\_RmaOpTest  
    OTF2\_EvtReaderCallbacks.h, 273  
OTF2\_EvtReaderCallback\_RmaPut  
    OTF2\_EvtReaderCallbacks.h, 274  
OTF2\_EvtReaderCallback\_RmaReleaseLock  
    OTF2\_EvtReaderCallbacks.h, 274  
OTF2\_EvtReaderCallback\_RmaRequestLock  
    OTF2\_EvtReaderCallbacks.h, 275  
OTF2\_EvtReaderCallback\_RmaSync  
    OTF2\_EvtReaderCallbacks.h, 276  
OTF2\_EvtReaderCallback\_RmaTryLock  
    OTF2\_EvtReaderCallbacks.h, 277  
OTF2\_EvtReaderCallback\_RmaWaitChange  
    OTF2\_EvtReaderCallbacks.h, 278  
OTF2\_EvtReaderCallback\_RmaWinCreate  
    OTF2\_EvtReaderCallbacks.h, 278  
OTF2\_EvtReaderCallback\_RmaWinDestroy  
    OTF2\_EvtReaderCallbacks.h, 279  
OTF2\_EvtReaderCallback\_ThreadAcquireLock  
    OTF2\_EvtReaderCallbacks.h, 280  
OTF2\_EvtReaderCallback\_ThreadFork  
    OTF2\_EvtReaderCallbacks.h, 280  
OTF2\_EvtReaderCallback\_ThreadJoin  
    OTF2\_EvtReaderCallbacks.h, 281  
OTF2\_EvtReaderCallback\_ThreadReleaseLock  
    OTF2\_EvtReaderCallbacks.h, 282  
OTF2\_EvtReaderCallback\_ThreadTaskComplete  
    OTF2\_EvtReaderCallbacks.h, 282  
OTF2\_EvtReaderCallback\_ThreadTaskCreate  
    OTF2\_EvtReaderCallbacks.h, 283  
OTF2\_EvtReaderCallback\_ThreadTaskSwitch  
    OTF2\_EvtReaderCallbacks.h, 284  
OTF2\_EvtReaderCallback\_ThreadTeamBegin  
    OTF2\_EvtReaderCallbacks.h, 285  
OTF2\_EvtReaderCallback\_ThreadTeamEnd  
    OTF2\_EvtReaderCallbacks.h, 285  
OTF2\_EvtReaderCallback\_Unknown  
    OTF2\_EvtReaderCallbacks.h, 285

## INDEX

---

OTF2_EvtReaderCallbacks.h, 286	OTF2_EvtReaderCallback_ParameterInt, 264
OTF2_EvtReaderCallbacks.h, 235	OTF2_EvtReaderCallback_ParameterString, 264
OTF2_EvtReaderCallback_BufferFlush, 248	OTF2_EvtReaderCallback_ParameterUnsignedInt, 265
OTF2_EvtReaderCallback_Enter, 248	OTF2_EvtReaderCallback_RmaAcquireLock, 266
OTF2_EvtReaderCallback_Leave, 249	OTF2_EvtReaderCallback_RmaAtomic, 267
OTF2_EvtReaderCallback_MeasurementOnOff, 250	OTF2_EvtReaderCallback_RmaCollectiveBegin, 268
OTF2_EvtReaderCallback_Metric, 250	OTF2_EvtReaderCallback_RmaCollectiveEnd, 268
OTF2_EvtReaderCallback_MpiCollectiveBegin, 251	OTF2_EvtReaderCallback_RmaGet, 269
OTF2_EvtReaderCallback_MpiCollectiveEnd, 252	OTF2_EvtReaderCallback_RmaGroupSync, 270
OTF2_EvtReaderCallback_MpiIrecv, 253	OTF2_EvtReaderCallback_RmaOpCompleteBlocking, 271
OTF2_EvtReaderCallback_MpiIrecvRequest, 254	OTF2_EvtReaderCallback_RmaOpCompleteNonBlocking, 271
OTF2_EvtReaderCallback_MpiIsend, 254	OTF2_EvtReaderCallback_RmaOpCompleteRemote, 272
OTF2_EvtReaderCallback_MpiIsendComplete, 255	OTF2_EvtReaderCallback_RmaOpTest, 273
OTF2_EvtReaderCallback_MpiRecv, 256	OTF2_EvtReaderCallback_RmaPut, 274
OTF2_EvtReaderCallback_MpiRequestCancelled, 257	OTF2_EvtReaderCallback_RmaReleaseLock, 274
OTF2_EvtReaderCallback_MpiRequestTest, 257	OTF2_EvtReaderCallback_RmaRequestLock, 275
OTF2_EvtReaderCallback_MpiSend, 258	OTF2_EvtReaderCallback_RmaSync, 276
OTF2_EvtReaderCallback_OmpAcquireLock, 259	OTF2_EvtReaderCallback_RmaTryLock, 277
OTF2_EvtReaderCallback_OmpFork, 259	OTF2_EvtReaderCallback_RmaWaitChange, 278
OTF2_EvtReaderCallback_OmpJoin, 260	OTF2_EvtReaderCallback_RmaWinCreate, 278
OTF2_EvtReaderCallback_OmpReleaseLock, 261	OTF2_EvtReaderCallback_RmaWinDestroy, 279
OTF2_EvtReaderCallback_OmpTaskComplete, 262	OTF2_EvtReaderCallback_ThreadAcquireLock, 280

---

## INDEX

OTF2_EvtReaderCallback_ThreadFork,	OTF2_EvtReaderCallbacks_SetMpiRecvCallback,
280	292
OTF2_EvtReaderCallback_ThreadJoin,	OTF2_EvtReaderCallbacks_SetMpiRequestCancelledCallback,
281	293
OTF2_EvtReaderCallback_ThreadRelease	OTF2_EvtReaderCallbacks_SetMpiRequestTestCallback,
282	293
OTF2_EvtReaderCallback_ThreadTaskComplete	OTF2_EvtReaderCallbacks_SetMpiSendCallback,
282	294
OTF2_EvtReaderCallback_ThreadTaskCreate	OTF2_EvtReaderCallbacks_SetOmpAcquireLockCallback,
283	294
OTF2_EvtReaderCallback_ThreadTaskSwitch	OTF2_EvtReaderCallbacks_SetOmpForkCallback,
284	295
OTF2_EvtReaderCallback_ThreadTeamBegin	OTF2_EvtReaderCallbacks_SetOmpJoinCallback,
285	295
OTF2_EvtReaderCallback_ThreadTeamEnd	OTF2_EvtReaderCallbacks_SetOmpReleaseLockCallback,
285	296
OTF2_EvtReaderCallback_Unknown,	OTF2_EvtReaderCallbacks_SetOmpTaskCompleteCallback,
286	296
OTF2_EvtReaderCallbacks_Clear,	OTF2_EvtReaderCallbacks_SetOmpTaskCreateCallback,
287	297
OTF2_EvtReaderCallbacks_Delete,	OTF2_EvtReaderCallbacks_SetOmpTaskSwitchCallback,
287	297
OTF2_EvtReaderCallbacks_New,	OTF2_EvtReaderCallbacks_SetParameterIntCallback,
287	298
OTF2_EvtReaderCallbacks_SetBufferFlush	OTF2_EvtReaderCallbacks_SetParameterStringCallback,
287	298
OTF2_EvtReaderCallbacks_SetEnterCall	OTF2_EvtReaderCallbacks_SetRmaAcquireLockCallback,
288	298
OTF2_EvtReaderCallbacks_SetLeaveCall	OTF2_EvtReaderCallbacks_SetRmaAtomicCallback,
288	299
OTF2_EvtReaderCallbacks_SetMeasurement	OTF2_EvtReaderCallbacks_SetRmaCollectiveBeginCallback,
289	299
OTF2_EvtReaderCallbacks_SetMetricCall	OTF2_EvtReaderCallbacks_SetRmaCollectiveEndCallback,
289	300
OTF2_EvtReaderCallbacks_SetMpiCollective	OTF2_EvtReaderCallbacks_SetRmaGetCallback,
290	300
OTF2_EvtReaderCallbacks_SetMpiCollective	OTF2_EvtReaderCallbacks_SetRmaGroupSyncCallback,
290	301
OTF2_EvtReaderCallbacks_SetMpiIrecv	OTF2_EvtReaderCallbacks_SetRmaOpCompleteBlockingCallback,
291	301
OTF2_EvtReaderCallbacks_SetMpiIrecv	OTF2_EvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback,
291	302
OTF2_EvtReaderCallbacks_SetMpiIsend	OTF2_EvtReaderCallbacks_SetRmaOpCompleteBlockingCallback,
292	302
OTF2_EvtReaderCallbacks_SetMpiIsend	OTF2_EvtReaderCallbacks_SetRmaOpCompleteNonBlockingCallback,
292	303

## INDEX

---

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_New  
304 OTF2\_EvtReaderCallbacks.h, 287

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetBufferFlushCallback  
304 OTF2\_EvtReaderCallbacks.h, 287

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetEnterCallback  
305 OTF2\_EvtReaderCallbacks.h, 288

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetLeaveCallback  
305 OTF2\_EvtReaderCallbacks.h, 288

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetMeasurementOnOffCallback  
305 OTF2\_EvtReaderCallbacks.h, 289

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetMetricCallback  
306 OTF2\_EvtReaderCallbacks.h, 289

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetMpiCollectiveBeginCallback  
306 OTF2\_EvtReaderCallbacks.h, 290

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetMpiCollectiveEndCallback  
307 OTF2\_EvtReaderCallbacks.h, 290

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetMpiIrecvCallback  
307 OTF2\_EvtReaderCallbacks.h, 291

OTF2\_EvtReaderCallbacks\_SetRmaOTF2\_EvtReaderCallbacks\_SetMpiIrecvRequestCallback  
308 OTF2\_EvtReaderCallbacks.h, 291

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetMpiIsendCallback  
308 OTF2\_EvtReaderCallbacks.h, 292

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetMpiIsendCompleteCallback  
309 OTF2\_EvtReaderCallbacks.h, 292

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetMpiRecvCallback  
309 OTF2\_EvtReaderCallbacks.h, 292

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetMpiRequestCancelledCallback  
310 OTF2\_EvtReaderCallbacks.h, 293

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetMpiRequestTestCallback  
310 OTF2\_EvtReaderCallbacks.h, 293

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetMpiSendCallback  
311 OTF2\_EvtReaderCallbacks.h, 294

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetOmpAcquireLockCallback  
311 OTF2\_EvtReaderCallbacks.h, 294

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetOmpForkCallback  
312 OTF2\_EvtReaderCallbacks.h, 295

OTF2\_EvtReaderCallbacks\_SetThreaOTF2\_EvtReaderCallbacks\_SetOmpJoinCallback  
312 OTF2\_EvtReaderCallbacks.h, 295

OTF2\_EvtReaderCallbacks\_SetUnknOTF2\_EvtReaderCallbacks\_SetOmpReleaseLockCallback  
313 OTF2\_EvtReaderCallbacks.h, 296

OTF2\_EvtReaderCallbacks\_Clear OTF2\_EvtReaderCallbacks\_SetOmpTaskCompleteCallback  
OTF2\_EvtReaderCallbacks.h, 287 OTF2\_EvtReaderCallbacks.h, 296

OTF2\_EvtReaderCallbacks\_Delete OTF2\_EvtReaderCallbacks\_SetOmpTaskCreateCallback  
OTF2\_EvtReaderCallbacks.h, 287 OTF2\_EvtReaderCallbacks.h, 297

## INDEX

## INDEX

---

OTF2\_EvtWriter\_MpiRequestCancelled, OTF2\_EvtWriter\_RmaWaitChange,  
329 348  
OTF2\_EvtWriter\_MpiRequestTest, 330 OTF2\_EvtWriter\_RmaWinCreate, 348  
OTF2\_EvtWriter\_MpiSend, 330 OTF2\_EvtWriter\_RmaWinDestroy,  
349  
OTF2\_EvtWriter\_OmpAcquireLock,  
331  
OTF2\_EvtWriter\_OmpFork, 332  
OTF2\_EvtWriter\_OmpJoin, 332  
OTF2\_EvtWriter\_OmpReleaseLock,  
333  
OTF2\_EvtWriter\_OmpTaskComplete,  
334  
OTF2\_EvtWriter\_OmpTaskCreate, 334  
OTF2\_EvtWriter\_OmpTaskSwitch,  
335  
OTF2\_EvtWriter\_ParameterInt, 336  
OTF2\_EvtWriter\_ParameterString, 336  
OTF2\_EvtWriter\_ParameterUnsignedInt,  
337  
OTF2\_EvtWriter\_Rewind, 338  
OTF2\_EvtWriter\_RmaAcquireLock,  
338  
OTF2\_EvtWriter\_RmaAtomic, 339  
OTF2\_EvtWriter\_RmaCollectiveBegin,  
340  
OTF2\_EvtWriter\_RmaCollectiveEnd,  
340  
OTF2\_EvtWriter\_RmaGet, 341  
OTF2\_EvtWriter\_RmaGroupSync, 341  
OTF2\_EvtWriter\_RmaOpCompleteBlocking  
342  
OTF2\_EvtWriter\_RmaOpCompleteNonBlocking  
343  
OTF2\_EvtWriter\_RmaOpCompleteRemote  
343  
OTF2\_EvtWriter\_RmaOpTest, 344  
OTF2\_EvtWriter\_RmaPut, 345  
OTF2\_EvtWriter\_RmaReleaseLock,  
345  
OTF2\_EvtWriter\_RmaRequestLock,  
346  
OTF2\_EvtWriter\_RmaSync, 347  
OTF2\_EvtWriter\_RmaTryLock, 347  
OTF2\_EvtWriter\_SetLocationID, 350  
OTF2\_EvtWriter\_SetUserData, 350  
OTF2\_EvtWriter\_StoreRewindPoint,  
350  
OTF2\_EvtWriter\_ThreadAcquireLock,  
351  
OTF2\_EvtWriter\_ThreadFork, 351  
OTF2\_EvtWriter\_ThreadJoin, 352  
OTF2\_EvtWriter\_ThreadReleaseLock,  
352  
OTF2\_EvtWriter\_ThreadTaskComplete,  
353  
OTF2\_EvtWriter\_ThreadTaskCreate,  
354  
OTF2\_EvtWriter\_ThreadTaskSwitch,  
354  
OTF2\_EvtWriter\_ThreadTeamBegin,  
355  
OTF2\_EvtWriter\_ThreadTeamEnd,  
356  
OTF2\_EvtWriter\_BufferFlush  
OTF2\_EvtWriter.h, 320  
OTF2\_EvtWriter\_ClearRewindPoint  
OTF2\_EvtWriter.h, 321  
OTF2\_EvtWriter\_Enter  
OTF2\_EvtWriter.h, 321  
OTF2\_EvtWriter\_GetLocationID  
OTF2\_EvtWriter.h, 322  
OTF2\_EvtWriter\_GetNumberOfEvents  
OTF2\_EvtWriter.h, 322  
OTF2\_EvtWriter\_GetUserData  
OTF2\_EvtWriter.h, 323  
OTF2\_EvtWriter\_Leave  
OTF2\_EvtWriter.h, 323  
OTF2\_EvtWriter\_MeasurementOnOff  
OTF2\_EvtWriter.h, 323  
OTF2\_EvtWriter\_Metric  
OTF2\_EvtWriter.h, 324  
OTF2\_EvtWriter\_MpiCollectiveBegin

---

## INDEX

---

OTF2\_EvtWriter.h, 325  
OTF2\_EvtWriter\_MpiCollectiveEnd  
    OTF2\_EvtWriter.h, 325  
OTF2\_EvtWriter\_MpiIrecv  
    OTF2\_EvtWriter.h, 326  
OTF2\_EvtWriter\_MpiIrecvRequest  
    OTF2\_EvtWriter.h, 327  
OTF2\_EvtWriter\_MpiIsend  
    OTF2\_EvtWriter.h, 327  
OTF2\_EvtWriter\_MpiIsendComplete  
    OTF2\_EvtWriter.h, 328  
OTF2\_EvtWriter\_MpiRecv  
    OTF2\_EvtWriter.h, 328  
OTF2\_EvtWriter\_MpiRequestCancelled  
    OTF2\_EvtWriter.h, 329  
OTF2\_EvtWriter\_MpiRequestTest  
    OTF2\_EvtWriter.h, 330  
OTF2\_EvtWriter\_MpiSend  
    OTF2\_EvtWriter.h, 330  
OTF2\_EvtWriter\_OmpAcquireLock  
    OTF2\_EvtWriter.h, 331  
OTF2\_EvtWriter\_OmpFork  
    OTF2\_EvtWriter.h, 332  
OTF2\_EvtWriter\_OmpJoin  
    OTF2\_EvtWriter.h, 332  
OTF2\_EvtWriter\_OmpReleaseLock  
    OTF2\_EvtWriter.h, 333  
OTF2\_EvtWriter\_OmpTaskComplete  
    OTF2\_EvtWriter.h, 334  
OTF2\_EvtWriter\_OmpTaskCreate  
    OTF2\_EvtWriter.h, 334  
OTF2\_EvtWriter\_OmpTaskSwitch  
    OTF2\_EvtWriter.h, 335  
OTF2\_EvtWriter\_ParameterInt  
    OTF2\_EvtWriter.h, 336  
OTF2\_EvtWriter\_ParameterString  
    OTF2\_EvtWriter.h, 336  
OTF2\_EvtWriter\_ParameterUnsignedInt  
    OTF2\_EvtWriter.h, 337  
OTF2\_EvtWriter\_Rewind  
    OTF2\_EvtWriter.h, 338  
OTF2\_EvtWriter\_RmaAcquireLock  
    OTF2\_EvtWriter.h, 338  
OTF2\_EvtWriter\_RmaAtomic  
    OTF2\_EvtWriter.h, 339  
OTF2\_EvtWriter\_RmaCollectiveBegin  
    OTF2\_EvtWriter.h, 340  
OTF2\_EvtWriter\_RmaCollectiveEnd  
    OTF2\_EvtWriter.h, 340  
OTF2\_EvtWriter\_RmaGet  
    OTF2\_EvtWriter.h, 341  
OTF2\_EvtWriter\_RmaGroupSync  
    OTF2\_EvtWriter.h, 341  
OTF2\_EvtWriter\_RmaOpCompleteBlocking  
    OTF2\_EvtWriter.h, 342  
OTF2\_EvtWriter\_RmaOpCompleteNonBlocking  
    OTF2\_EvtWriter.h, 343  
OTF2\_EvtWriter\_RmaOpCompleteRemote  
    OTF2\_EvtWriter.h, 343  
OTF2\_EvtWriter\_RmaOpTest  
    OTF2\_EvtWriter.h, 344  
OTF2\_EvtWriter\_RmaPut  
    OTF2\_EvtWriter.h, 345  
OTF2\_EvtWriter\_RmaReleaseLock  
    OTF2\_EvtWriter.h, 345  
OTF2\_EvtWriter\_RmaRequestLock  
    OTF2\_EvtWriter.h, 346  
OTF2\_EvtWriter\_RmaSync  
    OTF2\_EvtWriter.h, 347  
OTF2\_EvtWriter\_RmaTryLock  
    OTF2\_EvtWriter.h, 347  
OTF2\_EvtWriter\_RmaWaitChange  
    OTF2\_EvtWriter.h, 348  
OTF2\_EvtWriter\_RmaWinCreate  
    OTF2\_EvtWriter.h, 348  
OTF2\_EvtWriter\_RmaWinDestroy  
    OTF2\_EvtWriter.h, 349  
OTF2\_EvtWriter\_SetLocationID  
    OTF2\_EvtWriter.h, 350  
OTF2\_EvtWriter\_SetUserData  
    OTF2\_EvtWriter.h, 350  
OTF2\_EvtWriter\_StoreRewindPoint  
    OTF2\_EvtWriter.h, 350  
OTF2\_EvtWriter\_ThreadAcquireLock  
    OTF2\_EvtWriter.h, 351  
OTF2\_EvtWriter\_ThreadFork  
    OTF2\_EvtWriter.h, 351  
OTF2\_EvtWriter\_ThreadJoin

## INDEX

---

OTF2\_EvtWriter.h, 352  
OTF2\_EvtWriter\_ThreadReleaseLock  
    OTF2\_EvtWriter.h, 352  
OTF2\_EvtWriter\_ThreadTaskComplete  
    OTF2\_EvtWriter.h, 353  
OTF2\_EvtWriter\_ThreadTaskCreate  
    OTF2\_EvtWriter.h, 354  
OTF2\_EvtWriter\_ThreadTaskSwitch  
    OTF2\_EvtWriter.h, 354  
OTF2\_EvtWriter\_ThreadTeamBegin  
    OTF2\_EvtWriter.h, 355  
OTF2\_EvtWriter\_ThreadTeamEnd  
    OTF2\_EvtWriter.h, 356  
OTF2\_FileMode\_enum  
    OTF2\_GeneralDefinitions.h, 364  
OTF2\_FileSionCallbacks, 91  
OTF2\_FileSionClose  
    OTF2\_Callbacks.h, 152  
OTF2\_FileSionGetRank  
    OTF2\_Callbacks.h, 152  
OTF2\_FileSionOpen  
    OTF2\_Callbacks.h, 152  
OTF2\_FileSubstrate\_enum  
    OTF2\_GeneralDefinitions.h, 364  
OTF2\_FileType\_enum  
    OTF2\_GeneralDefinitions.h, 364  
OTF2\_FlushCallbacks, 91  
OTF2\_FlushType\_enum  
    OTF2\_GeneralDefinitions.h, 365  
OTF2\_GeneralDefinitions.h, 356  
    OTF2\_CallbackCode, 363  
        OTF2\_Compression\_enum, 363  
        OTF2\_FileMode\_enum, 364  
        OTF2\_FileSubstrate\_enum, 364  
        OTF2\_FileType\_enum, 364  
        OTF2\_FlushType\_enum, 365  
        OTF2\_MappingType\_enum, 365  
        OTF2\_Paradigm\_enum, 366  
        OTF2\_ThumbnailType\_enum, 366  
        OTF2\_Type\_enum, 367  
        OTF2\_UNDEFINED\_TYPE, 363  
OTF2\_GlobalDefReader.h, 367  
    OTF2\_GlobalDefReader\_ReadDefinitions, 368  
            OTF2\_GlobalDefReader\_SetCallbacks, 369  
OTF2\_GlobalDefReader\_ReadDefinitions  
    OTF2\_GlobalDefReader.h, 368  
OTF2\_GlobalDefReader\_SetCallbacks  
    OTF2\_GlobalDefReader.h, 369  
OTF2\_GlobalDefReaderCallback\_Attribute  
    OTF2\_GlobalDefReaderCallbacks.h, 375  
OTF2\_GlobalDefReaderCallback\_Callpath  
    OTF2\_GlobalDefReaderCallbacks.h, 375  
OTF2\_GlobalDefReaderCallback\_Callsite  
    OTF2\_GlobalDefReaderCallbacks.h, 376  
OTF2\_GlobalDefReaderCallback\_ClockProperties  
    OTF2\_GlobalDefReaderCallbacks.h, 376  
OTF2\_GlobalDefReaderCallback\_Comm  
    OTF2\_GlobalDefReaderCallbacks.h, 377  
OTF2\_GlobalDefReaderCallback\_Group  
    OTF2\_GlobalDefReaderCallbacks.h, 378  
OTF2\_GlobalDefReaderCallback\_Location  
    OTF2\_GlobalDefReaderCallbacks.h, 378  
OTF2\_GlobalDefReaderCallback\_LocationGroup  
    OTF2\_GlobalDefReaderCallbacks.h, 379  
OTF2\_GlobalDefReaderCallback\_MetricClass  
    OTF2\_GlobalDefReaderCallbacks.h, 380  
OTF2\_GlobalDefReaderCallback\_MetricClassRecorder  
    OTF2\_GlobalDefReaderCallbacks.h, 380  
OTF2\_GlobalDefReaderCallback\_MetricInstance  
    OTF2\_GlobalDefReaderCallbacks.h, 381  
OTF2\_GlobalDefReaderCallback\_MetricMember  
    OTF2\_GlobalDefReaderCallbacks.h, 382  
            OTF2\_GlobalDefReaderCallback\_Parameter

## INDEX

---

OTF2\_GlobalDefReaderCallbacks.h, 383  
OTF2\_GlobalDefReaderCallback\_Region  
    OTF2\_GlobalDefReaderCallbacks.h, 383  
OTF2\_GlobalDefReaderCallback\_RmaWin  
    OTF2\_GlobalDefReaderCallbacks.h, 384  
OTF2\_GlobalDefReaderCallback\_String  
    OTF2\_GlobalDefReaderCallbacks.h, 385  
OTF2\_GlobalDefReaderCallback\_SystemTreeNodString, 385  
    OTF2\_GlobalDefReaderCallbacks.h, 385  
OTF2\_GlobalDefReaderCallback\_SystemTreeNodeDomain, 386  
    OTF2\_GlobalDefReaderCallbacks.h, 386  
OTF2\_GlobalDefReaderCallback\_SystemTreeNodeProperty, 386  
    OTF2\_GlobalDefReaderCallbacks.h, 386  
OTF2\_GlobalDefReaderCallback\_Uncertain, 387  
    OTF2\_GlobalDefReaderCallbacks.h, 387  
OTF2\_GlobalDefReaderCallbacks.h, 389  
    OTF2\_GlobalDefReaderCallback\_Attribute, 375  
        OTF2\_GlobalDefReaderCallback\_Callpath, 375  
        OTF2\_GlobalDefReaderCallback\_Callsite, 376  
        OTF2\_GlobalDefReaderCallback\_ClockProperties, 376  
        OTF2\_GlobalDefReaderCallback\_Comm, 377  
        OTF2\_GlobalDefReaderCallback\_Group, 378  
        OTF2\_GlobalDefReaderCallback\_Location, 378  
        OTF2\_GlobalDefReaderCallback\_LocationGroup, 379  
        OTF2\_GlobalDefReaderCallback\_MetricClass, 380  
        OTF2\_GlobalDefReaderCallback\_MetricClassRecorder, 380  
    OTF2\_GlobalDefReaderCallback\_MetricInstance, 381  
    OTF2\_GlobalDefReaderCallback\_MetricMember, 382  
    OTF2\_GlobalDefReaderCallback\_Parameter, 383  
OTF2\_GlobalDefReaderCallback\_Region, 383  
OTF2\_GlobalDefReaderCallback\_RmaWin, 384  
OTF2\_GlobalDefReaderCallback\_SystemTreeNode, 385  
OTF2\_GlobalDefReaderCallback\_SystemTreeNodeDomain, 386  
OTF2\_GlobalDefReaderCallback\_SystemTreeNodeProperty, 386  
OTF2\_GlobalDefReaderCallback\_Unknown, 387  
OTF2\_GlobalDefReaderCallbacks\_Clear, 388  
OTF2\_GlobalDefReaderCallbacks\_Delete, 388  
OTF2\_GlobalDefReaderCallbacks\_New, 388  
OTF2\_GlobalDefReaderCallbacks\_SetAttributeCallback, 388  
OTF2\_GlobalDefReaderCallbacks\_SetCallpathCallback, 389  
OTF2\_GlobalDefReaderCallbacks\_SetCallsiteCallback, 389  
OTF2\_GlobalDefReaderCallbacks\_SetClockPropertiesCallback, 390  
OTF2\_GlobalDefReaderCallbacks\_SetCommCallback, 390  
OTF2\_GlobalDefReaderCallbacks\_SetGroupCallback, 391  
OTF2\_GlobalDefReaderCallbacks\_SetLocationCallback, 391  
OTF2\_GlobalDefReaderCallbacks\_SetLocationGroupCallback, 392  
OTF2\_GlobalDefReaderCallbacks\_SetMetricClassCallback, 392

## INDEX

---

OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks.h,  
    SetMetricClassRecorderCallback, [390](#)  
        [393](#) OTF2\_GlobalDefReaderCallbacks\_SetCommCallback  
OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks.h,  
    SetMetricInstanceCallback, [393](#) [390](#)  
OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetGroupCallback  
    SetMetricMemberCallback, [394](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_- [391](#)  
    SetParameterCallback, [394](#) OTF2\_GlobalDefReaderCallbacks\_SetLocationCallback  
OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks.h,  
    SetRegionCallback, [395](#) [391](#)  
OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetLocationGroupCallback  
    SetRmaWinCallback, [395](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_- [392](#)  
    SetStringCallback, [396](#) OTF2\_GlobalDefReaderCallbacks\_SetMetricClassCallback  
OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks.h,  
    SetSystemTreeNodeCallback, [396](#) [392](#)  
OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetMetricClassRecorderCallback  
    SetSystemTreeNodeDomainCallback,OTF2\_GlobalDefReaderCallbacks.h,  
        [397](#) [393](#)  
OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetMetricInstanceCallback  
    SetSystemTreeNodePropertyCallbackOTF2\_GlobalDefReaderCallbacks.h,  
        [398](#) [393](#)  
OTF2\_GlobalDefReaderCallbacks\_- OTF2\_GlobalDefReaderCallbacks\_SetMetricMemberCallback  
    SetUnknownCallback, [398](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_Clear [394](#)  
    OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetParameterCallback  
        [388](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_Delete [394](#)  
    OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetRegionCallback  
        [388](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_New [395](#)  
    OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetRmaWinCallback  
        [388](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_SetAttributeCall[396](#)  
    OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetStringCallback  
        [388](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_SetCallpathCall[396](#)  
    OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetSystemTreeNodeCallback  
        [389](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_SetCallsiteCall[396](#)  
    OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefReaderCallbacks\_SetSystemTreeNodeDomainCallback  
        [389](#) OTF2\_GlobalDefReaderCallbacks.h,  
OTF2\_GlobalDefReaderCallbacks\_SetClockProper[397](#)Callback

## INDEX

---

OTF2\_GlobalDefReaderCallbacks\_SetSystemTreeNode, OTF2\_GlobalDefWriter\_CallbackWriteSystemTreeNode,  
OTF2\_GlobalDefReaderCallbacks.h, 413  
398 OTF2\_GlobalDefWriter\_WriteSystemTreeNodeDomain,  
OTF2\_GlobalDefReaderCallbacks\_SetUnknownCallback, 414  
OTF2\_GlobalDefReaderCallbacks.h, OTF2\_GlobalDefWriter\_WriteSystemTreeNodeProperty,  
398 414  
OTF2\_GlobalDefWriter.h, 399 OTF2\_GlobalDefWriter\_GetNumberOfDefinitions  
OTF2\_GlobalDefWriter\_GetNumberOfDefinitions, OTF2\_GlobalDefWriter.h, 402  
402 OTF2\_GlobalDefWriter\_GetNumberOfLocations  
OTF2\_GlobalDefWriter\_GetNumberOfLocations, OTF2\_GlobalDefWriter.h, 402  
402 OTF2\_GlobalDefWriter\_WriteAttribute  
OTF2\_GlobalDefWriter\_WriteAttribute, OTF2\_GlobalDefWriter.h, 403  
403 OTF2\_GlobalDefWriter\_WriteCallpath  
OTF2\_GlobalDefWriter\_WriteCallpath, OTF2\_GlobalDefWriter.h, 403  
403 OTF2\_GlobalDefWriter\_WriteCallsite  
OTF2\_GlobalDefWriter\_WriteCallsite, OTF2\_GlobalDefWriter.h, 404  
404 OTF2\_GlobalDefWriter\_WriteClockProperties  
OTF2\_GlobalDefWriter\_WriteClockProperties, OTF2\_GlobalDefWriter.h, 404  
404 OTF2\_GlobalDefWriter\_WriteComm  
OTF2\_GlobalDefWriter\_WriteComm, OTF2\_GlobalDefWriter.h, 405  
405 OTF2\_GlobalDefWriter\_WriteGroup  
OTF2\_GlobalDefWriter\_WriteGroup, OTF2\_GlobalDefWriter.h, 406  
406 OTF2\_GlobalDefWriter\_WriteLocation  
OTF2\_GlobalDefWriter\_WriteLocation, OTF2\_GlobalDefWriter\_WriteLocationGroup  
406 406 OTF2\_GlobalDefWriter.h, 407  
OTF2\_GlobalDefWriter\_WriteLocationGroup, OTF2\_GlobalDefWriter\_WriteMetricClass  
407 OTF2\_GlobalDefWriter.h, 408  
OTF2\_GlobalDefWriter\_WriteMetricClass, OTF2\_GlobalDefWriter\_WriteMetricClassRecorder  
408 OTF2\_GlobalDefWriter.h, 408  
OTF2\_GlobalDefWriter\_WriteMetricClassRecorder, OTF2\_GlobalDefWriter\_WriteMetricInstance  
408 OTF2\_GlobalDefWriter.h, 409  
OTF2\_GlobalDefWriter\_WriteMetricInstance, OTF2\_GlobalDefWriter\_WriteMetricMember  
409 OTF2\_GlobalDefWriter.h, 410  
OTF2\_GlobalDefWriter\_WriteMetricMember, OTF2\_GlobalDefWriter\_WriteParameter  
410 OTF2\_GlobalDefWriter.h, 411  
OTF2\_GlobalDefWriter\_WriteParameter, OTF2\_GlobalDefWriter\_WriteRegion  
411 OTF2\_GlobalDefWriter.h, 411  
OTF2\_GlobalDefWriter\_WriteRegion, OTF2\_GlobalDefWriter\_WriteRmaWin  
411 OTF2\_GlobalDefWriter.h, 412  
OTF2\_GlobalDefWriter\_WriteRmaWin, OTF2\_GlobalDefWriter\_WriteString  
412 OTF2\_GlobalDefWriter.h, 413  
OTF2\_GlobalDefWriter\_WriteString, OTF2\_GlobalDefWriter\_WriteSystemTreeNode  
413 OTF2\_GlobalDefWriter.h, 413

## INDEX

OTF2\_GlobalEvtReader\_WriteSystemTreeNodes  
OTF2\_GlobalEvtReader.h, 414

OTF2\_GlobalEvtReader\_WriteSystemTreeNodes  
OTF2\_GlobalEvtReader.h, 414

OTF2\_GlobalEvtReader.h, 415

OTF2\_GlobalEvtReader\_HasEvent, 416

OTF2\_GlobalEvtReader\_ReadEvent, 416

OTF2\_GlobalEvtReader\_ReadEvents, 416

OTF2\_GlobalEvtReader\_SetCallbacks, 417

OTF2\_GlobalEvtReader\_HasEvent  
OTF2\_GlobalEvtReader.h, 416

OTF2\_GlobalEvtReader\_ReadEvent  
OTF2\_GlobalEvtReader.h, 416

OTF2\_GlobalEvtReader\_ReadEvents  
OTF2\_GlobalEvtReader.h, 416

OTF2\_GlobalEvtReader\_SetCallbacks  
OTF2\_GlobalEvtReader.h, 417

OTF2\_GlobalEvtReaderCallback\_BufferFlush  
OTF2\_GlobalEvtReaderCallbacks.h, 430

OTF2\_GlobalEvtReaderCallback\_Enter  
OTF2\_GlobalEvtReaderCallbacks.h, 431

OTF2\_GlobalEvtReaderCallback\_Leave  
OTF2\_GlobalEvtReaderCallbacks.h, 431

OTF2\_GlobalEvtReaderCallback\_MeasurementOnOff  
OTF2\_GlobalEvtReaderCallbacks.h, 432

OTF2\_GlobalEvtReaderCallback\_Metric  
OTF2\_GlobalEvtReaderCallbacks.h, 432

OTF2\_GlobalEvtReaderCallback\_MpiCollectiveBegin  
OTF2\_GlobalEvtReaderCallbacks.h, 433

OTF2\_GlobalEvtReaderCallback\_MpiCollectiveEnd  
OTF2\_GlobalEvtReaderCallbacks.h, 434

OTF2\_GlobalEvtReaderCallback\_MpiIrecv  
OTF2\_GlobalEvtReaderCallbacks.h, 435

OTF2\_GlobalEvtReaderCallback\_MpiIrecvRequest  
OTF2\_GlobalEvtReaderCallbacks.h, 435

OTF2\_GlobalEvtReaderCallback\_MpiIsend  
OTF2\_GlobalEvtReaderCallbacks.h, 436

OTF2\_GlobalEvtReaderCallback\_MpiIsendComplete  
OTF2\_GlobalEvtReaderCallbacks.h, 437

OTF2\_GlobalEvtReaderCallback\_MpiRecv  
OTF2\_GlobalEvtReaderCallbacks.h, 437

OTF2\_GlobalEvtReaderCallback\_MpiRequestCancelled  
OTF2\_GlobalEvtReaderCallbacks.h, 438

OTF2\_GlobalEvtReaderCallback\_MpiRequestTest  
OTF2\_GlobalEvtReaderCallbacks.h, 439

OTF2\_GlobalEvtReaderCallback\_MpiSend  
OTF2\_GlobalEvtReaderCallbacks.h, 439

OTF2\_GlobalEvtReaderCallback\_OmpAcquireLock  
OTF2\_GlobalEvtReaderCallbacks.h, 440

OTF2\_GlobalEvtReaderCallback\_OmpFork  
OTF2\_GlobalEvtReaderCallbacks.h, 441

OTF2\_GlobalEvtReaderCallback\_OmpJoin  
OTF2\_GlobalEvtReaderCallbacks.h, 442

OTF2\_GlobalEvtReaderCallback\_OmpReleaseLock  
OTF2\_GlobalEvtReaderCallbacks.h, 442

OTF2\_GlobalEvtReaderCallback\_OmpTaskComplete  
OTF2\_GlobalEvtReaderCallbacks.h, 443

OTF2\_GlobalEvtReaderCallback\_OmpTaskCreate  
OTF2\_GlobalEvtReaderCallbacks.h, 443

OTF2\_GlobalEvtReaderCallback\_OmpTaskSwitch  
OTF2\_GlobalEvtReaderCallbacks.h, 444

OTF2\_GlobalEvtReaderCallback\_MpiIrecv  
OTF2\_GlobalEvtReaderCallbacks.h, 444

## INDEX

## INDEX

---

OTF2\_GlobalEvtReaderCallback\_ThreadTeam  
    OTF2\_GlobalEvtReaderCallbacks.h,  
        465

OTF2\_GlobalEvtReaderCallback\_Unknown  
    OTF2\_GlobalEvtReaderCallbacks.h,  
        465

OTF2\_GlobalEvtReaderCallbacks.h, 418  
    OTF2\_GlobalEvtReaderCallback\_-  
        BufferFlush, 430

    OTF2\_GlobalEvtReaderCallback\_-  
        Enter, 431

    OTF2\_GlobalEvtReaderCallback\_-  
        Leave, 431

    OTF2\_GlobalEvtReaderCallback\_-  
        MeasurementOnOff, 432

    OTF2\_GlobalEvtReaderCallback\_-  
        Metric, 432

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiCollectiveBegin, 433

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiCollectiveEnd, 434

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiIrecv, 435

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiIrecvRequest, 435

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiIsend, 436

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiIsendComplete, 437

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiRecv, 437

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiRequestCancelled, 438

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiRequestTest, 439

    OTF2\_GlobalEvtReaderCallback\_-  
        MpiSend, 439

    OTF2\_GlobalEvtReaderCallback\_-  
        OmpAcquireLock, 440

    OTF2\_GlobalEvtReaderCallback\_-  
        OmpFork, 441

    OTF2\_GlobalEvtReaderCallback\_-  
        OmpJoin, 442

OTF2\_GlobalEvtReaderCallback\_-  
    OmpReleaseLock, 442

OTF2\_GlobalEvtReaderCallback\_-  
    OmpTaskComplete, 443

OTF2\_GlobalEvtReaderCallback\_-  
    OmpTaskCreate, 444

OTF2\_GlobalEvtReaderCallback\_-  
    OmpTaskSwitch, 444

OTF2\_GlobalEvtReaderCallback\_-  
    ParameterInt, 445

OTF2\_GlobalEvtReaderCallback\_-  
    ParameterString, 445

OTF2\_GlobalEvtReaderCallback\_-  
    ParameterUnsignedInt, 446

OTF2\_GlobalEvtReaderCallback\_-  
    RmaAcquireLock, 447

OTF2\_GlobalEvtReaderCallback\_-  
    RmaAtomic, 448

OTF2\_GlobalEvtReaderCallback\_-  
    RmaCollectiveBegin, 448

OTF2\_GlobalEvtReaderCallback\_-  
    RmaCollectiveEnd, 449

OTF2\_GlobalEvtReaderCallback\_-  
    RmaGet, 450

OTF2\_GlobalEvtReaderCallback\_-  
    RmaGroupSync, 451

OTF2\_GlobalEvtReaderCallback\_-  
    RmaOpCompleteBlocking, 451

OTF2\_GlobalEvtReaderCallback\_-  
    RmaOpCompleteNonBlocking,  
        452

OTF2\_GlobalEvtReaderCallback\_-  
    RmaOpCompleteRemote, 453

OTF2\_GlobalEvtReaderCallback\_-  
    RmaOpTest, 453

OTF2\_GlobalEvtReaderCallback\_-  
    RmaPut, 454

OTF2\_GlobalEvtReaderCallback\_-  
    RmaReleaseLock, 455

OTF2\_GlobalEvtReaderCallback\_-  
    RmaRequestLock, 455

OTF2\_GlobalEvtReaderCallback\_-  
    RmaSync, 456

## INDEX

---

OTF2\_GlobalEvtReaderCallback\_-  
    RmaTryLock, [457](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    RmaWaitChange, [457](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    RmaWinCreate, [458](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    RmaWinDestroy, [459](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadAcquireLock, [459](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadFork, [460](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadJoin, [461](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadReleaseLock, [461](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadTaskComplete, [462](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadTaskCreate, [463](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadTaskSwitch, [463](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadTeamBegin, [464](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    ThreadTeamEnd, [465](#)  
OTF2\_GlobalEvtReaderCallback\_-  
    Unknown, [465](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    Clear, [466](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    Delete, [466](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    New, [466](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetBufferFlushCallback, [467](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetEnterCallback, [467](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetLeaveCallback, [468](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMeasurementOnOffCallback,  
        [468](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMetricCallback, [469](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpICollectiveBeginCallback,  
        [469](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpICollectiveEndCallback,  
        [470](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpIRecvCallback, [470](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpIRecvRequestCallback, [471](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpISendCallback, [471](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpISendCompleteCallback,  
        [472](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpIRecvCallback, [472](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpIRequestCancelledCallback,  
        [473](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpIRequestTestCallback, [473](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetMpISendCallback, [474](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetOmpAcquireLockCallback,  
        [474](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetOmpForkCallback, [475](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetOmpJoinCallback, [475](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetOmpReleaseLockCallback, [476](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetOmpTaskCompleteCallback,  
        [476](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetOmpTaskCreateCallback, [477](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetOmpTaskSwitchCallback, [477](#)  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetParameterIntCallback, [478](#)

## INDEX

---

OTF2\_GlobalEvtReaderCallbacks\_-  
    SetParameterStringCallback, 478  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetParameterUnsignedIntCallback,  
        479  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaAcquireLockCallback, 480  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaAtomicCallback, 480  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaCollectiveBeginCallback,  
        481  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaCollectiveEndCallback,  
        481  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaGetCallback, 482  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaGroupSyncCallback, 482  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaOpCompleteBlockingCallback,  
        483  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaOpCompleteNonBlockingCallback  
        483  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaOpCompleteRemoteCallback, OTF2\_GlobalEvtReaderCallbacks\_-  
        484  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaOpTestCallback, 484  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaPutCallback, 485  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaReleaseLockCallback, 485  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaRequestLockCallback, 486  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaSyncCallback, 486  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaTryLockCallback, 487  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaWaitChangeCallback, 487  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaWinCreateCallback, 488  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetRmaWinDestroyCallback, 488  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadAcquireLockCallback,  
        489  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadForkCallback, 489  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadJoinCallback, 490  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadReleaseLockCallback,  
        490  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadTaskCompleteCallback,  
        491  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadTaskCreateCallback,  
        491  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadTaskSwitchCallback,  
        492  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadTeamBeginCallback,  
        492  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetThreadTeamEndCallback, 493  
OTF2\_GlobalEvtReaderCallbacks\_-  
    SetUnknownCallback, 493  
OTF2\_GlobalEvtReaderCallbacks\_Clear  
    OTF2\_GlobalEvtReaderCallbacks.h,  
        466  
OTF2\_GlobalEvtReaderCallbacks\_Delete  
    OTF2\_GlobalEvtReaderCallbacks.h,  
        466  
OTF2\_GlobalEvtReaderCallbacks\_New  
    OTF2\_GlobalEvtReaderCallbacks.h,  
        466  
OTF2\_GlobalEvtReaderCallbacks\_SetBufferFlushCallback  
    OTF2\_GlobalEvtReaderCallbacks.h,  
        467  
OTF2\_GlobalEvtReaderCallbacks\_SetEnterCallback  
    OTF2\_GlobalEvtReaderCallbacks.h,  
        467  
OTF2\_GlobalEvtReaderCallbacks\_SetLeaveCallback  
    OTF2\_GlobalEvtReaderCallbacks.h,  
        467

## INDEX

---

OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetOmpJoinCallback  
468 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMeasuremen405 OffCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetOmpReleaseLockCallback  
468 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMetricCallba476  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskCompleteCallback  
469 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpICollectiv476  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskCreateCallback  
469 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpICollectiv476  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetOmpTaskSwitchCallback  
470 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpIRecvCallba477  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetParameterIntCallback  
470 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpIRecvRequestCallback478  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetParameterStringCallback  
471 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpISendCallba478  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetParameterUnsignedIntCallback  
471 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpISendCompleteCallback479  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaAcquireLockCallback  
472 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpIRecvCallba480  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaAtomicCallback  
472 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpIRequestCancelledCallback480  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaCollectiveBeginCallback  
473 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpIRequestTestCallback481  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaCollectiveEndCallback  
473 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetMpISendCallba481  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaGetCallback  
474 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetOmpAcquireLockCallba482  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaGroupSyncCallback  
474 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetOmpForBlock482  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteBlockingCallba

## INDEX

---

OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadReleaseLockCallback  
483 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteNonBlockingCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskCompleteCallback  
483 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpCompleteRemoteCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskCreateCallback  
484 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaOpTestCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadTaskSwitchCallback  
484 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaPutCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadTeamBeginCallback  
485 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaReleaseLockCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetThreadTeamEndCallback  
485 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaRequestLockCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalEvtReaderCallbacks\_SetUnknownCallback  
486 OTF2\_GlobalEvtReaderCallbacks.h,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaSyncCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalSnapReader.h, 494  
486 OTF2\_GlobalSnapReader\_ReadSnapshots,  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaTryLockCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 495 OTF2\_GlobalSnapReader\_SetCallbacks,  
487 OTF2\_GlobalSnapReader.h, 495  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaWaitChangeCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalSnapReader\_ReadSnapshots  
487 OTF2\_GlobalSnapReader.h, 495  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaWinCreateCallback  
OTF2\_GlobalEvtReaderCallbacks.h, OTF2\_GlobalSnapReader.h, 495  
488 OTF2\_GlobalSnapReader\_SetCallbacks  
OTF2\_GlobalEvtReaderCallbacks\_SetRmaWinDestroyCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 502 OTF2\_GlobalSnapReaderCallback\_Enter  
488 OTF2\_GlobalSnapReaderCallback\_MeasurementOnOff  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadAcquireLockCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 502 OTF2\_GlobalSnapReaderCallbacks.h,  
489 OTF2\_GlobalSnapReaderCallback\_Metric  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadFreeLockCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 503 OTF2\_GlobalSnapReaderCallbacks.h,  
489 OTF2\_GlobalSnapReaderCallback\_MpiCollectiveBegin  
OTF2\_GlobalEvtReaderCallbacks\_SetThreadJoinCallback  
OTF2\_GlobalEvtReaderCallbacks.h, 504 OTF2\_GlobalSnapReaderCallbacks.h,  
490 OTF2\_GlobalSnapReaderCallback\_MpiCollectiveEnd

---

## INDEX

OTF2\_GlobalSnapReaderCallbacks.hOTF2\_GlobalSnapReaderCallback\_SnapshotStart  
505 OTF2\_GlobalSnapReaderCallbacks.h,  
OTF2\_GlobalSnapReaderCallback\_MpiIrecv 516  
OTF2\_GlobalSnapReaderCallbacks.hOTF2\_GlobalSnapReaderCallback\_Unknown  
506 OTF2\_GlobalSnapReaderCallbacks.h,  
OTF2\_GlobalSnapReaderCallback\_MpiIrecvRequest 517  
OTF2\_GlobalSnapReaderCallbacks.hOTF2\_GlobalSnapReaderCallbacks  
506 OTF2\_GlobalSnapReaderCallbacks.h,  
OTF2\_GlobalSnapReaderCallback\_MpiIsend 517  
OTF2\_GlobalSnapReaderCallbacks.hOTF2\_GlobalSnapReaderCallbacks.h, 496  
507 OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallback\_MpiIsendComplete 502  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_GlobalSnapReaderCallback\_-  
508 MeasurementOnOff, 502  
OTF2\_GlobalSnapReaderCallback\_MpiRecv OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, Metric, 503  
509 OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallback\_MpiSend OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, MpiCollectiveBegin, 504  
510 OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, MpiCollectiveEnd, 505  
OTF2\_GlobalSnapReaderCallback\_OmpAcquireLockOTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, MpiIrecv, 506  
510 OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallback\_OmpFork OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, MpiIrecvRequest, 506  
511 OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, MpiIsend, 507  
OTF2\_GlobalSnapReaderCallback\_OmpTaskOTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, MpiIsendComplete, 508  
512 OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallback\_OmpTaskSwitchMpiRecv, 509  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_GlobalSnapReaderCallback\_-  
513 MpiSend, 510  
OTF2\_GlobalSnapReaderCallback\_ParameterOTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, OmpAcquireLock, 510  
513 OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallback\_ParameterStringOmpFork, 511  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_GlobalSnapReaderCallback\_-  
514 OmpTaskCreate, 512  
OTF2\_GlobalSnapReaderCallback\_ParameterOTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallbacks.h, OmpTaskSwitch, 513  
515 OTF2\_GlobalSnapReaderCallback\_-  
OTF2\_GlobalSnapReaderCallback\_SnapshotEnd ParameterInt, 513  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_GlobalSnapReaderCallback\_-  
516 ParameterString, 514

---

## INDEX

---

OTF2\_GlobalSnapReaderCallback\_-  
ParameterUnsignedInt, 515  
OTF2\_GlobalSnapReaderCallback\_-  
SnapshotEnd, 516  
OTF2\_GlobalSnapReaderCallback\_-  
SnapshotStart, 516  
OTF2\_GlobalSnapReaderCallback\_-  
Unknown, 517  
OTF2\_GlobalSnapReaderCallbacks,  
517  
OTF2\_GlobalSnapReaderCallbacks\_-  
Clear, 518  
OTF2\_GlobalSnapReaderCallbacks\_-  
Delete, 518  
OTF2\_GlobalSnapReaderCallbacks\_-  
New, 518  
OTF2\_GlobalSnapReaderCallbacks\_-  
SetEnterCallback, 519  
OTF2\_GlobalSnapReaderCallbacks\_-  
SetMeasurementOnOffCallback,  
519  
OTF2\_GlobalSnapReaderCallbacks\_-  
SetMetricCallback, 520  
OTF2\_GlobalSnapReaderCallbacks\_-OTF2\_GlobalSnapReaderCallbacks\_Delete  
SetMpICollectiveBeginCallback, OTF2\_GlobalSnapReaderCallbacks.h,  
520 518  
OTF2\_GlobalSnapReaderCallbacks\_-OTF2\_GlobalSnapReaderCallbacks\_New  
SetMpICollectiveEndCallback, OTF2\_GlobalSnapReaderCallbacks.h,  
521 518  
OTF2\_GlobalSnapReaderCallbacks\_-OTF2\_GlobalSnapReaderCallbacks\_SetEnterCallback  
SetMpIRecvCallback, 522 OTF2\_GlobalSnapReaderCallbacks.h,  
OTF2\_GlobalSnapReaderCallbacks\_- 519  
SetMpIRecvRequestCallback, 520 OTF2\_GlobalSnapReaderCallbacks\_SetMeasurementOnOffCallback  
OTF2\_GlobalSnapReaderCallbacks\_- OTF2\_GlobalSnapReaderCallbacks.h,  
SetMpISendCallback, 523 519  
OTF2\_GlobalSnapReaderCallbacks\_-OTF2\_GlobalSnapReaderCallbacks\_SetMetricCallback  
SetMpISendCompleteCallback, OTF2\_GlobalSnapReaderCallbacks.h,  
523 520  
OTF2\_GlobalSnapReaderCallbacks\_-OTF2\_GlobalSnapReaderCallbacks\_SetMpICollectiveBeginCallback  
SetMpRecvCallback, 524 OTF2\_GlobalSnapReaderCallbacks.h,  
OTF2\_GlobalSnapReaderCallbacks\_- 520  
SetMpISendCallback, 525 OTF2\_GlobalSnapReaderCallbacks\_SetMpICollectiveEndCallback  
OTF2\_GlobalSnapReaderCallbacks\_- OTF2\_GlobalSnapReaderCallbacks.h,  
SetOmpAcquireLockCallback, 521

## INDEX

---

OTF2\_GlobalSnapReaderCallbacks\_SetMpIrcallback  
OTF2\_GlobalSnapReaderCallbacks.h, 530  
522 OTF2\_GlobalSnapReaderCallbacks\_SetUnknownCallback

OTF2\_GlobalSnapReaderCallbacks\_SetMpIrcallback  
OTF2\_GlobalSnapReaderCallbacks.h, 530  
522 OTF2\_GroupFlag\_enum

OTF2\_GlobalSnapReaderCallbacks\_SetMpIsendCallback  
OTF2\_GlobalSnapReaderCallbacks.h, 161  
OTF2\_GroupType\_enum  
523 OTF2\_Definitions.h, 161

OTF2\_GlobalSnapReaderCallbacks\_SetMpHsendCompleteCallback  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h, 533  
523 OTF2\_IdMap.h, 531

OTF2\_GlobalSnapReaderCallbacks\_SetMpIRecvCallback  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h, 533  
524 OTF2\_IdMap\_Clear, 534

OTF2\_GlobalSnapReaderCallbacks\_SetMpISendCallback  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_Create, 534  
525 OTF2\_IdMap\_CreateFromUint32Array,  
534

OTF2\_GlobalSnapReaderCallbacks\_SetOmpAcquireLockCallback  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_CreateFromUint64Array,  
535  
525 OTF2\_IdMap\_Free, 535

OTF2\_GlobalSnapReaderCallbacks\_SetOmpForkCallback  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_GetGlobalId, 536  
526 OTF2\_IdMap\_GetMode, 536  
OTF2\_IdMap\_GetSize, 536  
OTF2\_IdMap\_Traverse, 537

OTF2\_GlobalSnapReaderCallbacks\_SetOmpTaskCreateCallback  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMapMode, 533  
526 OTF2\_IdMapMode\_enum, 533

OTF2\_IdMap\_AddIdPair  
OTF2\_GlobalSnapReaderCallbacks\_SetOmpTaskSwitchCallback  
OTF2\_IdMap.h, 533  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_Clear  
527 OTF2\_IdMap.h, 534

OTF2\_GlobalSnapReaderCallbacks\_SetParameterIntCallback  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_Create  
527 OTF2\_IdMap\_CreateFromUint32Array

OTF2\_GlobalSnapReaderCallbacks\_SetParameterStringCallback  
OTF2\_IdMap.h, 534  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_CreateFromUint64Array  
528 OTF2\_IdMap.h, 535

OTF2\_GlobalSnapReaderCallbacks\_SetParameterUnsignedIntCallback  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap.h, 535  
529 OTF2\_IdMap\_GetGlobalId

OTF2\_GlobalSnapReaderCallbacks\_SetSnapshotCallback  
OTF2\_IdMap.h, 536  
OTF2\_GlobalSnapReaderCallbacks.h, OTF2\_IdMap\_GetMode  
529 OTF2\_IdMap.h, 536

OTF2\_GlobalSnapReaderCallbacks\_SetSnapshotBitmapSize

## INDEX

---

OTF2\_IdMap.h, 536  
OTF2\_IdMap\_Traverse  
    OTF2\_IdMap.h, 537  
OTF2\_IdMapMode  
    OTF2\_IdMap.h, 533  
OTF2\_IdMapMode\_enum  
    OTF2\_IdMap.h, 533  
OTF2\_LocationGroupType\_enum  
    OTF2\_Definitions.h, 162  
OTF2\_LocationType\_enum  
    OTF2\_Definitions.h, 162  
OTF2\_LockType\_enum  
    OTF2\_Events.h, 228  
OTF2\_MappingType\_enum  
    OTF2\_GeneralDefinitions.h, 365  
OTF2\_Marker.h, 537  
    OTF2\_MarkerScope\_enum, 538  
    OTF2\_MarkerSeverity\_enum, 538  
OTF2\_MarkerReader.h, 539  
    OTF2\_MarkerReader\_ReadMarkers,  
        540  
    OTF2\_MarkerReader\_SetCallbacks,  
        540  
OTF2\_MarkerReader\_ReadMarkers  
    OTF2\_MarkerReader.h, 540  
OTF2\_MarkerReader\_SetCallbacks  
    OTF2\_MarkerReader.h, 540  
OTF2\_MarkerReaderCallback\_DefMarker  
    OTF2\_MarkerReaderCallbacks.h, 542  
OTF2\_MarkerReaderCallback\_Marker  
    OTF2\_MarkerReaderCallbacks.h, 543  
OTF2\_MarkerReaderCallback\_Uncertain  
    OTF2\_MarkerReaderCallbacks.h, 543  
OTF2\_MarkerReaderCallbacks.h, 541  
    OTF2\_MasterSlaveMode  
        OTF2\_MarkerReaderCallback\_DefMarker  
            542  
        OTF2\_MasterSlaveMode\_enum  
    OTF2\_MarkerReaderCallback\_Marker,  
        OTF2\_Archive.h, 102  
            543  
        OTF2\_MeasurementMode\_enum  
    OTF2\_MarkerReaderCallback\_Uncertain  
        OTF2\_Events.h, 228  
            543  
        OTF2\_MemoryAllocate  
    OTF2\_MarkerReaderCallbacks\_Clear,  
        OTF2\_Callbacks.h, 153  
            544  
        OTF2\_MemoryCallbacks, 92  
    OTF2\_MarkerReaderCallbacks\_Delete  
        OTF2\_MemoryFreeAll  
            544  
        OTF2\_Callbacks.h, 154

## INDEX

---

OTF2\_MetricBase\_enum  
    OTF2\_Definitions.h, 163

OTF2\_MetricMode\_enum  
    OTF2\_Definitions.h, 163

OTF2\_MetricOccurrence\_enum  
    OTF2\_Definitions.h, 163

OTF2\_MetricScope\_enum  
    OTF2\_Definitions.h, 164

OTF2\_MetricTiming\_enum  
    OTF2\_Definitions.h, 164

OTF2\_MetricType\_enum  
    OTF2\_Definitions.h, 165

OTF2\_MetricValue\_union, 92

OTF2\_MetricValueProperty\_enum  
    OTF2\_Definitions.h, 165

OTF2\_Paradigm\_enum  
    OTF2\_GeneralDefinitions.h, 366

OTF2\_ParameterType\_enum  
    OTF2\_Definitions.h, 166

OTF2\_PostFlushCallback  
    OTF2\_Callbacks.h, 154

OTF2\_PreFlushCallback  
    OTF2\_Callbacks.h, 155

OTF2\_Reader.h, 549

    OTF2\_Reader\_Close, 554

    OTF2\_Reader\_CloseDefReader, 555

    OTF2\_Reader\_CloseEvtReader, 555

    OTF2\_Reader\_CloseGlobalDefReader,  
        555

    OTF2\_Reader\_CloseGlobalEvtReader,  
        556

    OTF2\_Reader\_CloseGlobalSnapReader,  
        556

    OTF2\_Reader\_CloseMarkerReader,  
        556

    OTF2\_Reader\_CloseMarkerWriter,  
        557

    OTF2\_Reader\_CloseSnapReader, 557

    OTF2\_Reader\_CloseThumbReader,  
        558

    OTF2\_Reader\_GetBoolProperty, 558

    OTF2\_Reader\_GetChunkSize, 559

    OTF2\_Reader\_GetCompression, 559

    OTF2\_Reader\_GetCreator, 559

OTF2\_Reader\_GetDefReader, 560

OTF2\_Reader\_GetDescription, 560

OTF2\_Reader\_GetEvtReader, 560

OTF2\_Reader\_GetFileSubstrate, 561

OTF2\_Reader\_GetGlobalDefReader,  
    561

OTF2\_Reader\_GetGlobalEvtReader,  
    561

OTF2\_Reader\_GetGlobalSnapReader,  
    562

OTF2\_Reader\_GetMachineName, 562

OTF2\_Reader\_GetMarkerReader, 562

OTF2\_Reader\_GetMarkerWriter, 563

OTF2\_Reader\_GetNumberOfGlobalDefinitions,  
    563

OTF2\_Reader\_GetNumberOfLocations,  
    564

OTF2\_Reader\_GetNumberOfSnapshots,  
    564

OTF2\_Reader\_GetNumberOfThumbnails,  
    564

OTF2\_ReaderGetProperty, 565

OTF2\_ReaderGetPropertyNames, 565

OTF2\_ReaderGetSnapReader, 566

OTF2\_ReaderGetThumbReader, 566

OTF2\_ReaderGetTraceId, 566

OTF2\_ReaderGetVersion, 567

OTF2\_ReaderHasGlobalEvent, 567

OTF2\_ReaderOpen, 568

OTF2\_ReaderReadAllGlobalDefinitions,  
    568

OTF2\_ReaderReadAllGlobalEvents,  
    568

OTF2\_ReaderReadAllGlobalSnapshots,  
    569

OTF2\_ReaderReadAllLocalDefinitions,  
    569

OTF2\_ReaderReadAllLocalEvents,  
    570

OTF2\_ReaderReadAllLocalSnapshots,  
    570

OTF2\_ReaderReadAllMarkers, 570

OTF2\_ReaderReadGlobalDefinitions,  
    571

## INDEX

---

OTF2\_Reader\_ReadGlobalEvent, [571](#) OTF2\_Reader\_CloseSnapReader  
OTF2\_Reader\_ReadGlobalEvents, [572](#) OTF2\_Reader.h, [557](#)  
OTF2\_Reader\_ReadGlobalSnapshots, OTF2\_Reader\_CloseThumbReader  
    [572](#) OTF2\_Reader.h, [558](#)  
OTF2\_Reader\_ReadLocalDefinitions, OTF2\_Reader\_GetBoolProperty  
    [573](#) OTF2\_Reader.h, [558](#)  
OTF2\_Reader\_ReadLocalEvents, [573](#) OTF2\_Reader\_GetChunkSize  
OTF2\_Reader\_ReadLocalEventsBackward, OTF2\_Reader.h, [559](#)  
    [574](#) OTF2\_Reader\_GetCompression  
OTF2\_Reader\_ReadLocalSnapshots, OTF2\_Reader.h, [559](#)  
    [574](#) OTF2\_Reader\_GetCreator  
OTF2\_Reader\_ReadMarkers, [575](#) OTF2\_Reader.h, [559](#)  
OTF2\_Reader\_RegisterDefCallbacks, OTF2\_Reader\_GetDefReader  
    [575](#) OTF2\_Reader.h, [560](#)  
OTF2\_Reader\_RegisterEvtCallbacks, OTF2\_Reader\_GetDescription  
    [576](#) OTF2\_Reader.h, [560](#)  
OTF2\_Reader\_RegisterGlobalDefCallbacks, OTF2\_Reader\_GetEvtReader  
    [576](#) OTF2\_Reader.h, [560](#)  
OTF2\_Reader\_RegisterGlobalEvtCallbacks, OTF2\_Reader\_GetFileSubstrate  
    [576](#) OTF2\_Reader.h, [561](#)  
OTF2\_Reader\_RegisterGlobalSnapCallbacks, OTF2\_Reader\_GetGlobalDefReader  
    [577](#) OTF2\_Reader.h, [561](#)  
OTF2\_Reader\_RegisterMarkerCallback, OTF2\_Reader\_GetGlobalEvtReader  
    [577](#) OTF2\_Reader.h, [561](#)  
OTF2\_Reader\_RegisterSnapCallback, OTF2\_Reader\_GetGlobalSnapReader  
    [578](#) OTF2\_Reader.h, [562](#)  
OTF2\_Reader\_SetFileSionCallbacks, OTF2\_Reader\_GetMachineName  
    [578](#) OTF2\_Reader.h, [562](#)  
OTF2\_Reader\_Close  
    OTF2\_Reader.h, [554](#) OTF2\_Reader\_GetMarkerReader  
OTF2\_Reader\_CloseDefReader  
    OTF2\_Reader.h, [555](#) OTF2\_Reader\_GetMarkerWriter  
OTF2\_Reader\_CloseEvtReader  
    OTF2\_Reader.h, [555](#) OTF2\_Reader\_GetNumberOfGlobalDefinitions  
OTF2\_Reader\_CloseGlobalDefReader  
    OTF2\_Reader.h, [555](#) OTF2\_Reader\_GetNumberOfLocations  
OTF2\_Reader\_CloseGlobalEvtReader  
    OTF2\_Reader.h, [556](#) OTF2\_Reader\_GetNumberOfSnapshots  
OTF2\_Reader\_CloseGlobalSnapReader  
    OTF2\_Reader.h, [556](#) OTF2\_Reader\_GetNumberOfThumbnails  
OTF2\_Reader\_CloseMarkerReader  
    OTF2\_Reader.h, [556](#) OTF2\_ReaderGetProperty  
OTF2\_Reader\_CloseMarkerWriter  
    OTF2\_Reader.h, [557](#) OTF2\_ReaderGetPropertyNames

## INDEX

---

OTF2\_Reader\_GetSnapReader  
    OTF2\_Reader.h, 566  
OTF2\_Reader\_GetThumbReader  
    OTF2\_Reader.h, 566  
OTF2\_Reader\_GetTraceId  
    OTF2\_Reader.h, 566  
OTF2\_Reader\_GetVersion  
    OTF2\_Reader.h, 567  
OTF2\_Reader\_HasGlobalEvent  
    OTF2\_Reader.h, 567  
OTF2\_Reader\_Open  
    OTF2\_Reader.h, 568  
OTF2\_Reader\_ReadAllGlobalDefinitions  
    OTF2\_Reader.h, 568  
OTF2\_Reader\_ReadAllGlobalEvents  
    OTF2\_Reader.h, 568  
OTF2\_Reader\_ReadAllGlobalSnapshots  
    OTF2\_Reader.h, 569  
OTF2\_Reader\_ReadAllLocalDefinitions  
    OTF2\_Reader.h, 569  
OTF2\_Reader\_ReadAllLocalEvents  
    OTF2\_Reader.h, 570  
OTF2\_Reader\_ReadAllLocalSnapshots  
    OTF2\_Reader.h, 570  
OTF2\_Reader\_ReadAllMarkers  
    OTF2\_Reader.h, 570  
OTF2\_Reader\_ReadGlobalDefinitions  
    OTF2\_Reader.h, 571  
OTF2\_Reader\_ReadGlobalEvent  
    OTF2\_Reader.h, 571  
OTF2\_Reader\_ReadGlobalEvents  
    OTF2\_Reader.h, 572  
OTF2\_Reader\_ReadGlobalSnapshots  
    OTF2\_Reader.h, 572  
OTF2\_Reader\_ReadLocalDefinitions  
    OTF2\_Reader.h, 573  
OTF2\_Reader\_ReadLocalEvents  
    OTF2\_Reader.h, 573  
OTF2\_Reader\_ReadLocalEventsBackward  
    OTF2\_Reader.h, 574  
OTF2\_Reader\_ReadLocalSnapshots  
    OTF2\_Reader.h, 574  
OTF2\_Reader\_ReadMarkers  
    OTF2\_Reader.h, 575  
OTF2\_Reader\_RegisterDefCallbacks  
    OTF2\_Reader.h, 575  
OTF2\_Reader\_RegisterEvtCallbacks  
    OTF2\_Reader.h, 576  
OTF2\_Reader\_RegisterGlobalDefCallbacks  
    OTF2\_Reader.h, 576  
OTF2\_Reader\_RegisterGlobalEvtCallbacks  
    OTF2\_Reader.h, 576  
OTF2\_Reader\_RegisterGlobalSnapCallbacks  
    OTF2\_Reader.h, 577  
OTF2\_Reader\_RegisterMarkerCallbacks  
    OTF2\_Reader.h, 577  
OTF2\_Reader\_RegisterSnapCallbacks  
    OTF2\_Reader.h, 578  
OTF2\_Reader\_SetFileSionCallbacks  
    OTF2\_Reader.h, 578  
OTF2RecorderKind\_enum  
    OTF2\_Definitions.h, 166  
OTF2RegionFlag\_enum  
    OTF2\_Definitions.h, 166  
OTF2RegionRole\_enum  
    OTF2\_Definitions.h, 167  
OTF2RmaAtomicType\_enum  
    OTF2\_Events.h, 229  
OTF2RmaSyncLevel\_enum  
    OTF2\_Events.h, 229  
OTF2RmaSyncType\_enum  
    OTF2\_Events.h, 229  
OTF2\_SnapReader.h, 579  
    OTF2\_SnapReader\_GetLocationID,  
        580  
    OTF2\_SnapReader\_ReadSnapshots,  
        580  
    OTF2\_SnapReader\_Seek, 581  
    OTF2\_SnapReader\_SetCallbacks, 581  
OTF2\_SnapReader\_GetLocationID  
    OTF2\_SnapReader.h, 580  
OTF2\_SnapReader\_ReadSnapshots  
    OTF2\_SnapReader.h, 580  
OTF2\_SnapReader\_Seek  
    OTF2\_SnapReader.h, 581  
OTF2\_SnapReader\_SetCallbacks  
    OTF2\_SnapReader.h, 581  
OTF2\_SnapReaderCallback\_Enter

## INDEX

---

OTF2_SnapReaderCallbacks.h, 587	OTF2_SnapReaderCallback_Enter, 587
OTF2_SnapReaderCallback_MeasurementOn	OTF2_SnapReaderCallback_MeasurementOnOff,
OTF2_SnapReaderCallbacks.h, 588	588
OTF2_SnapReaderCallback_Metric	OTF2_SnapReaderCallback_Metric,
OTF2_SnapReaderCallbacks.h, 589	589
OTF2_SnapReaderCallback_MpiCollectiveBegin	OTF2_SnapReaderCallback_MpiCollectiveBegin,
OTF2_SnapReaderCallbacks.h, 590	590
OTF2_SnapReaderCallback_MpiCollectiveEnd	OTF2_SnapReaderCallback_MpiCollectiveEnd,
OTF2_SnapReaderCallbacks.h, 590	590
OTF2_SnapReaderCallback_MpiIrecv	OTF2_SnapReaderCallback_MpiIrecv,
OTF2_SnapReaderCallbacks.h, 591	591
OTF2_SnapReaderCallback_MpiIrecvRequest	OTF2_SnapReaderCallback_MpiIrecvRequest,
OTF2_SnapReaderCallbacks.h, 592	592
OTF2_SnapReaderCallback_MpiIsend	OTF2_SnapReaderCallback_MpiIsend,
OTF2_SnapReaderCallbacks.h, 593	593
OTF2_SnapReaderCallback_MpiIsendComplete	OTF2_SnapReaderCallback_MpiIsendComplete,
OTF2_SnapReaderCallbacks.h, 594	594
OTF2_SnapReaderCallback_MpiRecv	OTF2_SnapReaderCallback_MpiRecv,
OTF2_SnapReaderCallbacks.h, 594	594
OTF2_SnapReaderCallback_MpiSend	OTF2_SnapReaderCallback_MpiSend,
OTF2_SnapReaderCallbacks.h, 595	595
OTF2_SnapReaderCallback_OmpAcquireLock	OTF2_SnapReaderCallback_OmpAcquireLock,
OTF2_SnapReaderCallbacks.h, 596	596
OTF2_SnapReaderCallback_OmpFork	OTF2_SnapReaderCallback_OmpFork,
OTF2_SnapReaderCallbacks.h, 597	597
OTF2_SnapReaderCallback_OmpTaskCreate	OTF2_SnapReaderCallback_OmpTaskCreate,
OTF2_SnapReaderCallbacks.h, 598	598
OTF2_SnapReaderCallback_OmpTaskSwitch	OTF2_SnapReaderCallback_OmpTaskSwitch,
OTF2_SnapReaderCallbacks.h, 598	598
OTF2_SnapReaderCallback_ParameterInt	OTF2_SnapReaderCallback_ParameterInt,
OTF2_SnapReaderCallbacks.h, 599	599
OTF2_SnapReaderCallback_ParameterString	OTF2_SnapReaderCallback_ParameterString,
OTF2_SnapReaderCallbacks.h, 600	600
OTF2_SnapReaderCallback_ParameterUnsignedInt	OTF2_SnapReaderCallback_ParameterUnsignedInt,
OTF2_SnapReaderCallbacks.h, 601	601
OTF2_SnapReaderCallback_SnapshotEnd	OTF2_SnapReaderCallback_SnapshotEnd,
OTF2_SnapReaderCallbacks.h, 602	602
OTF2_SnapReaderCallback_SnapshotStart	OTF2_SnapReaderCallback_SnapshotStart,
OTF2_SnapReaderCallbacks.h, 602	602
OTF2_SnapReaderCallback_Unknown	OTF2_SnapReaderCallback_Unknown,
OTF2_SnapReaderCallbacks.h, 603	603
OTF2_SnapReaderCallbacks	OTF2_SnapReaderCallbacks, 603
OTF2_SnapReaderCallbacks.h, 603	
OTF2_SnapReaderCallbacks.h, 582	OTF2_SnapReaderCallbacks_Clear, 604

## INDEX

---

OTF2\_SnapReaderCallbacks\_Delete, OTF2\_SnapReaderCallbacks\_SetUnknownCallback,  
604 616  
OTF2\_SnapReaderCallbacks\_New, OTF2\_SnapReaderCallbacks\_Clear  
604 OTF2\_SnapReaderCallbacks.h, 604  
OTF2\_SnapReaderCallbacks\_SetEnterCallback, OTF2\_SnapReaderCallbacks\_Delete  
604 OTF2\_SnapReaderCallbacks.h, 604  
OTF2\_SnapReaderCallbacks\_SetMeasurementOnOffCallback, OTF2\_SnapReaderCallbacks\_New  
605 OTF2\_SnapReaderCallbacks.h, 604  
OTF2\_SnapReaderCallbacks\_SetMetric, OTF2\_SnapReaderCallbacks\_SetEnterCallback  
606 OTF2\_SnapReaderCallbacks.h, 604  
OTF2\_SnapReaderCallbacks\_SetMpiOTF2\_SnapReaderCallbacks\_SetMeasurementOnOffCallback  
606 OTF2\_SnapReaderCallbacks.h, 605  
OTF2\_SnapReaderCallbacks\_SetMpiOTF2\_SnapReaderCallbacks\_SetMetricCallback  
607 OTF2\_SnapReaderCallbacks.h, 606  
OTF2\_SnapReaderCallbacks\_SetMpiOTF2\_SnapReaderCallbacks\_SetMpiCollectiveBeginCallback  
607 OTF2\_SnapReaderCallbacks.h, 606  
OTF2\_SnapReaderCallbacks\_SetMpiOTF2\_SnapReaderCallbacks\_SetMpiCollectiveEndCallback  
608 OTF2\_SnapReaderCallbacks.h, 607  
OTF2\_SnapReaderCallbacks\_SetMpiOTF2\_SnapReaderCallbacks\_SetMpiIrecvCallback  
608 OTF2\_SnapReaderCallbacks.h, 607  
OTF2\_SnapReaderCallbacks\_SetMpiOTF2\_SnapReaderCallbacks\_SetMpiIrecvRequestCallback  
609 OTF2\_SnapReaderCallbacks.h, 608  
OTF2\_SnapReaderCallbacks\_SetMpiOTF2\_SnapReaderCallbacks\_SetMpiIsendCallback  
610 OTF2\_SnapReaderCallbacks.h, 608  
OTF2\_SnapReaderCallbacks\_SetMpiOTF2\_SnapReaderCallbacks\_SetMpiIsendCompleteCallback  
610 OTF2\_SnapReaderCallbacks.h, 609  
OTF2\_SnapReaderCallbacks\_SetOmpOTF2\_SnapReaderCallbacks\_SetMpiRecvCallback  
611 OTF2\_SnapReaderCallbacks.h, 610  
OTF2\_SnapReaderCallbacks\_SetOmpOTF2\_SnapReaderCallbacks\_SetOmpAcquireLockCallback  
612 OTF2\_SnapReaderCallbacks.h, 611  
OTF2\_SnapReaderCallbacks\_SetOmpOTF2\_SnapReaderCallbacks\_SetOmpForkCallback  
612 OTF2\_SnapReaderCallbacks.h, 611  
OTF2\_SnapReaderCallbacks\_SetParameterOTF2\_SnapReaderCallbacks\_SetOmpTaskCreateCallback  
613 OTF2\_SnapReaderCallbacks.h, 612  
OTF2\_SnapReaderCallbacks\_SetParameterOTF2\_SnapReaderCallbacks\_SetOmpTaskSwitchCallback  
613 OTF2\_SnapReaderCallbacks.h, 612  
OTF2\_SnapReaderCallbacks\_SetParameterOTF2\_SnapReaderCallbacks\_SetParameterIntCallback  
614 OTF2\_SnapReaderCallbacks.h, 613  
OTF2\_SnapReaderCallbacks\_SetSnapshotOTF2\_SnapReaderCallbacks\_SetParameterStringCallback  
615 OTF2\_SnapReaderCallbacks.h, 613  
OTF2\_SnapReaderCallbacks\_SetSnapshotOTF2\_SnapReaderCallbacks\_SetParameterUnsignedIntCallback  
615 OTF2\_SnapReaderCallbacks.h, 614

## INDEX

---

OTF2\_SnapReaderCallbacks\_SetSnapshot  
OTF2\_SnapWriter\_GetLocationID  
OTF2\_SnapReaderCallbacks.h, 615      OTF2\_SnapWriter.h, 620

OTF2\_SnapReaderCallbacks\_SetSnapshot  
OTF2\_SnapWriter\_MeasurementOnOff  
OTF2\_SnapReaderCallbacks.h, 615      OTF2\_SnapWriter.h, 621

OTF2\_SnapReaderCallbacks\_SetUnknown  
OTF2\_SnapWriter\_Metric  
OTF2\_SnapReaderCallbacks.h, 616      OTF2\_SnapWriter.h, 621

OTF2\_SnapWriter  
OTF2\_SnapWriter.h, 619      OTF2\_SnapWriter\_MpiCollectiveBegin  
OTF2\_SnapWriter.h, 622

OTF2\_SnapWriter.h, 616      OTF2\_SnapWriter\_MpiCollectiveEnd  
OTF2\_SnapWriter, 619      OTF2\_SnapWriter.h, 623

OTF2\_SnapWriter\_Enter, 620      OTF2\_SnapWriter\_MpiIrecv  
OTF2\_SnapWriter\_GetLocationID,      OTF2\_SnapWriter.h, 623  
620      OTF2\_SnapWriter\_MpiIrecvRequest

OTF2\_SnapWriter\_MeasurementOnOff,      OTF2\_SnapWriter.h, 624  
621      OTF2\_SnapWriter\_MpiIsend

OTF2\_SnapWriter\_Metric, 621      OTF2\_SnapWriter.h, 625

OTF2\_SnapWriter\_MpiCollectiveBegin  
OTF2\_SnapWriter\_MpiIsendComplete  
622      OTF2\_SnapWriter.h, 626

OTF2\_SnapWriter\_MpiCollectiveEnd  
OTF2\_SnapWriter\_MpiRecv  
623      OTF2\_SnapWriter.h, 626

OTF2\_SnapWriter\_MpiIrecv, 623      OTF2\_SnapWriter\_MpiSend

OTF2\_SnapWriter\_MpiIrecvRequest,      OTF2\_SnapWriter.h, 627  
624      OTF2\_SnapWriter\_OmpAcquireLock

OTF2\_SnapWriter\_MpiIsend, 625      OTF2\_SnapWriter.h, 628

OTF2\_SnapWriter\_MpiIsendComplete  
OTF2\_SnapWriter\_OmpFork  
626      OTF2\_SnapWriter.h, 629

OTF2\_SnapWriter\_MpiRecv, 626      OTF2\_SnapWriter\_OmpTaskCreate

OTF2\_SnapWriter\_MpiSend, 627      OTF2\_SnapWriter.h, 629

OTF2\_SnapWriter\_OmpAcquireLock  
OTF2\_SnapWriter\_OmpTaskSwitch  
628      OTF2\_SnapWriter.h, 630

OTF2\_SnapWriter\_OmpFork, 629      OTF2\_SnapWriter\_ParameterInt

OTF2\_SnapWriter\_OmpTaskCreate,      OTF2\_SnapWriter.h, 631  
629      OTF2\_SnapWriter\_ParameterString

OTF2\_SnapWriter\_OmpTaskSwitch,      OTF2\_SnapWriter.h, 631  
630      OTF2\_SnapWriter\_ParameterUnsignedInt

OTF2\_SnapWriter\_ParameterInt, 631      OTF2\_SnapWriter.h, 632

OTF2\_SnapWriter\_ParameterString,      OTF2\_SnapWriter\_SnapshotEnd  
631      OTF2\_SnapWriter.h, 633

OTF2\_SnapWriter\_ParameterUnsigned  
OTF2\_SnapWriter\_SnapshotStart  
632      OTF2\_SnapWriter.h, 633

OTF2\_SnapWriter\_SnapshotEnd, 633  
OTF2\_SnapWriter\_SnapshotStart, 633      OTF2\_SystemTreeDomain\_enum

OTF2\_SnapWriter\_Enter      OTF2\_Definitions.h, 168

OTF2\_SnapWriter      OTF2\_Thumbnail.h, 634

OTF2\_SnapWriter.h, 620      OTF2\_ThumbReader\_GetHeader, 635

## INDEX

---

OTF2\_ThumbReader\_ReadSample,  
    [636](#)  
OTF2\_ThumbWriter\_WriteSample,  
    [636](#)  
OTF2\_ThumbnailType\_enum  
    OTF2\_GeneralDefinitions.h, [366](#)  
OTF2\_ThumbReader\_GetHeader  
    OTF2\_Thumbnail.h, [635](#)  
OTF2\_ThumbReader\_ReadSample  
    OTF2\_Thumbnail.h, [636](#)  
OTF2\_ThumbWriter\_WriteSample  
    OTF2\_Thumbnail.h, [636](#)  
OTF2\_Type\_enum  
    OTF2\_GeneralDefinitions.h, [367](#)  
OTF2\_UNDEFINED\_TYPE  
    OTF2\_GeneralDefinitions.h, [363](#)