

CoT
0.0.0

Generated by Doxygen 1.5.8

Wed Sep 16 15:00:11 2009

Contents

1 C oBIX Tools Full Reference Manual	1
2 Todo List	2
3 oBIX Client Library Overview	4
4 Data Structure Index	4
4.1 Data Structures	4
5 File Index	5
5.1 File List	5
6 Data Structure Documentation	7
6.1 _Comm_Stack Struct Reference	7
6.1.1 Detailed Description	8
6.1.2 Field Documentation	8
6.2 _Connection Struct Reference	10
6.2.1 Detailed Description	11
6.2.2 Field Documentation	11
6.3 _CURL_EXT Struct Reference	12
6.3.1 Detailed Description	12
6.3.2 Field Documentation	13
6.4 _Device Struct Reference	14
6.4.1 Detailed Description	15
6.4.2 Field Documentation	15
6.5 _Http_Connection Struct Reference	16
6.5.1 Detailed Description	18
6.5.2 Field Documentation	18
6.6 _Http_Device Struct Reference	21
6.6.1 Detailed Description	22
6.6.2 Field Documentation	22
6.7 _Listener Struct Reference	22
6.7.1 Detailed Description	22
6.7.2 Field Documentation	23
6.8 _oBIX_Batch Struct Reference	23
6.8.1 Detailed Description	25
6.8.2 Field Documentation	25

6.9	_oBIX_BatchCmd Struct Reference	26
6.9.1	Detailed Description	27
6.9.2	Field Documentation	27
6.10	_oBIX_BatchResult Struct Reference	28
6.10.1	Detailed Description	28
6.10.2	Field Documentation	28
6.11	_Periodic_Task Struct Reference	29
6.11.1	Detailed Description	30
6.11.2	Field Documentation	30
6.12	_Request Struct Reference	32
6.12.1	Detailed Description	32
6.12.2	Field Documentation	32
6.13	_Table Struct Reference	33
6.13.1	Detailed Description	33
6.13.2	Field Documentation	33
6.14	_Target Struct Reference	34
6.14.1	Detailed Description	34
6.14.2	Field Documentation	35
6.15	_Task_Thread Struct Reference	37
6.15.1	Detailed Description	38
6.15.2	Field Documentation	38
6.16	_TreeNode Struct Reference	39
6.16.1	Detailed Description	39
6.16.2	Field Documentation	39
6.17	oBIX_Watch Struct Reference	40
6.17.1	Detailed Description	41
6.17.2	Field Documentation	41
6.18	oBIX_Watch_Item Struct Reference	43
6.18.1	Detailed Description	44
6.18.2	Field Documentation	44
6.19	PollTaskParams Struct Reference	45
6.19.1	Detailed Description	46
6.19.2	Field Documentation	46
6.20	Response Struct Reference	46
6.20.1	Detailed Description	47
6.20.2	Field Documentation	48

7 File Documentation	49
7.1 adapters/example_timer.c File Reference	49
7.1.1 Detailed Description	50
7.1.2 Define Documentation	50
7.1.3 Function Documentation	51
7.1.4 Variable Documentation	52
7.2 adapters/sensor_floor_driver.c File Reference	53
7.2.1 Detailed Description	56
7.2.2 Define Documentation	56
7.2.3 Typedef Documentation	57
7.2.4 Function Documentation	58
7.2.5 Variable Documentation	63
7.3 client/curl_ext.c File Reference	64
7.3.1 Detailed Description	65
7.3.2 Define Documentation	66
7.3.3 Function Documentation	66
7.3.4 Variable Documentation	69
7.4 client/curl_ext.h File Reference	70
7.4.1 Detailed Description	71
7.4.2 Typedef Documentation	71
7.4.3 Function Documentation	71
7.5 client/obix_batch.c File Reference	73
7.5.1 Detailed Description	74
7.5.2 Define Documentation	75
7.5.3 Function Documentation	75
7.6 client/obix_batch.h File Reference	80
7.6.1 Detailed Description	81
7.6.2 Typedef Documentation	81
7.6.3 Enumeration Type Documentation	81
7.7 client/obix_client.c File Reference	81
7.7.1 Detailed Description	83
7.7.2 Define Documentation	84
7.7.3 Function Documentation	84
7.7.4 Variable Documentation	94
7.8 client/obix_client.h File Reference	95
7.8.1 Detailed Description	97

7.8.2	Usage	97
7.8.3	Typedef Documentation	98
7.8.4	Enumeration Type Documentation	99
7.8.5	Function Documentation	101
7.9	client/obix_comm.h File Reference	112
7.9.1	Detailed Description	113
7.9.2	Typedef Documentation	113
7.9.3	Enumeration Type Documentation	115
7.9.4	Function Documentation	116
7.10	client/obix_http.c File Reference	116
7.10.1	Detailed Description	119
7.10.2	Define Documentation	119
7.10.3	Function Documentation	122
7.10.4	Variable Documentation	131
7.11	client/obix_http.h File Reference	133
7.11.1	Detailed Description	135
7.11.2	Typedef Documentation	135
7.11.3	Function Documentation	135
7.11.4	Variable Documentation	138
7.12	client/table.c File Reference	139
7.12.1	Detailed Description	139
7.12.2	Function Documentation	139
7.13	client/table.h File Reference	141
7.13.1	Detailed Description	141
7.13.2	Typedef Documentation	142
7.13.3	Function Documentation	142
7.14	common/bool.h File Reference	143
7.14.1	Detailed Description	144
7.14.2	Define Documentation	144
7.14.3	Typedef Documentation	145
7.15	common/ixml_ext.c File Reference	145
7.15.1	Detailed Description	147
7.15.2	Function Documentation	147
7.16	common/ixml_ext.h File Reference	153
7.16.1	Detailed Description	155
7.16.2	Define Documentation	155

7.16.3 Function Documentation	155
7.17 common/log_utils.c File Reference	162
7.17.1 Detailed Description	163
7.17.2 Function Documentation	163
7.17.3 Variable Documentation	165
7.18 common/log_utils.h File Reference	166
7.18.1 Detailed Description	168
7.18.2 Define Documentation	168
7.18.3 Typedef Documentation	170
7.18.4 Enumeration Type Documentation	170
7.18.5 Function Documentation	170
7.18.6 Variable Documentation	171
7.19 common/obix_utils.c File Reference	172
7.19.1 Detailed Description	175
7.19.2 Function Documentation	175
7.19.3 Variable Documentation	177
7.20 common/obix_utils.h File Reference	184
7.20.1 Detailed Description	187
7.20.2 Enumeration Type Documentation	187
7.20.3 Function Documentation	188
7.20.4 Variable Documentation	189
7.21 common/ptask.c File Reference	196
7.21.1 Detailed Description	198
7.21.2 Typedef Documentation	198
7.21.3 Function Documentation	198
7.22 common/ptask.h File Reference	205
7.22.1 Detailed Description	206
7.22.2 Usage	206
7.22.3 Define Documentation	207
7.22.4 Typedef Documentation	207
7.22.5 Function Documentation	207
7.23 common/xml_config.c File Reference	211
7.23.1 Detailed Description	213
7.23.2 Function Documentation	213
7.23.3 Variable Documentation	218
7.24 common/xml_config.h File Reference	221

7.24.1	Detailed Description	223
7.24.2	Function Documentation	223
7.24.3	Variable Documentation	228
7.25	doxygen/cot_main.h File Reference	231
7.25.1	Detailed Description	231
7.26	doxygen/libcot_desc.h File Reference	231
7.26.1	Detailed Description	231
7.27	server/doctree.c File Reference	231
7.27.1	Detailed Description	232
7.27.2	Typedef Documentation	232
7.27.3	Function Documentation	232
7.27.4	Variable Documentation	233
7.28	server/doctree.h File Reference	233
7.28.1	Detailed Description	233
7.28.2	Function Documentation	234
7.29	server/obix_fcgi.c File Reference	234
7.29.1	Define Documentation	236
7.29.2	Function Documentation	236
7.29.3	Variable Documentation	239
7.30	server/obix_fcgi.h File Reference	241
7.30.1	Detailed Description	242
7.30.2	Function Documentation	242
7.31	server/post_handler.c File Reference	244
7.31.1	Detailed Description	245
7.31.2	Define Documentation	246
7.31.3	Function Documentation	246
7.31.4	Variable Documentation	251
7.32	server/post_handler.h File Reference	251
7.32.1	Detailed Description	252
7.32.2	Typedef Documentation	252
7.32.3	Function Documentation	253
7.33	server/request.h File Reference	253
7.33.1	Typedef Documentation	253
7.34	server/response.c File Reference	253
7.34.1	Detailed Description	254
7.34.2	Function Documentation	255

7.34.3 Variable Documentation	257
7.35 server/response.h File Reference	257
7.35.1 Detailed Description	258
7.35.2 Typedef Documentation	258
7.35.3 Function Documentation	258
7.36 server/server.c File Reference	261
7.36.1 Detailed Description	262
7.36.2 Define Documentation	262
7.36.3 Function Documentation	262
7.36.4 Variable Documentation	266
7.37 server/server.h File Reference	266
7.37.1 Detailed Description	267
7.37.2 Function Documentation	267
7.38 server/watch.c File Reference	269
7.38.1 Detailed Description	271
7.38.2 Function Documentation	271
7.38.3 Variable Documentation	278
7.39 server/watch.h File Reference	280
7.39.1 Detailed Description	281
7.39.2 Typedef Documentation	281
7.39.3 Function Documentation	282
7.39.4 Variable Documentation	286
7.40 server/xml_storage.c File Reference	287
7.40.1 Detailed Description	289
7.40.2 Function Documentation	289
7.40.3 Variable Documentation	298
7.41 server/xml_storage.h File Reference	300
7.41.1 Detailed Description	301
7.41.2 Function Documentation	301
7.41.3 Variable Documentation	308
7.42 test/test_client.c File Reference	308
7.42.1 Detailed Description	309
7.42.2 Define Documentation	309
7.42.3 Function Documentation	310
7.43 test/test_client.h File Reference	311
7.43.1 Detailed Description	311

7.43.2 Function Documentation	312
7.44 test/test_common.c File Reference	312
7.44.1 Detailed Description	313
7.44.2 Function Documentation	313
7.45 test/test_common.h File Reference	314
7.45.1 Detailed Description	315
7.45.2 Function Documentation	315
7.46 test/test_main.c File Reference	315
7.46.1 Function Documentation	316
7.47 test/test_main.h File Reference	317
7.47.1 Function Documentation	317
7.48 test/test_ptask.c File Reference	317
7.48.1 Function Documentation	318
7.49 test/test_ptask.h File Reference	320
7.49.1 Detailed Description	320
7.49.2 Function Documentation	320
7.50 test/test_server.c File Reference	321
7.50.1 Detailed Description	322
7.50.2 Function Documentation	322
7.50.3 Variable Documentation	326
7.51 test/test_server.h File Reference	327
7.51.1 Detailed Description	327
7.51.2 Function Documentation	327
7.52 test/test_table.c File Reference	327
7.52.1 Detailed Description	328
7.52.2 Function Documentation	328
7.53 test/test_table.h File Reference	329
7.53.1 Detailed Description	329
7.53.2 Function Documentation	329

1 C oBIX Tools Full Reference Manual

Todo

add description here

2 Todo List

page **C oBIX Tools Full Reference Manual** add description here

Global **LAST_EVENT_POLL_PERIOD** Check whether we can monitor it by including to the Watch.

Global **checkEventsTask** it is created because Sensor Floor's event feed doesn't show the Fallen event

File **curl_ext.c** add description here

File **curl_ext.h** add description here

File **obix_batch.c** add description here

File **obix_batch.h** add description here

File **obix_client.c** add description here

Global **OBIX_ERR_LIMIT_REACHED** Remove this error and enlarge arrays when needed.

File **obix_comm.h** add description here

File **obix_http.c** add description here

File **obix_http.h** add description here

File **table.c** It supposed to be a hashtable, but now it is quite slow.

File **table.h** add description here

File **ptask.c** Add description

File **xml_config.c** add description

File **doctree.c** add description here

File doctree.h add description here

File obix_fcgi.h add description

File post_handler.h add description here

Global obix_server_postHandler describe me

File response.c add description here

File response.h add description here

File server.c add description there

Global updateMetaWatch think about the whole conception of meta attributes (how to unify its usage)

File watch.c add description

Global obixWatch_create Move error codes to the global error enum.

Global obixWatch_createWatchItem describe me; rename to createWatchItem

File watch.h add description here

Global checkNode Check that all objects in the input document have required attributes and correct URI's.

File test_client.c add description here

File test_client.h add description here

File test_common.c add description here

File test_common.h add description here

File test_ptask.h add description here

File test_server.h add description here

File test_table.c add description here

File test_table.h add description here

3 oBIX Client Library Overview

oBIX Client Library (libcot) is a small library written in C, which allows creating lightweight oBIX client applications such as device adapters.

Library API consists of [obix_client.h](#) which declares oBIX Client API and several utility header files which provides helper tools commonly needed by oBIX client applications:

- [ptask.h](#) - Allows scheduling tasks to be executed in a separate thread;
- [obix_utils.h](#) - Contains names of most essential oBIX object, attributes, etc.
- [xml_config.h](#) - Helps to load application settings from XML file;
- [log_utils.h](#) - Provides interface to the logging utility which is used by the whole library.
- [ixml_ext.h](#) - Contains utilities for manipulating XML DOM structures.

All parts of the library produce log messages which give better understanding of what happens inside and help to investigate occurred problems. By default the logging system writes everything (including all debug messages) to `stdout` which can be inconvenient. Log settings can be changed either manually (using functions defined at [log_utils.h](#)) or during oBIX Client API initialization (using [obix_loadConfigFile\(\)](#)).

The example usage of the library can be found at [example_timer.c](#)

Compilation

The following string will compile application which uses oBIX Client Library:

```
gcc -I<cot_headers> -L<cot_lib> -lcot-client <source> -o <output_name>
```

where

- `<cot_headers>` - Path to header files of *libcot* (usually it is `<installation_prefix>/include/cot/`).
- `<cot_lib>` - Path to library binaries of libcot (usually it is `<installation_prefix>/lib`).
- `<sources>` - Your source files to be compiled.
- `<output_name>` - Name of the output binary.

Author:

Andrey Litvinov

4 Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

[_Comm_Stack](#)

7

_Connection	10
_CURL_EXT	12
_Device	14
_Http_Connection	16
_Http_Device	21
_Listener	22
_oBIX_Batch	23
_oBIX_BatchCmd	26
_oBIX_BatchResult (Contains outputs of the command, which was executed in a Batch)	28
_Periodic_Task	29
_Request	32
_Table	33
_Target (Target object which describes person's position on the sensor floor)	34
_Task_Thread	37
_TreeNode	39
oBIX_Watch (Represents an oBIX Watch object)	40
oBIX_Watch_Item (Represents a separate watch item)	43
PollTaskParams	45
Response	46

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

adapters/example_timer.c (Simple oBIX Timer device implementation, which demonstrates usage of oBIX client library)	49
adapters/sensor_floor_driver.c (This is an adapter for the Elsi Sensor Floor)	53
client/curl_ext.c	64
client/curl_ext.h	70
client/obix_batch.c	73
client/obix_batch.h	80

client/obix_client.c	81
client/obix_client.h (OBIX Client API)	95
client/obix_comm.h	112
client/obix_http.c	116
client/obix_http.h	133
client/table.c	139
client/table.h	141
common/bool.h (Defines boolean data type)	143
common/xml_ext.c (Implements utility methods for work with XML DOM structure)	145
common/xml_ext.h (Defines utility methods for work with XML DOM structure)	153
common/log_utils.c (Contains implementation of logging tools)	162
common/log_utils.h (Definitions of logging tools)	166
common/obix_utils.c (Contains names of oBIX objects, contracts, facets, etc)	172
common/obix_utils.h (Contains oBIX keywords (object names, contracts, facets, etc) and some utility functions)	184
common/ptask.c	196
common/ptask.h (Periodic Task - tool for asynchronous task execution)	205
common/xml_config.c	211
common/xml_config.h (Declares configuration API)	221
doxygen/cot_main.h (Doxygen documentation source file)	231
doxygen/libcot_desc.h (Doxygen documentation source file)	231
server/doctree.c	231
server/doctree.h	233
server/obix_fcgi.c	234
server/obix_fcgi.h	241
server/post_handler.c (Contains handlers for various oBIX invoke commands)	244
server/post_handler.h	251
server/request.h	253
server/response.c	253
server/response.h	257

server/server.c	261
server/server.h (TODO add description there)	266
server/watch.c	269
server/watch.h	280
server/xml_storage.c (Simple implementation of XML storage)	287
server/xml_storage.h (Defines the interface to the XML storage)	300
test/test_client.c	308
test/test_client.h	311
test/test_common.c	312
test/test_common.h	314
test/test_main.c	315
test/test_main.h	317
test/test_ptask.c	317
test/test_ptask.h	320
test/test_server.c (That's a temporary file to test various pieces of functionality)	321
test/test_server.h	327
test/test_table.c	327
test/test_table.h	329

6 Data Structure Documentation

6.1 _Comm_Stack Struct Reference

```
#include <obix_comm.h>
```

Data Fields

- `comm_closeConnection` `closeConnection`
- `comm_freeConnection` `freeConnection`
- `comm_initConnection` `initConnection`
- `comm_openConnection` `openConnection`
- `comm_read` `read`
- `comm_readValue` `readValue`
- `comm_registerDevice` `registerDevice`
- `comm_registerListener` `registerListener`
- `comm_sendBatch` `sendBatch`

- [comm_unregisterDevice](#) `unregisterDevice`
- [comm_unregisterListener](#) `unregisterListener`
- [comm_writeValue](#) `writeValue`

6.1.1 Detailed Description

Definition at line 74 of file obix_comm.h.

6.1.2 Field Documentation

6.1.2.1 [comm_closeConnection](#) `_Comm_Stack::closeConnection`

Definition at line 78 of file obix_comm.h.

Referenced by `obix_closeConnection()`.

6.1.2.2 [comm_freeConnection](#) `_Comm_Stack::freeConnection`

Definition at line 79 of file obix_comm.h.

Referenced by `connection_free()`.

6.1.2.3 [comm_initConnection](#) `_Comm_Stack::initConnection`

Definition at line 76 of file obix_comm.h.

Referenced by `connection_create()`.

6.1.2.4 [comm_openConnection](#) `_Comm_Stack::openConnection`

Definition at line 77 of file obix_comm.h.

Referenced by `obix_openConnection()`.

6.1.2.5 [comm_read](#) `_Comm_Stack::read`

Definition at line 84 of file obix_comm.h.

Referenced by `obix_read()`.

6.1.2.6 [comm_readValue](#) `_Comm_Stack::readValue`

Definition at line 85 of file obix_comm.h.

Referenced by obix_readValue().

6.1.2.7 comm_registerDevice _Comm_Stack::registerDevice

Definition at line 80 of file obix_comm.h.

Referenced by device_register().

6.1.2.8 comm_registerListener _Comm_Stack::registerListener

Definition at line 82 of file obix_comm.h.

Referenced by listener_register().

6.1.2.9 comm_sendBatch _Comm_Stack::sendBatch

Definition at line 87 of file obix_comm.h.

Referenced by obix_batch_send().

6.1.2.10 comm_unregisterDevice _Comm_Stack::unregisterDevice

Definition at line 81 of file obix_comm.h.

Referenced by device_unregister().

6.1.2.11 comm_unregisterListener _Comm_Stack::unregisterListener

Definition at line 83 of file obix_comm.h.

Referenced by listener_unregister().

6.1.2.12 comm_writeValue _Comm_Stack::writeValue

Definition at line 86 of file obix_comm.h.

Referenced by obix_writeValue().

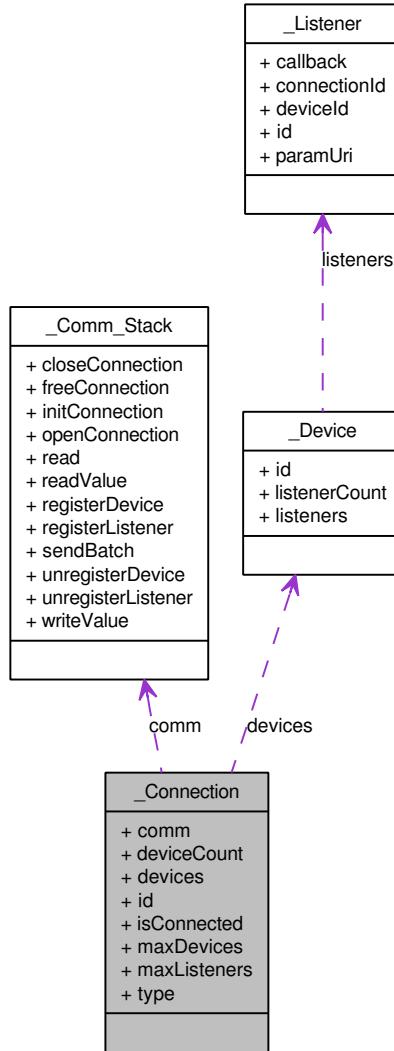
The documentation for this struct was generated from the following file:

- client/[obix_comm.h](#)

6.2 _Connection Struct Reference

```
#include <obix_comm.h>
```

Collaboration diagram for _Connection:



Data Fields

- const `Comm_Stack` * `comm`
common connection settings
- int `deviceCount`
- `Device` ** `devices`
list of devices registered using this connection
- int `id`
- `BOOL` `isConnected`
- int `maxDevices`

- int maxListeners
- [Connection_Type](#) type

6.2.1 Detailed Description

Definition at line 107 of file obix_comm.h.

6.2.2 Field Documentation

6.2.2.1 const Comm_Stack* _Connection::comm

common connection settings

Definition at line 110 of file obix_comm.h.

Referenced by `connection_create()`, `connection_free()`, `device_register()`, `device_unregister()`, `listener_register()`, `listener_unregister()`, `obix_batch_send()`, `obix_closeConnection()`, `obix_openConnection()`, `obix_read()`, `obix_readValue()`, and `obix_writeValue()`.

6.2.2.2 int _Connection::deviceCount

Definition at line 120 of file obix_comm.h.

Referenced by `connection_create()`, `device_register()`, `device_unregister()`, `obix_closeConnection()`, and `obix_registerDevice()`.

6.2.2.3 Device** _Connection::devices

list of devices registered using this connection

Definition at line 119 of file obix_comm.h.

Referenced by `connection_create()`, `connection_free()`, `device_get()`, `device_register()`, `device_unregister()`, `device_unregisterAllListeners()`, `obix_closeConnection()`, and `obix_registerDevice()`.

6.2.2.4 int _Connection::id

Definition at line 116 of file obix_comm.h.

Referenced by `connection_create()`, and `listener_register()`.

6.2.2.5 BOOL _Connection::isConnected

Definition at line 112 of file obix_comm.h.

Referenced by connection_create(), connection_free(), connection_get(), obix_closeConnection(), and obix_openConnection().

6.2.2.6 int _Connection::maxDevices

Definition at line 113 of file obix_comm.h.

Referenced by connection_create(), device_get(), obix_closeConnection(), and obix_registerDevice().

6.2.2.7 int _Connection::maxListeners

Definition at line 114 of file obix_comm.h.

Referenced by connection_create(), device_register(), device_unregisterAllListeners(), obix_registerListener(), and obix_unregisterListener().

6.2.2.8 Connection_Type _Connection::type

Definition at line 111 of file obix_comm.h.

Referenced by connection_create().

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

- client/[obix_comm.h](#)

6.3 _CURL_EXT Struct Reference

```
#include <curl_ext.h>
```

Data Fields

- CURL * [curl](#)
- char * [errorBuffer](#)
- char * [inputBuffer](#)
- int [inputBufferFree](#)
- int [inputBufferSize](#)
- const char * [outputBuffer](#)
- int [outputPos](#)
- int [outputSize](#)

6.3.1 Detailed Description

Definition at line 35 of file curl_ext.h.

6.3.2 Field Documentation

6.3.2.1 CURL* _CURL_EXT::curl

Definition at line 37 of file curl_ext.h.

Referenced by curl_ext_allocateMemory(), curl_ext_create(), curl_ext_freeMemory(), curl_ext_get(), curl_ext_post(), curl_ext_put(), and sendRequest().

6.3.2.2 char* _CURL_EXT::errorBuffer

Definition at line 50 of file curl_ext.h.

Referenced by curl_ext_allocateMemory(), curl_ext_create(), curl_ext_freeMemory(), sendRequest(), and testCurlExtRequest().

6.3.2.3 char* _CURL_EXT::inputBuffer

Definition at line 40 of file curl_ext.h.

Referenced by createWatch(), curl_ext_allocateMemory(), curl_ext_freeMemory(), http_registerDevice(), inputWriter(), parseXmlInput(), sendRequest(), setWatchTimeParam(), testCurlExt(), testCurlExtRequest(), and writeValue().

6.3.2.4 int _CURL_EXT::inputBufferFree

Definition at line 43 of file curl_ext.h.

Referenced by curl_ext_allocateMemory(), inputWriter(), and sendRequest().

6.3.2.5 int _CURL_EXT::inputBufferSize

Definition at line 42 of file curl_ext.h.

Referenced by curl_ext_allocateMemory(), inputWriter(), and sendRequest().

6.3.2.6 const char* _CURL_EXT::outputBuffer

Definition at line 45 of file curl_ext.h.

Referenced by addWatchItem(), createWatch(), curl_ext_allocateMemory(), curl_ext_post(), curl_ext_put(), http_registerDevice(), http_sendBatch(), http_unregisterListener(), outputReader(), removeWatch(), testCurlExt(), watchPollTask(), and writeValue().

6.3.2.7 int _CURL_EXT::outputPos

Definition at line 48 of file curl_ext.h.

Referenced by curl_ext_allocateMemory(), curl_ext_post(), curl_ext_put(), and outputReader().

6.3.2.8 int _CURL_EXT::outputSize

Definition at line 47 of file curl_ext.h.

Referenced by curl_ext_allocateMemory(), curl_ext_post(), curl_ext_put(), and outputReader().

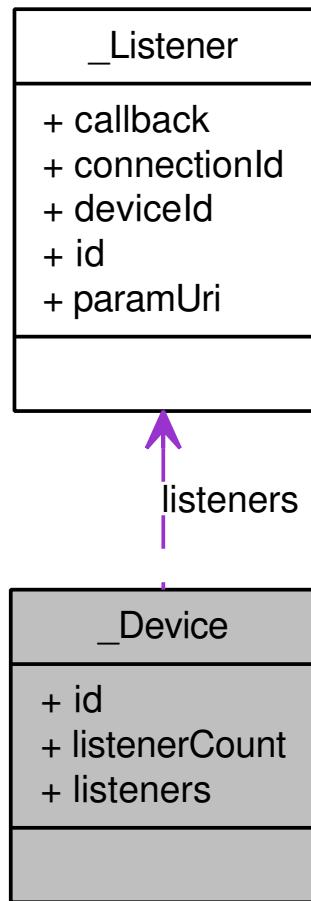
The documentation for this struct was generated from the following file:

- client/[curl_ext.h](#)

6.4 _Device Struct Reference

```
#include <obix_comm.h>
```

Collaboration diagram for _Device:



Data Fields

- int `id`
- int `listenerCount`
- `Listener ** listeners`

6.4.1 Detailed Description

Definition at line 100 of file obix_comm.h.

6.4.2 Field Documentation

6.4.2.1 int _Device::id

Definition at line 102 of file obix_comm.h.

Referenced by `device_register()`, `listener_register()`, and `listener_unregister()`.

6.4.2.2 int _Device::listenerCount

Definition at line 104 of file obix_comm.h.

Referenced by device_register(), device_unregisterAllListeners(), listener_register(), listener_unregister(), and obix_registerListener().

6.4.2.3 Listener _Device::listeners**

Definition at line 103 of file obix_comm.h.

Referenced by device_free(), device_register(), device_unregisterAllListeners(), listener_register(), listener_unregister(), obix_registerListener(), and obix_unregisterListener().

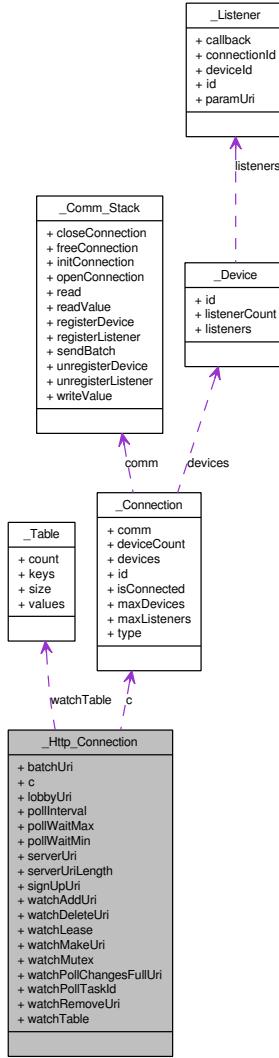
The documentation for this struct was generated from the following file:

- client/[obix_comm.h](#)

6.5 _Http_Connection Struct Reference

```
#include <obix_http.h>
```

Collaboration diagram for _Http_Connection:



Data Fields

- `char * batchUri`
- `Connection c`
- `char * lobbyUri`
- `long pollInterval`
- `long pollWaitMax`
- `long pollWaitMin`
- `char * serverUri`
- `int serverUriLength`
- `char * signUpUri`
- `char * watchAddUri`
- `char * watchDeleteUri`
- `long watchLease`
- `char * watchMakeUri`

- pthread_mutex_t [watchMutex](#)
- char * [watchPollChangesFullUri](#)
- int [watchPollTaskId](#)
- char * [watchRemoveUri](#)
- [Table](#) * [watchTable](#)

6.5.1 Detailed Description

Definition at line 36 of file obix_http.h.

6.5.2 Field Documentation

6.5.2.1 `char* _Http_Connection::batchUri`

Definition at line 49 of file obix_http.h.

Referenced by `http_freeConnection()`, `http_initConnection()`, `http_openConnection()`, and `http_sendBatch()`.

6.5.2.2 `Connection _Http_Connection::c`

Definition at line 38 of file obix_http.h.

Referenced by `http_initConnection()`.

6.5.2.3 `char* _Http_Connection::lobbyUri`

Definition at line 42 of file obix_http.h.

Referenced by `http_freeConnection()`, `http_initConnection()`, and `http_openConnection()`.

6.5.2.4 `long _Http_Connection::pollInterval`

Definition at line 43 of file obix_http.h.

Referenced by `addListener()`, and `http_initConnection()`.

6.5.2.5 `long _Http_Connection::pollWaitMax`

Definition at line 46 of file obix_http.h.

Referenced by `addListener()`, `http_initConnection()`, and `setWatchPollWaitTime()`.

6.5.2.6 long _Http_Connection::pollWaitMin

Definition at line 45 of file obix_http.h.

Referenced by http_initConnection(), and setWatchPollWaitTime().

6.5.2.7 char* _Http_Connection::serverUri

Definition at line 40 of file obix_http.h.

Referenced by addWatchItem(), createWatch(), getAbsUri(), http_closeConnection(), http_freeConnection(), http_initConnection(), http_openConnection(), http_registerDevice(), http_registerListener(), http_sendBatch(), http_unregisterListener(), ixmlelement_getFullHref(), parseWatchOut(), removeServerAddress(), removeWatch(), and setWatchTimeParam().

6.5.2.8 int _Http_Connection::serverUriLength

Definition at line 41 of file obix_http.h.

Referenced by addWatchItem(), createWatch(), getAbsUri(), http_initConnection(), http_registerDevice(), http_sendBatch(), http_unregisterListener(), ixmlelement_getFullHref(), parseWatchOut(), removeServerAddress(), and removeWatch().

6.5.2.9 char* _Http_Connection::signUpUri

Definition at line 48 of file obix_http.h.

Referenced by http_freeConnection(), http_initConnection(), http_openConnection(), and http_registerDevice().

6.5.2.10 char* _Http_Connection::watchAddUri

Definition at line 52 of file obix_http.h.

Referenced by addListener(), addWatchItem(), createWatch(), http_initConnection(), and resetWatchUris().

6.5.2.11 char* _Http_Connection::watchDeleteUri

Definition at line 54 of file obix_http.h.

Referenced by createWatch(), http_closeConnection(), http_initConnection(), removeWatch(), and resetWatchUris().

6.5.2.12 long _Http_Connection::watchLease

Definition at line 44 of file obix_http.h.

Referenced by http_initConnection(), and setWatchLeaseTime().

6.5.2.13 char* _Http_Connection::watchMakeUri

Definition at line 50 of file obix_http.h.

Referenced by createWatch(), http_freeConnection(), http_initConnection(), and http_openConnection().

6.5.2.14 pthread_mutex_t _Http_Connection::watchMutex

Definition at line 57 of file obix_http.h.

Referenced by addListener(), http_freeConnection(), http_initConnection(), removeListener(), removeWatch(), and watchPollTask().

6.5.2.15 char* _Http_Connection::watchPollChangesFullUri

Definition at line 51 of file obix_http.h.

Referenced by createWatch(), http_initConnection(), resetWatchUris(), and watchPollTask().

6.5.2.16 int _Http_Connection::watchPollTaskId

Definition at line 58 of file obix_http.h.

Referenced by addListener(), removeWatch(), and watchPollTask().

6.5.2.17 char* _Http_Connection::watchRemoveUri

Definition at line 53 of file obix_http.h.

Referenced by createWatch(), http_initConnection(), http_unregisterListener(), and resetWatchUris().

6.5.2.18 Table* _Http_Connection::watchTable

Definition at line 56 of file obix_http.h.

Referenced by addListener(), http_freeConnection(), http_initConnection(), parseWatchOut(), recreateWatch(), removeListener(), and removeWatch().

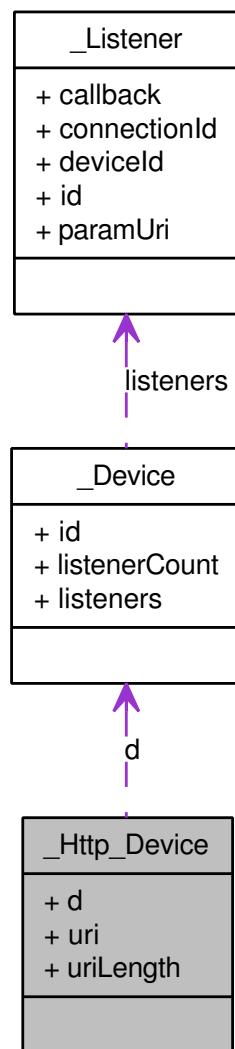
The documentation for this struct was generated from the following file:

- client/[obix_http.h](#)

6.6 _Http_Device Struct Reference

```
#include <obix_http.h>
```

Collaboration diagram for _Http_Device:



Data Fields

- [Device d](#)
- [char * uri](#)
- [int uriLength](#)

6.6.1 Detailed Description

Definition at line 65 of file obix_http.h.

6.6.2 Field Documentation

6.6.2.1 Device _Http_Device::d

Definition at line 67 of file obix_http.h.

Referenced by http_registerDevice().

6.6.2.2 char* _Http_Device::uri

Definition at line 69 of file obix_http.h.

Referenced by getAbsUri(), getRelUri(), http_registerDevice(), and http_unregisterDevice().

6.6.2.3 int _Http_Device::uriLength

Definition at line 70 of file obix_http.h.

Referenced by getAbsUri(), getRelUri(), and http_registerDevice().

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

- client/[obix_http.h](#)

6.7 _Listener Struct Reference

```
#include <obix_comm.h>
```

Data Fields

- [obix_update_listener callback](#)
- int [connectionId](#)
- int [deviceId](#)
- int [id](#)
- char * [paramUri](#)

6.7.1 Detailed Description

Definition at line 91 of file obix_comm.h.

6.7.2 Field Documentation

6.7.2.1 obix_update_listener _Listener::callback

Definition at line 97 of file obix_comm.h.

Referenced by listener_register(), and parseWatchOut().

6.7.2.2 int _Listener::connectionId

Definition at line 95 of file obix_comm.h.

Referenced by listener_register(), and parseWatchOut().

6.7.2.3 int _Listener::deviceId

Definition at line 94 of file obix_comm.h.

Referenced by listener_register(), and parseWatchOut().

6.7.2.4 int _Listener::id

Definition at line 93 of file obix_comm.h.

Referenced by listener_register(), and parseWatchOut().

6.7.2.5 char* _Listener::paramUri

Definition at line 96 of file obix_comm.h.

Referenced by http_unregisterListener(), listener_free(), and listener_register().

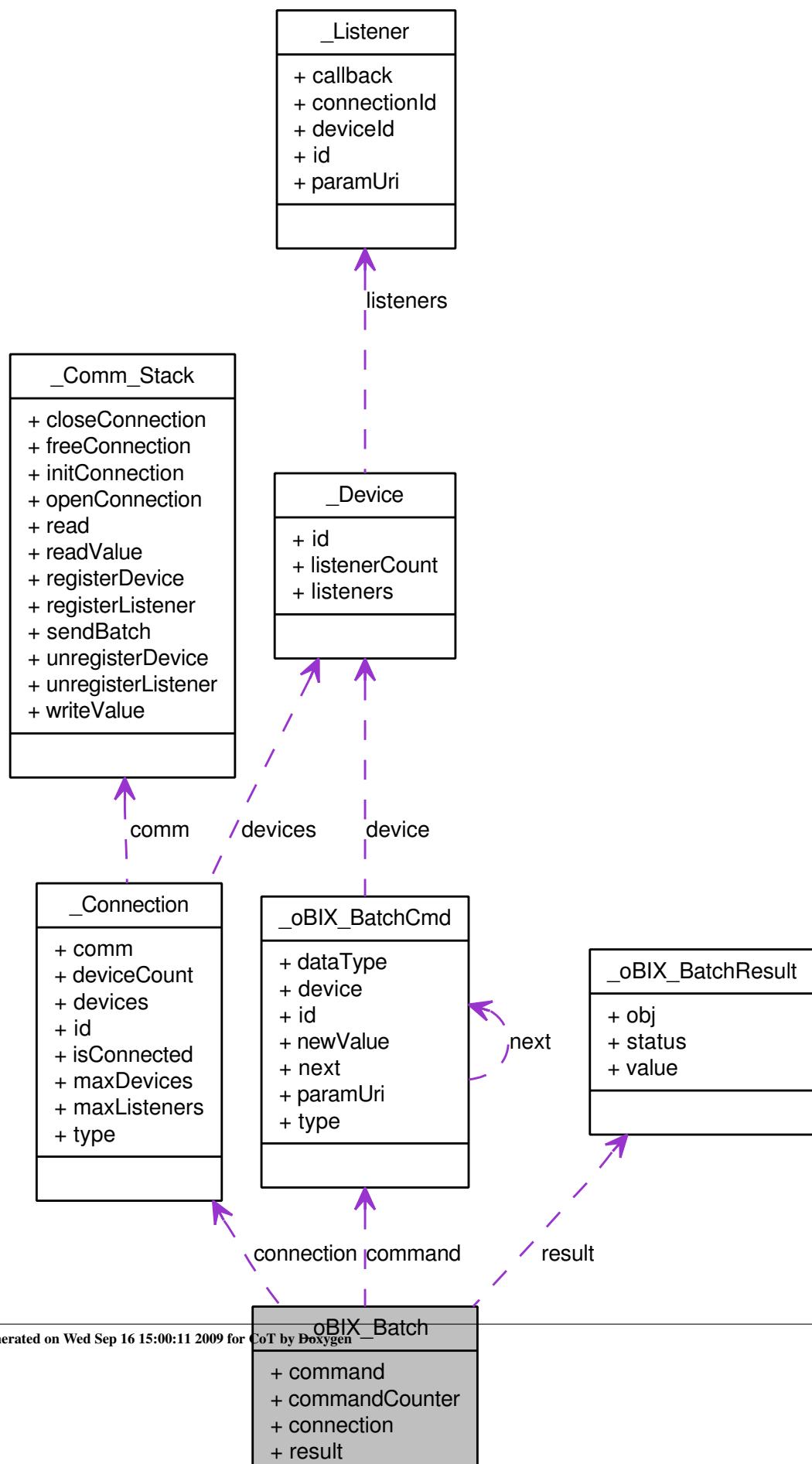
The documentation for this struct was generated from the following file:

- client/[obix_comm.h](#)

6.8 _oBIX_Batch Struct Reference

```
#include <obix_batch.h>
```

Collaboration diagram for _oBIX_Batch:



Data Fields

- `oBIX_BatchCmd * command`
- `int commandCounter`
- `Connection * connection`
- `oBIX_BatchResult * result`

6.8.1 Detailed Description

Definition at line 53 of file obix_batch.h.

6.8.2 Field Documentation

6.8.2.1 `oBIX_BatchCmd* _oBIX_Batch::command`

Definition at line 57 of file obix_batch.h.

Referenced by `getStrBatch()`, `http_sendBatch()`, `obix_batch_addCommand()`, `obix_batch_create()`, `obix_batch_free()`, and `obix_batch_removeCommand()`.

6.8.2.2 `int _oBIX_Batch::commandCounter`

Definition at line 56 of file obix_batch.h.

Referenced by `getStrBatch()`, `obix_batch_addCommand()`, `obix_batch_create()`, `obix_batch_resultClear()`, and `obix_batch_resultInit()`.

6.8.2.3 `Connection* _oBIX_Batch::connection`

Definition at line 55 of file obix_batch.h.

Referenced by `http_sendBatch()`, `obix_batch_addCommand()`, `obix_batch_create()`, and `obix_batch_send()`.

6.8.2.4 `oBIX_BatchResult* _oBIX_Batch::result`

Definition at line 58 of file obix_batch.h.

Referenced by `http_sendBatch()`, `obix_batch_create()`, `obix_batch_free()`, `obix_batch_getResult()`, `obix_batch_resultClear()`, and `obix_batch_resultInit()`.

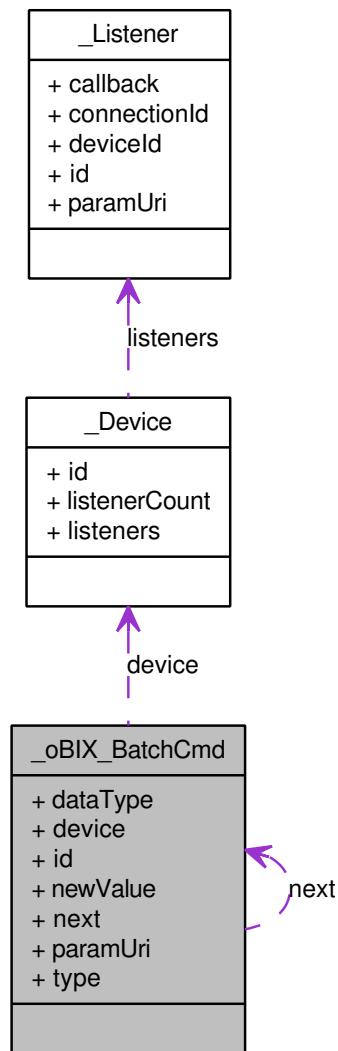
The documentation for this struct was generated from the following file:

- client/[obix_batch.h](#)

6.9 _oBIX_BatchCmd Struct Reference

```
#include <obix_batch.h>
```

Collaboration diagram for _oBIX_BatchCmd:



Data Fields

- `OBIX_DATA_TYPE dataType`
- `Device * device`
- `int id`
- `char * newValue`
- `struct _oBIX_BatchCmd * next`
- `char * paramUri`
- `OBIX_BATCH_CMD_TYPE type`

6.9.1 Detailed Description

Definition at line 41 of file obix_batch.h.

6.9.2 Field Documentation

6.9.2.1 OBIX_DATA_TYPE _oBIX_BatchCmd::dataType

Definition at line 48 of file obix_batch.h.

Referenced by getStrBatch(), and obix_batch_addCommand().

6.9.2.2 Device* _oBIX_BatchCmd::device

Definition at line 45 of file obix_batch.h.

Referenced by getStrBatch(), and obix_batch_addCommand().

6.9.2.3 int _oBIX_BatchCmd::id

Definition at line 44 of file obix_batch.h.

Referenced by getStrBatch(), http_sendBatch(), obix_batch_addCommand(), and obix_batch_removeCommand().

6.9.2.4 char* _oBIX_BatchCmd::newValue

Definition at line 47 of file obix_batch.h.

Referenced by getStrBatch(), obix_batch_addCommand(), and obix_batch_commandFree().

6.9.2.5 struct _oBIX_BatchCmd* _oBIX_BatchCmd::next [read]

Definition at line 49 of file obix_batch.h.

Referenced by getStrBatch(), http_sendBatch(), obix_batch_addCommand(), obix_batch_free(), and obix_batch_removeCommand().

6.9.2.6 char* _oBIX_BatchCmd::paramUri

Definition at line 46 of file obix_batch.h.

Referenced by getStrBatch(), obix_batch_addCommand(), and obix_batch_commandFree().

6.9.2.7 OBIX_BATCH_CMD_TYPE _oBIX_BatchCmd::type

Definition at line 43 of file obix_batch.h.

Referenced by getStrBatch(), http_sendBatch(), and obix_batch_addCommand().

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

- client/[obix_batch.h](#)

6.10 _oBIX_BatchResult Struct Reference

Contains outputs of the command, which was executed in a [Batch](#).

```
#include <obix_client.h>
```

Data Fields

- IXML_Element * **obj**
XML object returned by the function, if available (e.g.
- int **status**
Return value of the executed command.
- char * **value**
String value returned by the function, if available (e.g.

6.10.1 Detailed Description

Contains outputs of the command, which was executed in a [Batch](#).

Definition at line 467 of file obix_client.h.

6.10.2 Field Documentation

6.10.2.1 IXML_Element* _oBIX_BatchResult::obj

XML object returned by the function, if available (e.g.

for [obix_batch_read\(\)](#)).

Definition at line 483 of file obix_client.h.

Referenced by http_sendBatch(), obix_batch_resultClear(), and testBatch().

6.10.2.2 int _oBIX_BatchResult::status

Return value of the executed command.

It is identical to the return value of the corresponding command executed without Batch.

Definition at line 473 of file obix_client.h.

Referenced by http_sendBatch(), obix_batch_getResult(), obix_batch_resultClear(), obix_batch_resultInit(), and testBatch().

6.10.2.3 char* _oBIX_BatchResult::value

String value returned by the function, if available (e.g.

for [obix_batch_readValue\(\)](#)).

Definition at line 478 of file obix_client.h.

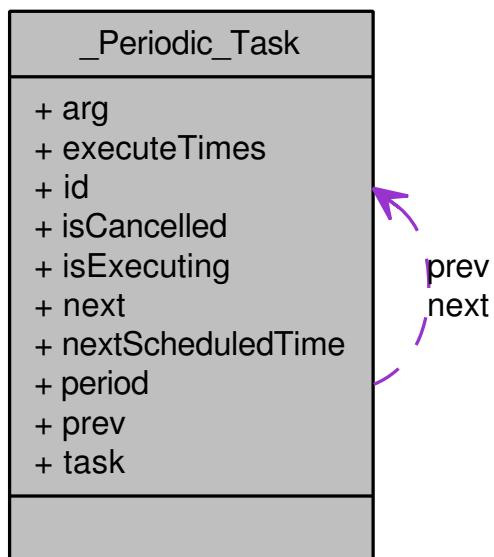
Referenced by http_sendBatch(), obix_batch_resultClear(), and testBatch().

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

- client/[obix_client.h](#)

6.11 _Periodic_Task Struct Reference

Collaboration diagram for _Periodic_Task:



Data Fields

- void * [arg](#)
- int [executeTimes](#)
- int [id](#)
- BOOL [isCancelled](#)
- BOOL [isExecuting](#)
- struct [_Periodic_Task](#) * [next](#)

- struct timespec [nextScheduledTime](#)
- struct timespec [period](#)
- struct [_Periodic_Task](#) * [prev](#)
- [periodic_task](#) [task](#)

6.11.1 Detailed Description

Definition at line 37 of file ptask.c.

6.11.2 Field Documentation

6.11.2.1 void* [_Periodic_Task::arg](#)

Definition at line 44 of file ptask.c.

Referenced by [periodicTask_create\(\)](#), and [periodicTask_execute\(\)](#).

6.11.2.2 int [_Periodic_Task::executeTimes](#)

Definition at line 42 of file ptask.c.

Referenced by [periodicTask_execute\(\)](#), [periodicTask_setPeriod\(\)](#), and [ptask_reschedule\(\)](#).

6.11.2.3 int [_Periodic_Task::id](#)

Definition at line 39 of file ptask.c.

Referenced by [periodicTask_create\(\)](#), [periodicTask_get\(\)](#), [ptask_schedule\(\)](#), [testGetClosestTask\(\)](#), [testPeriodicTask\(\)](#), and [threadCycle\(\)](#).

6.11.2.4 BOOL [_Periodic_Task::isCancelled](#)

Definition at line 45 of file ptask.c.

Referenced by [periodicTask_create\(\)](#), [periodicTask_execute\(\)](#), and [ptask_cancel\(\)](#).

6.11.2.5 BOOL [_Periodic_Task::isExecuting](#)

Definition at line 46 of file ptask.c.

Referenced by [periodicTask_create\(\)](#), [periodicTask_execute\(\)](#), and [ptask_cancel\(\)](#).

6.11.2.6 struct _Periodic_Task* _Periodic_Task::next [read]

Definition at line 48 of file ptask.c.

Referenced by periodicTask_create(), periodicTask_deleteRecursive(), periodicTask_get(), periodicTask_getClosest(), periodicTask_removeFromList(), ptask_schedule(), and testPeriodicTask().

6.11.2.7 struct timespec _Periodic_Task::nextScheduledTime [read]

Definition at line 40 of file ptask.c.

Referenced by periodicTask_generateNextExecTime(), periodicTask_getClosest(), periodicTask_resetExecTime(), ptask_reschedule(), testPtaskReschedule(), and threadCycle().

6.11.2.8 struct timespec _Periodic_Task::period [read]

Definition at line 41 of file ptask.c.

Referenced by periodicTask_generateNextExecTime(), periodicTask_setPeriod(), ptask_reschedule(), and ptask_schedule().

6.11.2.9 struct _Periodic_Task* _Periodic_Task::prev [read]

Definition at line 47 of file ptask.c.

Referenced by periodicTask_create(), periodicTask_removeFromList(), and ptask_schedule().

6.11.2.10 periodic_task _Periodic_Task::task

Definition at line 43 of file ptask.c.

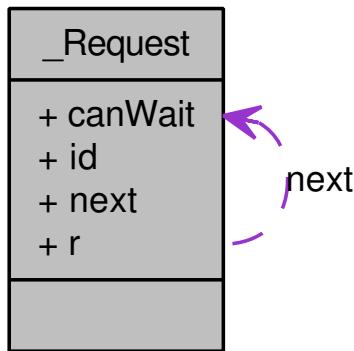
Referenced by periodicTask_create(), and periodicTask_execute().

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

- common/[ptask.c](#)

6.12 _Request Struct Reference

Collaboration diagram for _Request:



Data Fields

- `BOOL canWait`
- `int id`
- `struct _Request * next`
- `FCGX_Request r`

6.12.1 Detailed Description

Definition at line 66 of file obix_fcgi.c.

6.12.2 Field Documentation

6.12.2.1 `BOOL _Request::canWait`

Definition at line 70 of file obix_fcgi.c.

Referenced by `obix_fcgi_handleRequest()`, and `obixRequest_getHead()`.

6.12.2.2 `int _Request::id`

Definition at line 69 of file obix_fcgi.c.

Referenced by `main()`, and `obixRequest_create()`.

6.12.2.3 `struct _Request* _Request::next [read]`

Definition at line 71 of file obix_fcgi.c.

Referenced by obixRequest_create(), obixRequest_free(), obixRequest_getHead(), and obixRequest_release().

6.12.2.4 FCGX_Request _Request::r

Definition at line 68 of file obix_fcgi.c.

Referenced by main(), obix_fcgi_dumpEnvironment(), obix_fcgi_handleRequest(), obix_fcgi_readRequestInput(), obix_fcgi_sendResponse(), obix_fcgi_sendStaticErrorMessage(), obixRequest_create(), and obixRequest_free().

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

- server/[obix_fcgi.c](#)

6.13 _Table Struct Reference

```
#include <table.h>
```

Data Fields

- int [count](#)
- char ** [keys](#)
- int [size](#)
- void ** [values](#)

6.13.1 Detailed Description

Definition at line 32 of file table.h.

6.13.2 Field Documentation

6.13.2.1 int _Table::count

Definition at line 35 of file table.h.

Referenced by removeListener(), removeWatch(), table_create(), table_getKeys(), table_getValues(), table_put(), and table_remove().

6.13.2.2 char** _Table::keys

Definition at line 37 of file table.h.

Referenced by table_create(), table_extend(), table_free(), table_get(), table_getKeys(), table_put(), and table_remove().

6.13.2.3 int _Table::size

Definition at line 34 of file table.h.

Referenced by table_create(), table_extend(), table_free(), table_get(), table_put(), and table_remove().

6.13.2.4 void** _Table::values

Definition at line 38 of file table.h.

Referenced by table_create(), table_extend(), table_free(), table_get(), table_getValues(), and table_put().

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

- client

6.14 _Target Struct Reference

Target object which describes person's position on the sensor floor.

Data Fields

- **BOOL changed**
Defines whether this target has been changed recently.
- **int id**
Id of this target at the sensor floor.
- **BOOL new**
Defines whether this target new or not.
- **int removeTask**
*Id of the task which is scheduled to remove this target when it is not changed for **TARGET_REMOVE_TIMEOUT** milliseconds.*
- **char * uri**
Constant address of the target object at the oBIX server.
- **char * x**
X coordinate of the person.
- **char * y**
Y coordinate of the person.

6.14.1 Detailed Description

Target object which describes person's position on the sensor floor.

Definition at line 64 of file sensor_floor_driver.c.

6.14.2 Field Documentation

6.14.2.1 BOOL _Target::changed

Defines whether this target has been changed recently.

Definition at line 77 of file sensor_floor_driver.c.

Referenced by checkTargets(), target_sendUpdate(), target_setX(), target_setY(), and targetRemoveTask().

6.14.2.2 int _Target::id

Id of this target at the sensor floor.

Definition at line 73 of file sensor_floor_driver.c.

Referenced by checkTargets(), target_get(), and targetRemoveTask().

6.14.2.3 BOOL _Target::new

Defines whether this target new or not.

Definition at line 75 of file sensor_floor_driver.c.

Referenced by target_get(), target_sendUpdate(), and targetRemoveTask().

6.14.2.4 int _Target::removeTask

Id of the task which is scheduled to remove this target when it is not changed for **TARGET_REMOVE_TIMEOUT** milliseconds.

Definition at line 80 of file sensor_floor_driver.c.

Referenced by checkTargets(), target_get(), and targetRemoveTask().

6.14.2.5 char* _Target::uri

Constant address of the target object at the oBIX server.

Definition at line 67 of file sensor_floor_driver.c.

Referenced by loadDeviceData(), and target_sendUpdate().

6.14.2.6 char* _Target::x

X coordinate of the person.

Definition at line 69 of file [sensor_floor_driver.c](#).

Referenced by [target_sendUpdate\(\)](#), and [target_setX\(\)](#).

6.14.2.7 `char* _Target::y`

Y coordinate of the person.

Definition at line 71 of file [sensor_floor_driver.c](#).

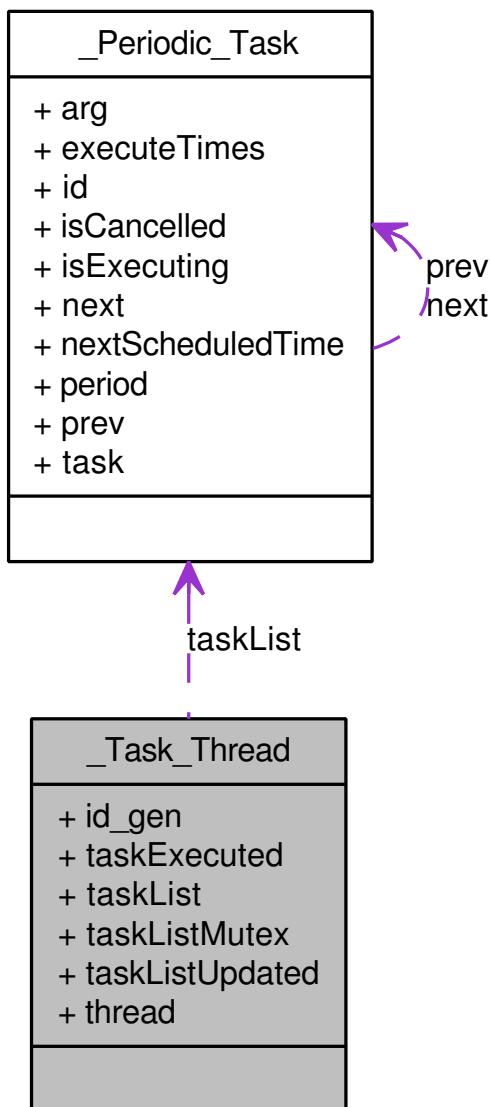
Referenced by [target_sendUpdate\(\)](#), and [target_setY\(\)](#).

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

- adapters/[sensor_floor_driver.c](#)

6.15 _Task_Thread Struct Reference

Collaboration diagram for _Task_Thread:



Data Fields

- int `id_gen`
- pthread_cond_t `taskExecuted`
- `Periodic_Task` * `taskList`
- pthread_mutex_t `taskListMutex`
- pthread_cond_t `taskListUpdated`
- pthread_t `thread`

6.15.1 Detailed Description

Definition at line 52 of file ptask.c.

6.15.2 Field Documentation

6.15.2.1 int _Task_Thread::id_gen

Definition at line 54 of file ptask.c.

Referenced by generateId(), ptask_init(), and test_ptask().

6.15.2.2 pthread_cond_t _Task_Thread::taskExecuted

Definition at line 59 of file ptask.c.

Referenced by periodicTask_execute(), ptask_cancel(), ptask_init(), stopTask(), and test_ptask().

6.15.2.3 Periodic_Task* _Task_Thread::taskList

Definition at line 55 of file ptask.c.

Referenced by periodicTask_get(), periodicTask_getClosest(), periodicTask_removeFromList(), ptask_init(), ptask_schedule(), stopTask(), test_ptask(), and testPeriodicTask().

6.15.2.4 pthread_mutex_t _Task_Thread::taskListMutex

Definition at line 57 of file ptask.c.

Referenced by periodicTask_execute(), ptask_cancel(), ptask_init(), ptask_reschedule(), ptask_reset(), ptask_schedule(), stopTask(), test_ptask(), testGetClosestTask(), and threadCycle().

6.15.2.5 pthread_cond_t _Task_Thread::taskListUpdated

Definition at line 58 of file ptask.c.

Referenced by ptask_cancel(), ptask_init(), ptask_reschedule(), ptask_reset(), ptask_schedule(), stopTask(), test_ptask(), and threadCycle().

6.15.2.6 pthread_t _Task_Thread::thread

Definition at line 56 of file ptask.c.

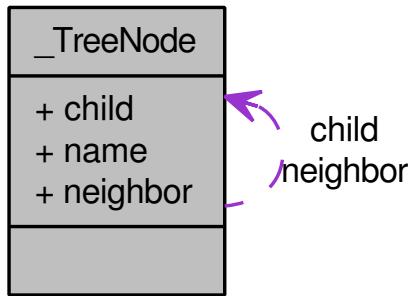
Referenced by ptask_dispose(), and ptask_init().

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

- common/ptask.c

6.16 _TreeNode Struct Reference

Collaboration diagram for _TreeNode:



Data Fields

- struct [_TreeNode](#) * [child](#)
- char * [name](#)
- struct [_TreeNode](#) * [neighbor](#)

6.16.1 Detailed Description

Definition at line 33 of file doctree.c.

6.16.2 Field Documentation

6.16.2.1 struct _TreeNode* _TreeNode::child [read]

Definition at line 37 of file doctree.c.

Referenced by [doctree_findNode\(\)](#).

6.16.2.2 char* _TreeNode::name

Definition at line 35 of file doctree.c.

Referenced by [doctree_findNode\(\)](#).

6.16.2.3 struct _TreeNode* _TreeNode::neighbor [read]

Definition at line 36 of file doctree.c.

Referenced by doctree_findNode().

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

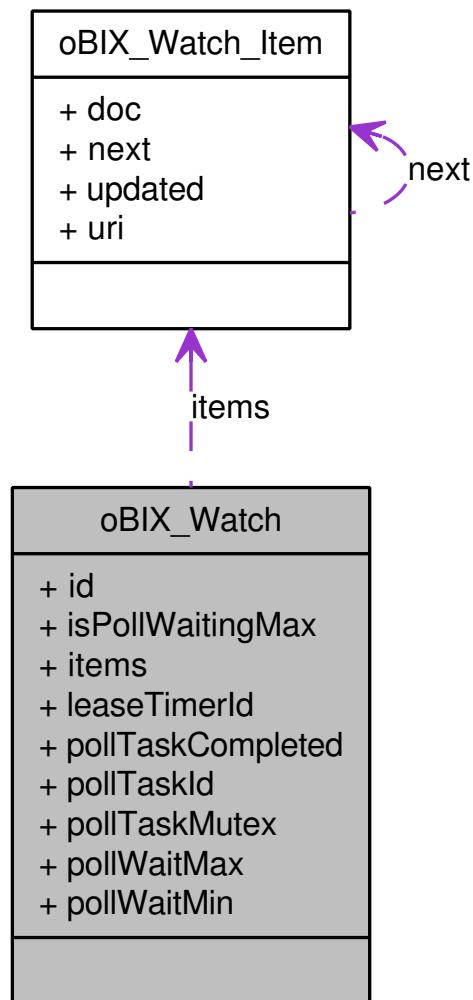
- server/doctree.c

6.17 oBIX_Watch Struct Reference

Represents an oBIX Watch object.

```
#include <watch.h>
```

Collaboration diagram for oBIX_Watch:



Data Fields

- int **id**
Id of the watch object.

- **BOOL isPollWaitingMax**
Defines whether long poll task is now waiting for max time.
- **oBIX_Watch_Item * items**
Pointer to the list of items monitored by this Watch object.
- **int leaseTimerId**
Id of the timer which removes old unused Watch object.
- **pthread_cond_t pollTaskCompleted**
Condition saying that long poll handler is completed.
- **int pollTaskId**
Id of the scheduled long poll request handler.
- **pthread_mutex_t pollTaskMutex**
Mutex for synchronization with scheduled long poll handler.
- **long pollWaitMax**
Maximum waiting time for long poll requests.
- **long pollWaitMin**
Minimum waiting time for long poll requests.

6.17.1 Detailed Description

Represents an oBIX Watch object.

Definition at line 76 of file watch.h.

6.17.2 Field Documentation

6.17.2.1 int oBIX_Watch::id

Id of the watch object.

Definition at line 79 of file watch.h.

Referenced by obixWatch_create(), obixWatch_createWatchItem(), obixWatch_free(), obixWatch_getUri(), obixWatch_processTimeUpdates(), obixWatch_resetLeaseTimer(), and taskDeleteWatch().

6.17.2.2 BOOL oBIX_Watch::isPollWaitingMax

Defines whether long poll task is now waiting for max time.

Definition at line 89 of file watch.h.

Referenced by obixWatch_create(), obixWatch_holdPoll(), and obixWatch_notifyPollTask().

6.17.2.3 oBIX_Watch_Item* oBIX_Watch::items

Pointer to the list of items monitored by this Watch object.

Definition at line 95 of file watch.h.

Referenced by findWatchItem(), obixWatch_appendWatchItem(), obixWatch_create(), obixWatch_deleteWatchItem(), obixWatch_free(), and pollWatchItemIterator().

6.17.2.4 int oBIX_Watch::leaseTimerId

Id of the timer which removes old unused Watch object.

Definition at line 81 of file watch.h.

Referenced by obixWatch_create(), obixWatch_delete(), and obixWatch_resetLeaseTimer().

6.17.2.5 pthread_cond_t oBIX_Watch::pollTaskCompleted

Condition saying that long poll handler is completed.

Definition at line 87 of file watch.h.

Referenced by obixWatch_create(), obixWatch_deleteHelper(), obixWatch_free(), and obixWatch_longPollTask().

6.17.2.6 int oBIX_Watch::pollTaskId

Id of the scheduled long poll request handler.

Definition at line 83 of file watch.h.

Referenced by obixWatch_create(), obixWatch_deleteHelper(), obixWatch_holdPoll(), obixWatch_longPollTask(), and obixWatch_notifyPollTask().

6.17.2.7 pthread_mutex_t oBIX_Watch::pollTaskMutex

Mutex for synchronization with scheduled long poll handler.

Definition at line 85 of file watch.h.

Referenced by obixWatch_create(), obixWatch_deleteHelper(), obixWatch_free(), obixWatch_holdPoll(), obixWatch_longPollTask(), and obixWatch_notifyPollTask().

6.17.2.8 long oBIX_Watch::pollWaitMax

Maximum waiting time for long poll requests.

Definition at line 93 of file watch.h.

Referenced by obixWatch_create(), obixWatch_holdPoll(), obixWatch_isLongPoll(), obixWatch_notifyPollTask(), and obixWatch_processTimeUpdates().

6.17.2.9 long oBIX_Watch::pollWaitMin

Minimum waiting time for long poll requests.

Definition at line 91 of file watch.h.

Referenced by obixWatch_create(), obixWatch_holdPoll(), obixWatch_notifyPollTask(), and obixWatch_processTimeUpdates().

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

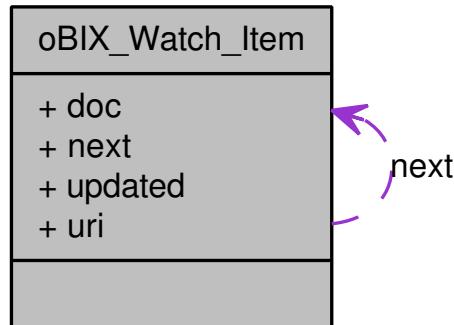
- server/[watch.h](#)

6.18 oBIX_Watch_Item Struct Reference

Represents a separate watch item.

```
#include <watch.h>
```

Collaboration diagram for oBIX_Watch_Item:



Data Fields

- IXML_Element * **doc**
Link to the corresponding object in the storage.
- struct oBIX_Watch_Item * **next**
Link to the next watch item in a list of items.
- IXML_Node * **updated**
Shows whether the object has been updated since last request.
- char * **uri**

URI of the object which state is monitored.

6.18.1 Detailed Description

Represents a separate watch item.

Watch item is a reference to the object which state is monitored by watch.

Definition at line 48 of file watch.h.

6.18.2 Field Documentation

6.18.2.1 IXML_Element* oBIX_Watch_Item::doc

Link to the corresponding object in the storage.

Definition at line 59 of file watch.h.

Referenced by handlerWatchAdd(), obixWatch_createWatchItem(), and pollWatchItemIterator().

6.18.2.2 struct oBIX_Watch_Item* oBIX_Watch_Item::next [read]

Link to the next watch item in a list of items.

Definition at line 69 of file watch.h.

Referenced by findWatchItem(), obixWatch_appendWatchItem(), obixWatch_createWatchItem(), obixWatch_deleteWatchItem(), obixWatchItem_free(), obixWatchItem_freeRecursive(), and pollWatchItemIterator().

6.18.2.3 IXML_Node* oBIX_Watch_Item::updated

Shows whether the object has been updated since last request.

In fact it is a link to the updated attribute of meta tag stored at [doc](#).

Definition at line 65 of file watch.h.

Referenced by obixWatch_createWatchItem(), obixWatchItem_free(), obixWatchItem_isUpdated(), and obixWatchItem_setUpdated().

6.18.2.4 char* oBIX_Watch_Item::uri

URI of the object which state is monitored.

Definition at line 55 of file watch.h.

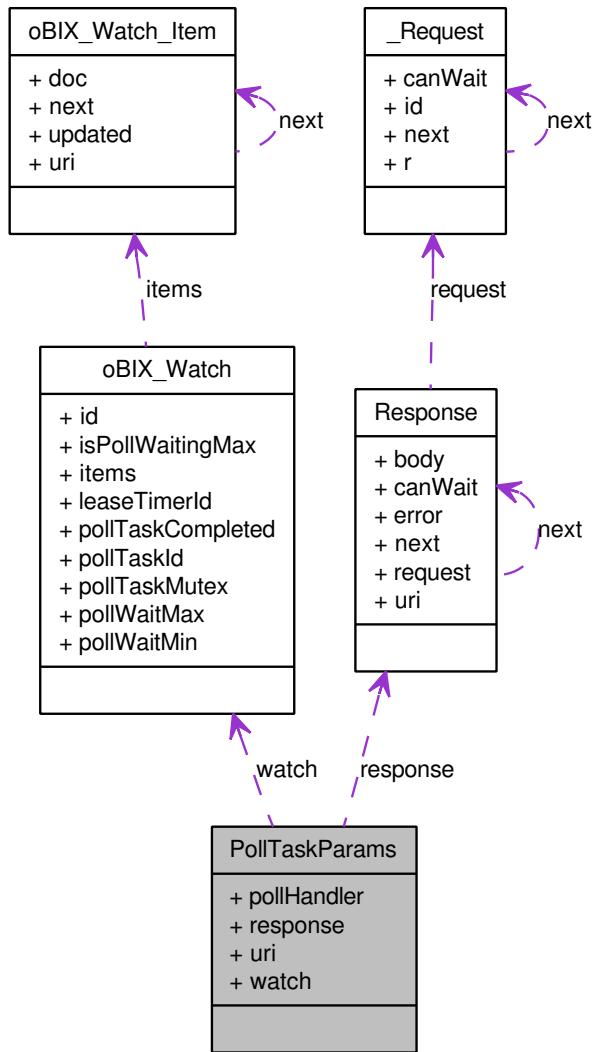
Referenced by findWatchItem(), obixWatch_createWatchItem(), obixWatchItem_free(), and pollWatchItemIterator().

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

- server/[watch.h](#)

6.19 PollTaskParams Struct Reference

Collaboration diagram for PollTaskParams:



Data Fields

- [obixWatch_pollHandler](#) pollHandler
- [Response *](#) response
- const char * uri
- [oBIX_Watch *](#) watch

6.19.1 Detailed Description

Definition at line 40 of file watch.c.

6.19.2 Field Documentation

6.19.2.1 obixWatch_pollHandler PollTaskParams::pollHandler

Definition at line 42 of file watch.c.

Referenced by obixWatch_holdPoll(), and obixWatch_longPollTask().

6.19.2.2 Response* PollTaskParams::response

Definition at line 44 of file watch.c.

Referenced by obixWatch_holdPoll(), and obixWatch_longPollTask().

6.19.2.3 const char* PollTaskParams::uri

Definition at line 45 of file watch.c.

Referenced by obixWatch_holdPoll(), and obixWatch_longPollTask().

6.19.2.4 oBIX_Watch* PollTaskParams::watch

Definition at line 43 of file watch.c.

Referenced by obixWatch_holdPoll(), and obixWatch_longPollTask().

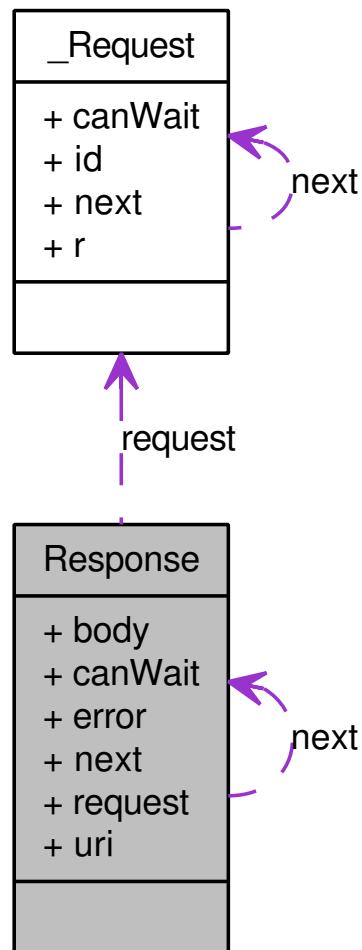
The documentation for this struct was generated from the following file:

- server/[watch.c](#)

6.20 Response Struct Reference

```
#include <response.h>
```

Collaboration diagram for Response:



Data Fields

- `char * body`
- `BOOL canWait`
- `BOOL error`
- `struct Response * next`
- `Request * request`
- `char * uri`

6.20.1 Detailed Description

Definition at line 35 of file response.h.

6.20.2 Field Documentation

6.20.2.1 **char* Response::body**

Definition at line 37 of file response.h.

Referenced by checkResponse(), findInResponse(), obix_fcgi_dumpEnvironment(), obix_fcgi_sendResponse(), obixResponse_create(), obixResponse_free(), obixResponse_setError(), obixResponse_setText(), printResponse(), and testGenerateResponse().

6.20.2.2 **BOOL Response::canWait**

Definition at line 40 of file response.h.

Referenced by obixResponse_create(), and obixWatch_holdPoll().

6.20.2.3 **BOOL Response::error**

Definition at line 39 of file response.h.

Referenced by normalizeObixDocument(), obix_server_generateObixErrorMessage(), obix_server_init(), obix_server_write(), obixResponse_create(), obixResponse_isError(), obixResponse_setError(), and obixResponse_setErrorFlag().

6.20.2.4 **struct Response* Response::next [read]**

Definition at line 42 of file response.h.

Referenced by checkResponse(), findInResponse(), handlerBatch(), handlerWatchPollHelper(), obix_fcgi_dumpEnvironment(), obix_fcgi_sendResponse(), obixResponse_add(), obixResponse_create(), obixResponse_free(), printResponse(), and sendErrorMessage().

6.20.2.5 **Request* Response::request**

Definition at line 41 of file response.h.

Referenced by checkResponse(), obix_fcgi_dumpEnvironment(), obix_fcgi_sendResponse(), obixResponse_create(), and obixResponse_isHead().

6.20.2.6 **char* Response::uri**

Definition at line 38 of file response.h.

Referenced by obix_fcgi_sendResponse(), obix_server_generateResponse(), obixResponse_create(), obixResponse_free(), obixResponse_setRightUri(), and testResponse_setRightUri().

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

- server/response.h

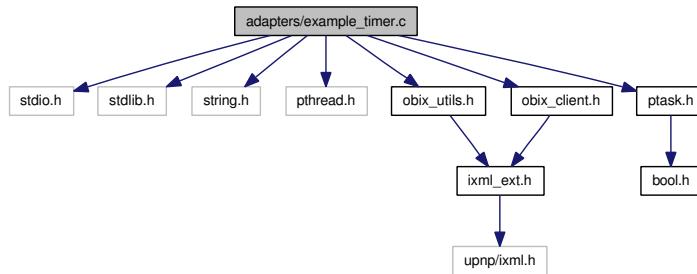
7 File Documentation

7.1 adapters/example_timer.c File Reference

Simple oBIX Timer device implementation, which demonstrates usage of oBIX client library.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <obix_utils.h>
#include <obix_client.h>
#include <ptask.h>
```

Include dependency graph for example_timer.c:



Defines

- #define CONNECTION_ID 0
ID of the connection which is described in configuration file.

Functions

- char * [getDeviceData](#) (char *deviceUri)
Generates device data in oBIX format.
- int [main](#) (int argc, char **argv)
Entry point of the Timer application.
- int [resetListener](#) (int connectionId, int deviceId, int listenerId, const char *newValue)

Handles changes of "reset" value.

- void **timerTask** (void *arg)
Updates timer value and writes it to oBIX server.

Variables

- Task_Thread * **_taskThread**
Separate thread for timer value updating.
- long **_time**
Elapsed time is stored here.
- pthread_mutex_t **_time_mutex** = PTHREAD_MUTEX_INITIALIZER
Need mutex for synchronization, because _time variable is accessed from two threads.
- int **_timerTaskId**
ID of the task which increases time every second.
- const char * **DEVICE_DATA**
Data which is posted to oBIX server.

7.1.1 Detailed Description

Simple oBIX Timer device implementation, which demonstrates usage of oBIX client library.

It shows the time elapsed after timer was started or reset by user. Device registers itself at oBIX server, regularly updates elapsed time on it and listens to updates of "reset" parameter. If someone changes "reset" to true, than elapsed time is set to 0. Configuration file template can be found at res/example_timer-config.xml

Author:

Andrey Litvinov

Version:

1.1

Definition in file [example_timer.c](#).

7.1.2 Define Documentation

7.1.2.1 #define CONNECTION_ID 0

ID of the connection which is described in configuration file.

Definition at line 46 of file example_timer.c.

Referenced by main(), and timerTask().

7.1.3 Function Documentation

7.1.3.1 `char* getDeviceData (char * deviceUri)`

Generates device data in oBIX format.

Parameters:

`deviceUri` Address at the server where device will be stored to. This address will be written as the `href` attribute of the root object.

Returns:

Data which should be posted to the server.

Definition at line 215 of file example_timer.c.

References DEVICE_DATA.

Referenced by main().

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

Entry point of the Timer application.

It takes the name of the configuration file (use example_timer_config.xml).

See also:

[example_timer_config.xml](#)

Definition at line 233 of file example_timer.c.

References _timerTaskId, CONNECTION_ID, EXECUTE_INDEFINITE, getDeviceData(), obix_dispose(), obix_loadConfigFile(), obix_openConnection(), obix_registerDevice(), obix_registerListener(), OBIX_SUCCESS, ptask_cancel(), ptask_dispose(), ptask_init(), ptask_schedule(), resetListener(), timerTask(), and TRUE.

7.1.3.3 `int resetListener (int connectionId, int deviceId, int listenerId, const char * newValue)`

Handles changes of "reset" value.

The function implements [obix_update_listener\(\)](#) prototype and is registered as a listener of "reset" param using [obix_registerListener\(\)](#). If "reset" value is changed at oBIX server to "true" it will set it back to "false" and reset timer.

See also:

[obix_update_listener\(\)](#), [obix_registerListener\(\)](#).

Definition at line 76 of file example_timer.c.

References `_time`, `_time_mutex`, `_timerTaskId`, `obix_batch_create()`, `obix_batch_send()`, `obix_batch_writeValue()`, `OBIX_SUCCESS`, `OBIX_T_BOOL`, `OBIX_T_RELTIME`, `ptask_reset()`, and `XML_FALSE`. Referenced by `main()`.

7.1.3.4 void timerTask (void * *arg*)

Updates timer value and writes it to oBIX server.

Implements `periodic_task()` prototype and is scheduled using `ptask_schedule()`. This method is executed in a separate thread that is why it uses `_time_mutex` for synchronization with `resetListener()` which sets timer to 0.

See also:

[periodic_task\(\)](#), [ptask_schedule\(\)](#).

Parameters:

arg Assumes that a pointer to the device ID is passed here. Device ID is used for updating time value at the server.

Definition at line 182 of file example_timer.c.

References `_time`, `_time_mutex`, `CONNECTION_ID`, `obix_reltimes_fromLong()`, `OBIX_SUCCESS`, `OBIX_T_RELTIME`, `obix_writeValue()`, and `RELTIME_DAY`.

Referenced by `main()`.

7.1.4 Variable Documentation

7.1.4.1 Task_Thread* _taskThread

Separate thread for timer value updating.

Definition at line 63 of file example_timer.c.

7.1.4.2 long _time

Elapsed time is stored here.

Definition at line 55 of file example_timer.c.

Referenced by `resetListener()`, and `timerTask()`.

7.1.4.3 pthread_mutex_t _time_mutex = PTHREAD_MUTEX_INITIALIZER

Need mutex for synchronization, because `_time` variable is accessed from two threads.

Definition at line 61 of file example_timer.c.

Referenced by resetListener(), and timerTask().

7.1.4.4 int _timerTaskId

ID of the task which increases time every second.

Definition at line 65 of file example_timer.c.

Referenced by main(), and resetListener().

7.1.4.5 const char* DEVICE_DATA

Initial value:

```
"<obj name=\"ExampleTimer\" displayName=\"Example Timer\" href=\"%s\">\r\n"
"  <reltime name=\"time\" displayName=\"Elapsed Time\" href=\"%stime\" val=\"PT0S\" writable=\"true\"/>
"  <bool name=\"reset\" displayName=\"Reset Timer\" href=\"%sreset\" val=\"false\" writable=\"true\"/>
"</obj>"
```

Data which is posted to oBIX server.

Definition at line 48 of file example_timer.c.

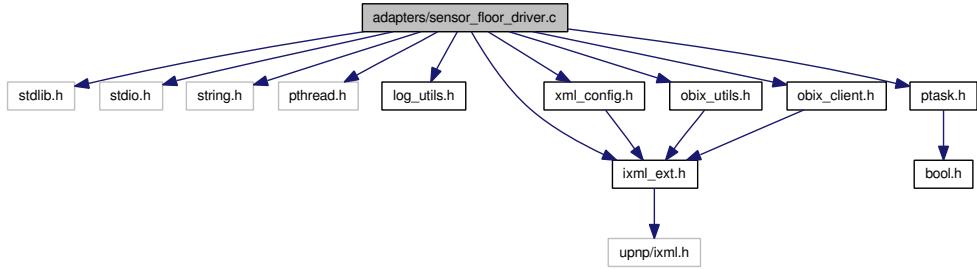
Referenced by getDeviceData().

7.2 adapters/sensor_floor_driver.c File Reference

This is an adapter for the Elsi Sensor Floor.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <log_utils.h>
#include <ixml_ext.h>
#include <xml_config.h>
#include <ptask.h>
#include <obix_utils.h>
#include <obix_client.h>
```

Include dependency graph for sensor_floor_driver.c:



Data Structures

- struct `_Target`
Target object which describes person's position on the sensor floor.

Defines

- #define `LAST_EVENT_POLL_PERIOD` 3000
We have to poll manually last event in order to check for the fallen event.
- #define `SENSOR_FLOOR_CONNECTION` 0
Id of the connection to the sensor floor described in configuration file.
- #define `SENSOR_FLOOR_FEED_CONNECTION` 1
Id of the 2nd connection to the sensor floor form configuration file.
- #define `SERVER_CONNECTION` 2
Id of the connection to the target oBIX server.
- #define `TARGET_REMOVE_TIMEOUT` 15000
Timeout after which the target is consider unavailable.

Typedefs

- typedef struct `_Target` `Target`
Target object which describes person's position on the sensor floor.

Functions

- void `checkEventsTask` (void *args)
Checks periodically the last sensor floor event and invokes event feed listener.
- int `checkTargets` ()
Checks whether targets are updated.

- int **eventFeedListener** (int connectionId, int deviceId, int listenerId, const char *newValue)
Listens to the event feed of the sensor floor.
- int **loadDeviceData** (IXML_Element *deviceConf)
Generates device data which will be published on the oBIX server.
- int **loadSettings** (const char *fileName)
Loads driver settings from the specified configuration file.
- int **main** (int argc, char **argv)
Entry point of the driver.
- int **parseTarget** (IXML_Node *node)
Parses target object received from the sensor floor.
- void **resetTargets** ()
Sets all targets at server to 'not available' and removes all target values.
- int **sendFallEventUpdate** (IXML_Document *events)
Sets the fall variable at the oBIX server to true if the fallen event is received from the sensor floor.
- void **targetRemoveTask** (void *arg)
Task which clears target object at the oBIX server if it is not changed for TARGET_REMOVE_TIMEOUT milliseconds.
- int **targetsListener** (int connectionId, int deviceId, int listenerId, const char *newValue)
Listens to target updates at the sensor floor.
- void **testCycle** ()
Emulates target data from the sensor floor.

Targets management

Utility methods for work with target object.

- Target * **target_get** (int id)
Searches for target with specified Id.
- int **target_sendUpdate** (Target *target)
Sends update of the target's values to the oBIX server.
- void **target_setX** (Target *target, const char *newValue)
Sets X value of the target object.
- void **target_setY** (Target *target, const char *newValue)
Sets Y value of the target object.

Variables

- IXML_Document * [_deviceData](#)
Contains device data which is published to the oBIX server.
- int [_deviceId](#)
Id of the device which is registered at the oBIX server.
- pthread_mutex_t [_targetMutex](#) = PTHREAD_MUTEX_INITIALIZER
Mutex for thread synchronization: Targets are changed from one thread and removed from another.
- Target * [_targets](#)
Head of the targets list.
- int [_targetsCount](#)
Number of targets in the targets list.
- Task_Thread * [_taskThread](#)
Thread for scheduling asynchronous tasks.
- IXML_Document * [_testData](#)
Contains data for testing.

7.2.1 Detailed Description

This is an adapter for the Elsi Sensor Floor.

The sensor floor itself is already oBIX-enabled: it has an oBIX server which is publishing people positions and different events generated by the floor. This adapter just copies this data from floor's oBIX server to another one and changes the data structure: The floor's server uses 'target' objects to describe person's position, which are dynamically appear and disappear depending on the available data from the floor. The adapter gathers data from these targets and writes it to a static objects on the target server which are easier to deal with.

Author:

Andrey Litvinov

Version:

1.1

Definition in file [sensor_floor_driver.c](#).

7.2.2 Define Documentation

7.2.2.1 #define LAST_EVENT_POLL_PERIOD 3000

We have to poll manually last event in order to check for the fallen event.

Todo

Check whether we can monitor it by including to the Watch.

Definition at line 61 of file sensor_floor_driver.c.

Referenced by main().

7.2.2.2 #define SENSOR_FLOOR_CONNECTION 0

Id of the connection to the sensor floor described in configuration file.

Definition at line 50 of file sensor_floor_driver.c.

Referenced by main(), and testCycle().

7.2.2.3 #define SENSOR_FLOOR_FEED_CONNECTION 1

Id of the 2nd connection to the sensor floor form configuration file.

Definition at line 52 of file sensor_floor_driver.c.

Referenced by checkEventsTask(), and main().

7.2.2.4 #define SERVER_CONNECTION 2

Id of the connection to the target oBIX server.

Definition at line 54 of file sensor_floor_driver.c.

Referenced by main(), sendFallEventUpdate(), and target_sendUpdate().

7.2.2.5 #define TARGET_REMOVE_TIMEOUT 15000

Timeout after which the target is consider unavailable.

Definition at line 57 of file sensor_floor_driver.c.

Referenced by target_get().

7.2.3 Typedef Documentation**7.2.3.1 typedef struct _Target Target**

Target object which describes person's position on the sensor floor.

7.2.4 Function Documentation

7.2.4.1 void checkEventsTask (void * *args*)

Checks periodically the last sensor floor event and invokes event feed listener.

Todo

it is created because Sensor Floor's event feed doesn't show the Fallen event

Parameters:

args Parameter is not used (created to match [periodic_task](#) prototype)

Definition at line 783 of file sensor_floor_driver.c.

References `ixmlElement_freeOwnerDocument()`, `ixmlElement_getNode()`, `log_debug`, `log_error`, `obix_-read()`, `OBIX_SUCCESS`, `sendFallEventUpdate()`, and `SENSOR_FLOOR_FEED_CONNECTION`.

Referenced by `main()`.

7.2.4.2 int checkTargets ()

Checks whether targets are updated.

If yes - sends updated values to the server.

Returns:

0 on success, -1 on error

Definition at line 409 of file sensor_floor_driver.c.

References `_targetsCount`, `_Target::changed`, `_Target::id`, `OBIX_SUCCESS`, `ptask_reset()`, `_Target::removeTask`, `target_sendUpdate()`, and `TRUE`.

Referenced by `targetsListener()`.

7.2.4.3 int eventFeedListener (int *connectionId*, int *deviceId*, int *listenerId*, const char * *newValue*)

Listens to the event feed of the sensor floor.

Implements [obix_update_listener](#). `obix_update_listener`

Definition at line 566 of file sensor_floor_driver.c.

References `log_debug`, `log_error`, and `sendFallEventUpdate()`.

Referenced by `main()`.

7.2.4.4 int loadDeviceData (IXML_Element * *deviceConf*)

Generates device data which will be published on the oBIX server.

Uses settings from configuration file.

Parameters:

deviceConf XML configuration loaded from the file. return *0* on success, *-1* on error.

Definition at line 607 of file sensor_floor_driver.c.

References *_deviceData*, *_targetsCount*, *config_getChildTag()*, *config_getTagAttrIntValue()*, *ixmlElement_cloneWithLog()*, *ixmlElement_getNode()*, *ixmlNode_convertToElement()*, *OBIX_ATTR_DISPLAY_NAME*, *OBIX_ATTR_HREF*, *OBIX_ATTR_NAME*, *OBIX_OBJ*, *TRUE*, and *_Target::uri*.

Referenced by *loadSettings()*.

7.2.4.5 int loadSettings (const char **fileName*)

Loads driver settings from the specified configuration file.

Parameters:

fileName Name of the configuration file.

Returns:

0 on success, *-1* on error.

Definition at line 687 of file sensor_floor_driver.c.

References *_testData*, *config_finishInit()*, *config_getChildTag()*, *config_loadFile()*, *config_log()*, *FALSE*, *ixmlElement_cloneWithLog()*, *ixmlElement_getNode()*, *loadDeviceData()*, *obix_loadConfig()*, *OBIX_SUCCESS*, and *TRUE*.

Referenced by *main()*.

7.2.4.6 int main (int *argc*, char *argv*)**

Entry point of the driver.

Parameters:

argc It takes exactly 1 input argument.

argv The only argument is the name of configuration file.

Returns:

0 on success, negative error code otherwise.

Definition at line 814 of file sensor_floor_driver.c.

References *_deviceData*, *_deviceId*, *checkEventsTask()*, *eventFeedListener()*, *EXECUTE_INDEFINITE*, *FALSE*, *LAST_EVENT_POLL_PERIOD*, *loadSettings()*, *log_error*, *obix_dispose()*,

obix_openConnection(), obix_registerDevice(), obix_registerListener(), OBIX_SUCCESS, obix_unregisterListener(), ptask_cancel(), ptask_dispose(), ptask_init(), ptask_schedule(), resetTargets(), SENSOR_FLOOR_CONNECTION, SENSOR_FLOOR_FEED_CONNECTION, SERVER_CONNECTION, targetsListener(), and TRUE.

7.2.4.7 int parseTarget (IXML_Node * *node*)

Parses target object received from the sensor floor.

Parameters:

node XML representation of the received target object.

Returns:

0 on success, -1 on error.

Definition at line 286 of file sensor_floor_driver.c.

References ixmlNode_convertToElement(), log_error, OBIX_ATTR_NAME, OBIX_ATTR_VAL, obix_obj_implementsContract(), target_get(), target_setX(), and target_setY().

Referenced by targetsListener().

7.2.4.8 void resetTargets ()

Sets all targets at server to 'not available' and removes all target values.

Definition at line 758 of file sensor_floor_driver.c.

References _targetsCount, and targetRemoveTask().

Referenced by main().

7.2.4.9 int sendFallEventUpdate (IXML_Document * *events*)

Sets the *fall* variable at the oBIX server to *true* if the fallen event is received from the sensor floor.

Parameters:

events XML document which contains received events from the sensor floor.

Returns:

- 1 If fallen even is detected and the *fall* variable is changed on the oBIX server.
- 0 If fallen event is not detected.
- -1 If error occurred when update was sent to the server.

Definition at line 508 of file sensor_floor_driver.c.

References `_deviceId`, `_targetMutex`, `FALSE`, `ixmlDocument_getElementByAttrValue()`, `log_error`, `log_warning`, `OBIX_ATTR_VAL`, `OBIX_OBJ`, `OBIX_SUCCESS`, `OBIX_T_BOOL`, `obix_writeValue()`, `SERVER_CONNECTION`, `TRUE`, and `XML_TRUE`.

Referenced by `checkEventsTask()`, and `eventFeedListener()`.

7.2.4.10 Target* target_get (int *id*)

Searches for target with specified Id.

Parameters:

id Id of the target which should be found.

Returns:

Target with specified Id. If no such target found, search for a free target object and assigns provided Id to it. If there are no free targets, `NULL` is returned.

Definition at line 152 of file `sensor_floor_driver.c`.

References `_targetsCount`, `_Target::id`, `_Target::new`, `ptask_schedule()`, `_Target::removeTask`, `TARGET_REMOVE_TIMEOUT`, `targetRemoveTask()`, and `TRUE`.

Referenced by `parseTarget()`.

7.2.4.11 int target_sendUpdate (Target * *target*)

Sends update of the target's values to the oBIX server.

Parameters:

target Target object which should be sent.

Returns:

`OBIX_SUCCESS` if data is successfully sent, error code otherwise.

Definition at line 191 of file `sensor_floor_driver.c`.

References `_deviceId`, `_Target::changed`, `FALSE`, `_Target::new`, `obix_batch_create()`, `obix_batch_free()`, `obix_batch_send()`, `obix_batch_writeValue()`, `OBIX_ERR_NO_MEMORY`, `OBIX_SUCCESS`, `OBIX_T_BOOL`, `OBIX_T_REAL`, `SERVER_CONNECTION`, `TRUE`, `_Target::uri`, `_Target::x`, `XML_FALSE`, `XML_TRUE`, and `_Target::y`.

Referenced by `checkTargets()`, and `targetRemoveTask()`.

7.2.4.12 void target_setX (Target * *target*, const char * *newValue*)

Sets X value of the target object.

Parameters:

target Target object which should be changed.

newValue New X value.

Definition at line 114 of file sensor_floor_driver.c.

References _Target::changed, TRUE, and _Target::x.

Referenced by parseTarget(), and targetRemoveTask().

7.2.4.13 void target_setY (Target * *target*, const char * *newValue*)

Sets Y value of the target object.

Parameters:

target Target object which should be changed.

newValue New Y value.

Definition at line 132 of file sensor_floor_driver.c.

References _Target::changed, TRUE, and _Target::y.

Referenced by parseTarget(), and targetRemoveTask().

7.2.4.14 void targetRemoveTask (void * *arg*)

Task which clears target object at the oBIX server if it is not changed for TARGET_REMOVE_TIMEOUT milliseconds.

It is scheduled for each target object. Implements [periodic_task](#) prototype.

Parameters:

arg Reference to the target object which should be removed.

Definition at line 371 of file sensor_floor_driver.c.

References _targetMutex, _Target::changed, FALSE, _Target::id, log_error, _Target::new, OBIX_SUCCESS, ptask_reset(), _Target::removeTask, target_sendUpdate(), target_setX(), target_setY(), and TRUE.

Referenced by resetTargets(), and target_get().

7.2.4.15 int targetsListener (int *connectionId*, int *deviceId*, int *listenerId*, const char * *newValue*)

Listens to target updates at the sensor floor.

Implements [obix_update_listener](#) prototype. [obix_update_listener](#)

Definition at line 442 of file sensor_floor_driver.c.

References `_targetMutex`, `_targetsCount`, `checkTargets()`, `log_error`, `log_warning`, `OBIX_OBJ`, and `parseTarget()`.

Referenced by `main()`, and `testCycle()`.

7.2.4.16 void testCycle ()

Emulates target data from the sensor floor.

Test data is read from the configuration file and sent step by step to the targets listener. For testing purposes only.

Definition at line 732 of file `sensor_floor_driver.c`.

References `_testData`, `log_warning`, `OBIX_OBJ_LIST`, `SENSOR_FLOOR_CONNECTION`, and `targetsListener()`.

7.2.5 Variable Documentation

7.2.5.1 IXML_Document* _deviceData

Contains device data which is published to the oBIX server.

Definition at line 85 of file `sensor_floor_driver.c`.

Referenced by `loadDeviceData()`, and `main()`.

7.2.5.2 int _deviceId

Id of the device which is registered at the oBIX server.

Definition at line 91 of file `sensor_floor_driver.c`.

Referenced by `main()`, `sendFallEventUpdate()`, and `target_sendUpdate()`.

7.2.5.3 pthread_mutex_t _targetMutex = PTHREAD_MUTEX_INITIALIZER

Mutex for thread synchronization: Targets are changed from one thread and removed from another.

Definition at line 96 of file `sensor_floor_driver.c`.

Referenced by `sendFallEventUpdate()`, `targetRemoveTask()`, and `targetsListener()`.

7.2.5.4 Target* _targets

Head of the targets list.

Definition at line 87 of file `sensor_floor_driver.c`.

7.2.5.5 int _targetsCount

Number of targets in the targets list.

Definition at line 89 of file sensor_floor_driver.c.

Referenced by checkTargets(), loadDeviceData(), resetTargets(), target_get(), and targetsListener().

7.2.5.6 Task_Thread* _taskThread

Thread for scheduling asynchronous tasks.

Definition at line 93 of file sensor_floor_driver.c.

7.2.5.7 IXML_Document* _testData

Contains data for testing.

See also:

[testCycle\(\)](#).

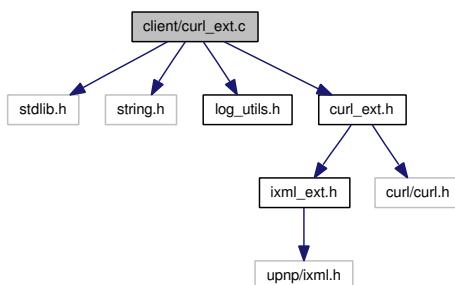
Definition at line 99 of file sensor_floor_driver.c.

Referenced by loadSettings(), and testCycle().

7.3 client(curl_ext.c) File Reference

```
#include <stdlib.h>
#include <string.h>
#include <log_utils.h>
#include "curl_ext.h"
```

Include dependency graph for curl_ext.c:



Defines

- #define [DEF_INPUT_BUFFER_SIZE](#) 2048

- #define REQUEST_HTTP_POST 1
- #define REQUEST_HTTP_PUT 0

Functions

- static CURL_EXT * curl_ext_allocateMemory ()
- int curl_ext_create (CURL_EXT **handle)
- void curl_ext_dispose ()
- void curl_ext_free (CURL_EXT *handle)
- static void curl_ext_freeMemory (CURL_EXT *handle)
- int curl_ext_get (CURL_EXT *handle, const char *uri)
- int curl_ext_getDOM (CURL_EXT *handle, const char *uri, IXML_Document **response)
- int curl_ext_init (int defaultInputBufferSize)
- int curl_ext_post (CURL_EXT *handle, const char *uri)
- int curl_ext_postDOM (CURL_EXT *handle, const char *uri, IXML_Document **response)
- int curl_ext_put (CURL_EXT *handle, const char *uri)

Helper function for curl_ext_put and curl_ext_post.

- int curl_ext_putDOM (CURL_EXT *handle, const char *uri, IXML_Document **response)
- static size_t inputWriter (char *inputData, size_t size, size_t nmemb, void *arg)

libcurl write callback function.

- static size_t outputReader (char *outputData, size_t size, size_t nmemb, void *arg)

libcurl read callback function.

- static int parseXmlInput (CURL_EXT *handle, IXML_Document **doc)
- static int sendRequest (CURL_EXT *handle, const char *uri)

Helper function which performs actual HTTP request assumes that type of request was already set by a caller.

Variables

- static int _defaultInputBufferSize = DEF_INPUT_BUFFER_SIZE
- static struct curl_slist * _header

7.3.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [curl_ext.c](#).

7.3.2 Define Documentation

7.3.2.1 `#define DEF_INPUT_BUFFER_SIZE 2048`

Definition at line 34 of file curl_ext.c.

7.3.2.2 `#define REQUEST_HTTP_POST 1`

Definition at line 36 of file curl_ext.c.

Referenced by testCurlExt(), and testCurlExtRequest().

7.3.2.3 `#define REQUEST_HTTP_PUT 0`

Definition at line 35 of file curl_ext.c.

Referenced by testCurlExt(), and testCurlExtRequest().

7.3.3 Function Documentation

7.3.3.1 `static CURL_EXT* curl_ext_allocateMemory () [static]`

Definition at line 133 of file curl_ext.c.

References `_defaultInputBufferSize`, `_CURL_EXT::curl`, `curl_ext_freeMemory()`, `_CURL_EXT::errorBuffer`, `_CURL_EXT::inputBuffer`, `_CURL_EXT::inputBufferFree`, `_CURL_EXT::inputBufferSize`, `_CURL_EXT::outputBuffer`, `_CURL_EXT::outputPos`, and `_CURL_EXT::outputSize`.

Referenced by curl_ext_create().

7.3.3.2 `int curl_ext_create (CURL_EXT ** handle)`

Definition at line 203 of file curl_ext.c.

References `_header`, `_CURL_EXT::curl`, `curl_ext_allocateMemory()`, `curl_ext_freeMemory()`, `_CURL_EXT::errorBuffer`, `inputWriter()`, `log_error`, and `outputReader()`.

Referenced by http_init(), and testCurlExt().

7.3.3.3 `void curl_ext_dispose ()`

Definition at line 194 of file curl_ext.c.

References `_header`.

Referenced by http_dispose().

7.3.3.4 void curl_ext_free (CURL_EXT * *handle*)

Definition at line 288 of file curl_ext.c.

References curl_ext_freeMemory().

Referenced by http_dispose().

7.3.3.5 static void curl_ext_freeMemory (CURL_EXT * *handle*) [static]

Definition at line 113 of file curl_ext.c.

References _CURL_EXT::curl, _CURL_EXT::errorBuffer, and _CURL_EXT::inputBuffer.

Referenced by curl_ext_allocateMemory(), curl_ext_create(), and curl_ext_free().

7.3.3.6 int curl_ext_get (CURL_EXT * *handle*, const char * *uri*)

Definition at line 326 of file curl_ext.c.

References _CURL_EXT::curl, log_debug, log_error, and sendRequest().

Referenced by curl_ext_getDOM(), and testCurlExtRequest().

7.3.3.7 int curl_ext_getDOM (CURL_EXT * *handle*, const char * *uri*, IXML_Document ** *response*)

Definition at line 534 of file curl_ext.c.

References curl_ext_get(), and parseXmlInput().

Referenced by http.openConnection(), http_read(), and http_registerDevice().

7.3.3.8 int curl_ext_init (int *defaultInputBufferSize*)

Definition at line 170 of file curl_ext.c.

References _defaultInputBufferSize, _header, and log_error.

Referenced by http_init(), and testCurlExt().

7.3.3.9 int curl_ext_post (CURL_EXT * *handle*, const char * *uri*)

Definition at line 469 of file curl_ext.c.

References `_CURL_EXT::curl`, `log_debug`, `log_error`, `_CURL_EXT::outputBuffer`, `_CURL_EXT::outputPos`, `_CURL_EXT::outputSize`, and `sendRequest()`.

Referenced by `curl_ext_postDOM()`, and `testCurlExtRequest()`.

7.3.3.10 int curl_ext_postDOM (CURL_EXT * *handle*, const char * *uri*, IXML_Document ** *response*)

Definition at line 556 of file `curl_ext.c`.

References `curl_ext_post()`, and `parseXmlInput()`.

Referenced by `addWatchItem()`, `createWatch()`, `http_registerDevice()`, `http_sendBatch()`, `http_unregisterListener()`, `removeWatch()`, and `watchPollTask()`.

7.3.3.11 int curl_ext_put (CURL_EXT * *handle*, const char * *uri*)

Helper function for `curl_ext_put` and `curl_ext_post`.

Both this requests send data to the server in the same manner.

Definition at line 435 of file `curl_ext.c`.

References `_CURL_EXT::curl`, `log_debug`, `log_error`, `_CURL_EXT::outputBuffer`, `_CURL_EXT::outputPos`, `_CURL_EXT::outputSize`, and `sendRequest()`.

Referenced by `curl_ext_putDOM()`, `testCurlExtRequest()`, and `writeValue()`.

7.3.3.12 int curl_ext_putDOM (CURL_EXT * *handle*, const char * *uri*, IXML_Document ** *response*)

Definition at line 545 of file `curl_ext.c`.

References `curl_ext_put()`, and `parseXmlInput()`.

7.3.3.13 static size_t inputWriter (char * *inputData*, size_t *size*, size_t *nmemb*, void * *arg*) [static]

libcurl write callback function.

Called each time when something is received to write down the received data.

Definition at line 46 of file `curl_ext.c`.

References `_defaultInputBufferSize`, `_CURL_EXT::inputBuffer`, `_CURL_EXT::inputBufferFree`, `_CURL_EXT::inputBufferSize`, `log_debug`, `log_error`, and `log_warning`.

Referenced by `curl_ext_create()`.

7.3.3.14 static size_t outputReader (char * *outputData*, size_t *size*, size_t *nmemb*, void * *arg*) [static]

libcurl read callback function.

Called each time when something is sent to read the sending buffer.

Definition at line 87 of file curl_ext.c.

References log_debug, _CURL_EXT::outputBuffer, _CURL_EXT::outputPos, and _CURL_EXT::outputSize.

Referenced by curl_ext_create().

7.3.3.15 static int parseXmlInput (CURL_EXT * *handle*, IXML_Document ** *doc*) [static]

Definition at line 515 of file curl_ext.c.

References _CURL_EXT::inputBuffer, and log_error.

Referenced by curl_ext_getDOM(), curl_ext_postDOM(), and curl_ext_putDOM().

7.3.3.16 static int sendRequest (CURL_EXT * *handle*, const char * *uri*) [static]

Helper function which performs actual HTTP request assumes that type of request was already set by a caller.

Definition at line 297 of file curl_ext.c.

References _CURL_EXT::curl, _CURL_EXT::errorBuffer, _CURL_EXT::inputBuffer, _CURL_EXT::inputBufferFree, _CURL_EXT::inputBufferSize, log_debug, and log_error.

Referenced by curl_ext_get(), curl_ext_post(), and curl_ext_put().

7.3.4 Variable Documentation

7.3.4.1 int _defaultInputBufferSize = DEF_INPUT_BUFFER_SIZE [static]

Definition at line 38 of file curl_ext.c.

Referenced by curl_ext_allocateMemory(), curl_ext_init(), and inputWriter().

7.3.4.2 struct curl_slist* _header [static]

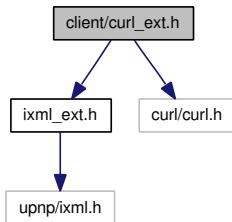
Definition at line 40 of file curl_ext.c.

Referenced by curl_ext_create(), curl_ext_dispose(), and curl_ext_init().

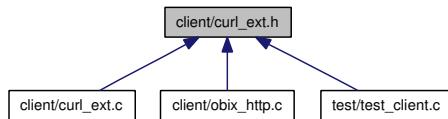
7.4 client/curl_ext.h File Reference

```
#include <ixml_ext.h>
#include <curl/curl.h>
```

Include dependency graph for curl_ext.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [_CURL_EXT](#)

TypeDefs

- [typedef struct _CURL_EXT CURL_EXT](#)

Functions

- [int curl_ext_create \(_CURL_EXT **handle\)](#)
- [void curl_ext_dispose \(\)](#)
- [void curl_ext_free \(_CURL_EXT *handle\)](#)
- [int curl_ext_get \(_CURL_EXT *handle, const char *uri\)](#)
- [int curl_ext_getDOM \(_CURL_EXT *handle, const char *uri, IXML_Document **response\)](#)
- [int curl_ext_init \(int defaultInputBufferSize\)](#)
- [int curl_ext_post \(_CURL_EXT *handle, const char *uri\)](#)
- [int curl_ext_postDOM \(_CURL_EXT *handle, const char *uri, IXML_Document **response\)](#)
- [int curl_ext_put \(_CURL_EXT *handle, const char *uri\)](#)

Helper function for curl_ext_put and curl_ext_post.

- [int curl_ext_putDOM \(_CURL_EXT *handle, const char *uri, IXML_Document **response\)](#)

7.4.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [curl_ext.h](#).

7.4.2 Typedef Documentation

7.4.2.1 `typedef struct _CURL_EXT CURL_EXT`

7.4.3 Function Documentation

7.4.3.1 `int curl_ext_create (CURL_EXT ** handle)`

Definition at line 203 of file curl_ext.c.

References `_header`, `_CURL_EXT::curl`, `curl_ext_allocateMemory()`, `curl_ext_freeMemory()`, `_CURL_EXT::errorBuffer`, `inputWriter()`, `log_error`, and `outputReader()`.

Referenced by `http_init()`, and `testCurlExt()`.

7.4.3.2 `void curl_ext_dispose ()`

Definition at line 194 of file curl_ext.c.

References `_header`.

Referenced by `http_dispose()`.

7.4.3.3 `void curl_ext_free (CURL_EXT * handle)`

Definition at line 288 of file curl_ext.c.

References `curl_ext_freeMemory()`.

Referenced by `http_dispose()`.

7.4.3.4 int curl_ext_get (CURL_EXT * *handle*, const char * *uri*)

Definition at line 326 of file curl_ext.c.

References _CURL_EXT::curl, log_debug, log_error, and sendRequest().

Referenced by curl_ext_getDOM(), and testCurlExtRequest().

7.4.3.5 int curl_ext_getDOM (CURL_EXT * *handle*, const char * *uri*, IXML_Document ** *response*)

Definition at line 534 of file curl_ext.c.

References curl_ext_get(), and parseXmlInput().

Referenced by http.openConnection(), http_read(), and http_registerDevice().

7.4.3.6 int curl_ext_init (int *defaultInputBufferSize*)

Definition at line 170 of file curl_ext.c.

References _defaultInputBufferSize, _header, and log_error.

Referenced by http_init(), and testCurlExt().

7.4.3.7 int curl_ext_post (CURL_EXT * *handle*, const char * *uri*)

Definition at line 469 of file curl_ext.c.

References _CURL_EXT::curl, log_debug, log_error, _CURL_EXT::outputBuffer, _CURL_EXT::outputPos, _CURL_EXT::outputSize, and sendRequest().

Referenced by curl_ext_postDOM(), and testCurlExtRequest().

7.4.3.8 int curl_ext_postDOM (CURL_EXT * *handle*, const char * *uri*, IXML_Document ** *response*)

Definition at line 556 of file curl_ext.c.

References curl_ext_post(), and parseXmlInput().

Referenced by addWatchItem(), createWatch(), http_registerDevice(), http_sendBatch(), http_unregisterListener(), removeWatch(), and watchPollTask().

7.4.3.9 int curl_ext_put (CURL_EXT * *handle*, const char * *uri*)

Helper function for curl_ext_put and curl_ext_post.

Both this requests send data to the server in the same manner.

Definition at line 435 of file curl_ext.c.

References _CURL_EXT::curl, log_debug, log_error, _CURL_EXT::outputBuffer, _CURL_EXT::outputPos, _CURL_EXT::outputSize, and sendRequest().

Referenced by curl_ext_putDOM(), testCurlExtRequest(), and writeValue().

7.4.3.10 int curl_ext_putDOM (CURL_EXT * *handle*, const char * *uri*, IXML_Document ** *response*)

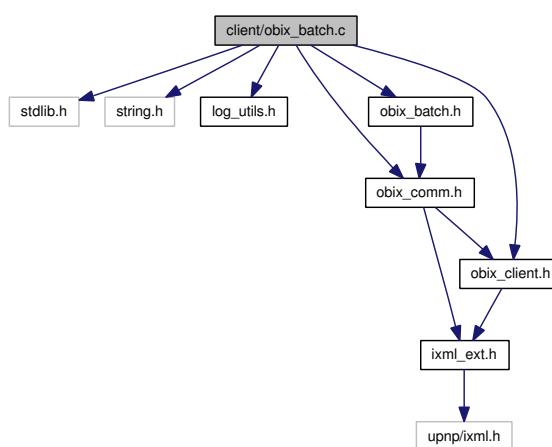
Definition at line 545 of file curl_ext.c.

References curl_ext_put(), and parseXmlInput().

7.5 client/obix_batch.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <log_utils.h>
#include "obix_comm.h"
#include "obix_client.h"
#include "obix_batch.h"
```

Include dependency graph for obix_batch.c:



Defines

- #define OBIX_BATCH_EMPTY_RESULT 1

Functions

- static int `obix_batch_addCommand` (`oBIX_Batch` *batch, `OBIX_BATCH_CMD_TYPE` cmdType, int deviceId, const char *paramUri, const char *newValue, `OBIX_DATA_TYPE` dataType)
 - static void `obix_batch_commandFree` (`oBIX_BatchCmd` *command)
 - `oBIX_Batch` * `obix_batch_create` (int connectionId)
 - Creates a new Batch instance.*
- void `obix_batch_free` (`oBIX_Batch` *batch)
 - Releases memory allocated for the provided Batch object.*
- const `oBIX_BatchResult` * `obix_batch_getResult` (`oBIX_Batch` *batch, int commandId)
 - Returns execution results of the command from specified Batch object.*
- int `obix_batch_read` (`oBIX_Batch` *batch, int deviceId, const char *paramUri)
 - Adds read operation to the provided Batch.*
- int `obix_batch_readValue` (`oBIX_Batch` *batch, int deviceId, const char *paramUri)
 - Adds readValue operation to the provided Batch.*
- int `obix_batch_removeCommand` (`oBIX_Batch` *batch, int commandId)
 - Removes command with specified ID from the Batch object.*
- static void `obix_batch_resultClear` (`oBIX_Batch` *batch, `BOOL` totally)
- static int `obix_batch_resultInit` (`oBIX_Batch` *batch)
- int `obix_batch_send` (`oBIX_Batch` *batch)
 - Sends the Batch object to the obIX server.*
- int `obix_batch_writeValue` (`oBIX_Batch` *batch, int deviceId, const char *paramUri, const char *newValue, `OBIX_DATA_TYPE` dataType)
 - Adds writeValue operation to the provided Batch.*
- static int `strnullcpy` (char **dest, const char *source)

7.5.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file `obix_batch.c`.

7.5.2 Define Documentation

7.5.2.1 `#define OBIX_BATCH_EMPTY_RESULT 1`

Definition at line 36 of file obix_batch.c.

Referenced by `obix_batch_getResult()`, `obix_batch_resultClear()`, and `obix_batch_resultInit()`.

7.5.3 Function Documentation

7.5.3.1 `static int obix_batch_addCommand (oBIX_Batch * batch, OBIX_BATCH_CMD_TYPE cmdType, int deviceId, const char * paramUri, const char * newValue, OBIX_DATA_TYPE dataType) [static]`

Definition at line 155 of file obix_batch.c.

References `_oBIX_Batch::command`, `_oBIX_Batch::commandCounter`, `_oBIX_Batch::connection`, `_oBIX_BatchCmd::dataType`, `_oBIX_BatchCmd::device`, `device_get()`, `_oBIX_BatchCmd::id`, `log_error`, `_oBIX_BatchCmd::newValue`, `_oBIX_BatchCmd::next`, `obix_batch_resultClear()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_ERR_NO_MEMORY`, `OBIX_SUCCESS`, `_oBIX_BatchCmd::paramUri`, `strncpy()`, `TRUE`, and `_oBIX_BatchCmd::type`.

Referenced by `obix_batch_read()`, `obix_batch_readValue()`, and `obix_batch_writeValue()`.

7.5.3.2 `static void obix_batch_commandFree (oBIX_BatchCmd * command) [static]`

Definition at line 38 of file obix_batch.c.

References `_oBIX_BatchCmd::newValue`, and `_oBIX_BatchCmd::paramUri`.

Referenced by `obix_batch_free()`, and `obix_batch_removeCommand()`.

7.5.3.3 `oBIX_Batch* obix_batch_create (int connectionId)`

Creates a new Batch instance.

oBIX Batch contains several operations which can be executed by one request to the server. Commands are executed at the server in the order they were added to the Batch.

Parameters:

`connectionId` ID of the connection for which batch is created.

Returns:

New Batch instance or `NULL` on error.

Definition at line 112 of file obix_batch.c.

References `_oBIX_Batch::command`, `_oBIX_Batch::commandCounter`, `_oBIX_Batch::connection`, `connection_get()`, `log_error`, `OBIX_SUCCESS`, `_oBIX_Batch::result`, and `TRUE`.

Referenced by `resetListener()`, `target_sendUpdate()`, and `testBatch()`.

7.5.3.4 void obix_batch_free (`oBIX_Batch *batch`)

Releases memory allocated for the provided Batch object.

Batch object will become unusable after calling this method.

Parameters:

`batch` Batch object to be deleted.

Definition at line 330 of file obix_batch.c.

References `_oBIX_Batch::command`, `_oBIX_BatchCmd::next`, `obix_batch_commandFree()`, and `_oBIX_Batch::result`.

Referenced by `target_sendUpdate()`, and `testBatch()`.

7.5.3.5 const `oBIX_BatchResult*` obix_batch_getResult (`oBIX_Batch *batch`, `int commandId`)

Returns execution results of the command from specified Batch object.

Parameters:

`batch` Batch object in which the command was executed.

`commandId` ID of the command whose results should be returned.

Returns:

An instance of `oBIX_BatchResult`, containing returned error code of the executed command and other return values if any.

Note:

Do not free the returned values. They will be freed automatically by next execution of `obix_batch_send()` or `obix_batch_free()`.

Definition at line 312 of file obix_batch.c.

References `OBIX_BATCH_EMPTY_RESULT`, `_oBIX_Batch::result`, and `_oBIX_BatchResult::status`.

Referenced by `testBatch()`.

7.5.3.6 int obix_batch_read (`oBIX_Batch *batch`, `int deviceId`, `const char *paramUri`)

Adds read operation to the provided Batch.

When Batch is executed, this operation will act like `obix_read()`. Read object will be stored at the corresponding `oBIX_BatchResult::obj`.

Parameters:

batch Batch object where read operation should be added to.
deviceId ID of the device whose parameter should be read.
paramUri Uri of the parameter which should be read.

Returns:

- >0 - ID of the added command. IDs are assigned according to the order of adding commands to the Batch. Thus ID of the first added command will be 1, ID of the second - 2, and so on.
- <0 - Error code indicating that adding command to the Batch failed. **Note** that this is not the return code of the execution of read command: It will be stored in the corresponding [oBIX_BatchResult::status](#) after the whole Batch is executed.

Note:

Results of the previous execution of the Batch will become unavailable after calling this method.

See also:

[obix_read\(\)](#)

Definition at line 233 of file obix_batch.c.

References [obix_batch_addCommand\(\)](#), OBIX_BATCH_READ, and OBIX_T_BOOL.

Referenced by [testBatch\(\)](#).

7.5.3.7 int obix_batch_readValue (oBIX_Batch * *batch*, int *deviceId*, const char * *paramUri*)

Adds readValue operation to the provided Batch.

When Batch is executed, this operation will act like [obix_readValue\(\)](#). Read value will be stored at the corresponding [oBIX_BatchResult::value](#).

Parameters:

batch Batch object where readValue operation should be added to.
deviceId ID of the device whose parameter should be read.
paramUri Uri of the parameter which should be read.

Returns:

- >0 - ID of the added command. IDs are assigned according to the order of adding commands to the Batch. Thus ID of the first added command will be 1, ID of the second - 2, and so on.
- <0 - Error code indicating that adding command to the Batch failed. **Note** that this is not the return code of the execution of read command: It will be stored in the corresponding [oBIX_BatchResult::status](#) after the whole Batch is executed.

Note:

Results of the previous execution of the Batch will become unavailable after calling this method.

See also:

[obix_readValue\(\)](#)

Definition at line 222 of file obix_batch.c.

References obix_batch_addCommand(), OBIX_BATCH_READ_VALUE, and OBIX_T_BOOL.

Referenced by testBatch().

7.5.3.8 int obix_batch_removeCommand (oBIX_Batch * *batch*, int *commandId*)

Removes command with specified ID from the Batch object.

Parameters:

batch Batch object, from which the command should be removed.

commandId ID of the command which should be removed.

Returns:

- OBIX_SUCCESS if the command is removed successfully;
- OBIX_ERR_INVALID_ARGUMENT if batch is *NULL*;
- OBIX_ERR_INVALID_STATE if no command with specified ID is found in the Batch.

Note:

Results of the previous execution of the Batch will become unavailable after calling this method.

Definition at line 258 of file obix_batch.c.

References _oBIX_Batch::command, _oBIX_BatchCmd::id, _oBIX_BatchCmd::next, obix_batch_commandFree(), obix_batch_resultClear(), OBIX_ERR_INVALID_ARGUMENT, OBIX_ERR_INVALID_STATE, OBIX_SUCCESS, and TRUE.

7.5.3.9 static void obix_batch_resultClear (oBIX_Batch * *batch*, BOOL *totally*) [static]

Definition at line 51 of file obix_batch.c.

References _oBIX_Batch::commandCounter, ixmlelement_freeOwnerDocument(), OBIX_BATCH_EMPTY_RESULT, _oBIX_BatchResult::obj, _oBIX_Batch::result, _oBIX_BatchResult::status, and _oBIX_BatchResult::value.

Referenced by obix_batch_addCommand(), obix_batch_removeCommand(), and obix_batch_resultInit().

7.5.3.10 static int obix_batch_resultInit (oBIX_Batch * *batch*) [static]

Definition at line 84 of file obix_batch.c.

References _oBIX_Batch::commandCounter, FALSE, log_error, OBIX_BATCH_EMPTY_RESULT, obix_batch_resultClear(), OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, _oBIX_Batch::result, and _oBIX_BatchResult::status.

Referenced by obix_batch_send().

7.5.3.11 int obix_batch_send (oBIX_Batch * *batch*)

Sends the Batch object to the oBIX server.

After successful execution, a set of [oBIX_BatchResult](#) objects will be generated containing execution results for each command in Batch. Same Batch object can be sent to the server several times. In that case previous results will be freed and new one will be generated.

Parameters:

batch Batch object to be sent.

Returns:

[OBIX_SUCCESS](#) if the Batch object is sent successfully and server returned no errors. Error code is returned in case if at least one command in batch caused error.

Definition at line 296 of file obix_batch.c.

References `_Connection::comm`, `_oBIX_Batch::connection`, `obix_batch_resultInit()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_SUCCESS`, and `_Comm_Stack::sendBatch`.

Referenced by `resetListener()`, `target_sendUpdate()`, and `testBatch()`.

7.5.3.12 int obix_batch_writeValue (oBIX_Batch * *batch*, int *deviceId*, const char * *paramUri*, const char * *newValue*, OBIX_DATA_TYPE *dataType*)

Adds writeValue operation to the provided Batch.

When Batch is executed, this operation will act like [obix_writeValue\(\)](#).

Parameters:

batch Batch object where writeValue operation should be added to.

deviceId ID of the device whose parameter's value should be written.

paramUri Uri of the parameter which should be written.

newValue Text representation of the new value to be written. It should be a new value for the *val* attribute of the oBIX object on the server, not the whole object.

dataType Type of data which is written to the server.

Returns:

- >0 - ID of the added command. IDs are assigned according to the order of adding commands to the Batch. Thus ID of the first added command will be 1, ID of the second - 2, and so on.
- <0 - Error code indicating that adding command to the Batch failed. **Note** that this is not the return code of the execution of write command: It will be stored in the corresponding [oBIX_BatchResult::status](#) after the whole Batch is executed.

Note:

Only value of an object (*val* attribute) can be written using this method. It's not possible to overwrite a whole oBIX object on the server.

Results of the previous execution of the Batch will become unavailable after calling this method.

See also:

[obix_writeValue\(\)](#)

Definition at line 244 of file obix_batch.c.

References obix_batch_addCommand(), and OBIX_BATCH_WRITE_VALUE.

Referenced by resetListener(), target_sendUpdate(), and testBatch().

7.5.3.13 static int strnulcpy (char ** dest, const char * source) [static]

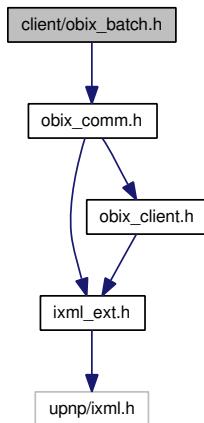
Definition at line 137 of file obix_batch.c.

Referenced by obix_batch_addCommand().

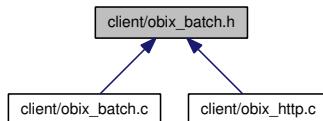
7.6 client/obix_batch.h File Reference

```
#include "obix_comm.h"
```

Include dependency graph for obix_batch.h:



This graph shows which files directly or indirectly include this file:

**Data Structures**

- struct [_oBIX_Batch](#)
- struct [_oBIX_BatchCmd](#)

Typedefs

- `typedef struct _oBIX_BatchCmd oBIX_BatchCmd`

Enumerations

- `enum OBIX_BATCH_CMD_TYPE { OBIX_BATCH_READ_VALUE, OBIX_BATCH_READ, OBIX_BATCH_WRITE_VALUE }`

7.6.1 Detailed Description**Todo**

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [obix_batch.h](#).

7.6.2 Typedef Documentation**7.6.2.1 `typedef struct _oBIX_BatchCmd oBIX_BatchCmd`****7.6.3 Enumeration Type Documentation****7.6.3.1 `enum OBIX_BATCH_CMD_TYPE`****Enumerator:**

OBIX_BATCH_READ_VALUE
OBIX_BATCH_READ
OBIX_BATCH_WRITE_VALUE

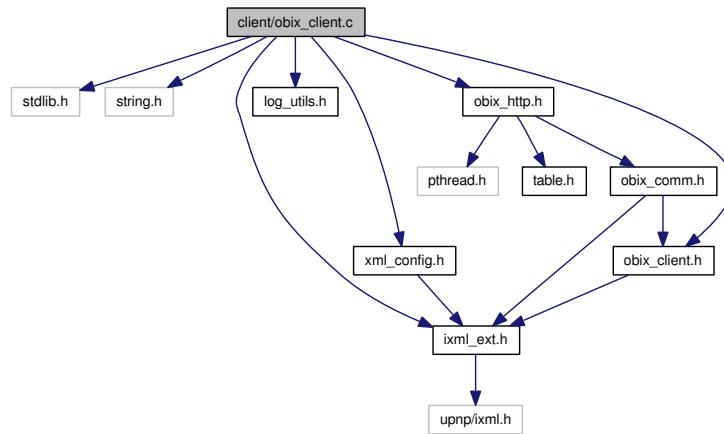
Definition at line 34 of file obix_batch.h.

7.7 client/obix_client.c File Reference

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

```
#include <log_utils.h>
#include <xml_config.h>
#include <obix_http.h>
#include "obix_client.h"
```

Include dependency graph for obix_client.c:



Defines

- #define DEFAULT_MAX_DEVICES 10
- #define DEFAULT_MAX_LISTENERS 10

Functions

- static int connection_create (IXML_Element *connItem)
- static int connection_free (Connection *connection)
- int connection_get (int connectionId, BOOL isConnected, Connection **connection)
- static void device_free (Device *device)
- int device_get (Connection *connection, int deviceId, Device **device)
- static int device_register (Connection *connection, int deviceId, const char *data)
- static int device_unregister (Connection *connection, int deviceId)
- static int device_unregisterAllListeners (Connection *connection, int deviceId)
- static void listener_free (Listener *listener)
- static int listener_register (Connection *connection, Device *device, int listenerId, const char *paramUri, obix_update_listener callback)
- static int listener_unregister (Connection *connection, Device *device, int listenerId)
- int obix_closeConnection (int connectionId)

Closes specified connection releasing all used resources.

- int obix_dispose ()
Releases all resources allocated by library.
- int obix_loadConfig (IXML_Element *config)
Initializes library and loads connection setting from provided DOM structure.

- int **obix_loadConfigFile** (const char *fileName)
Initializes library and loads connection setting from XML file.
- int **obix_openConnection** (int connectionId)
Opens connection to the oBIX server.
- int **obix_read** (int connectionId, int deviceId, const char *paramUri, IXML_Element **output)
Reads the whole oBIX object from the server and returns it as a DOM structure.
- int **obix_readValue** (int connectionId, int deviceId, const char *paramUri, char **output)
Reads value of the specified parameter from the oBIX server.
- int **obix_registerDevice** (int connectionId, const char *obixData)
Posts the provided device information to the oBIX server.
- int **obix_registerListener** (int connectionId, int deviceId, const char *paramUri, **obix_update_listener** listener)
Registers listener for device parameter updates.
- int **obix_unregisterDevice** (int connectionId, int deviceId)
Removes device record from the oBIX server.
- int **obix_unregisterListener** (int connectionId, int deviceId, int listenerId)
Unregisters listener of device parameter updates.
- int **obix_writeValue** (int connectionId, int deviceId, const char *paramUri, const char *newValue, **OBIX_DATA_TYPE** dataType)
Overwrites value of the specified device parameter at the oBIX server.

Variables

- static int **_connectionCount**
- static **Connection** ** **_connections**
Internal storage of connections.
- static const char * **CT_CONNECTION** = "connection"
- static const char * **CT_MAX_DEVICES** = "max-devices"
- static const char * **CT_MAX_LISTENERS** = "max-listeners"
- static const char * **CTA_CONNECTION_ID** = "id"
- static const char * **CTA_CONNECTION_TYPE** = "type"
- static const char * **CTAV_CONNECTION_TYPE_HTTP** = "http"

7.7.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [obix_client.c](#).

7.7.2 Define Documentation

7.7.2.1 #define DEFAULT_MAX_DEVICES 10

Definition at line 37 of file obix_client.c.

Referenced by `connection_create()`.

7.7.2.2 #define DEFAULT_MAX_LISTENERS 10

Definition at line 38 of file obix_client.c.

Referenced by `connection_create()`.

7.7.3 Function Documentation

7.7.3.1 static int connection_create (IXML_Element * *connItem*) [static]

Definition at line 269 of file obix_client.c.

References `_connectionCount`, `_Connection::comm`, `config_getChildTag()`, `config_getTagAttributeValue()`, `config_getTagAttrIntValue()`, `connection_free()`, `CT_CONNECTION`, `CT_MAX_DEVICES`, `CT_MAX_LISTENERS`, `CTA_CONNECTION_ID`, `CTA_CONNECTION_TYPE`, `CTA_VALUE`, `CTAV_CONNECTION_TYPE_HTTP`, `DEFAULT_MAX_DEVICES`, `DEFAULT_MAX_LISTENERS`, `device_register()`, `_Connection::deviceCount`, `_Connection::devices`, `FALSE`, `http_init()`, `_Connection::id`, `_Comm_Stack::initConnection`, `_Connection::isConnected`, `log_error`, `_Connection::maxDevices`, `_Connection::maxListeners`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_ERR_NO_MEMORY`, `OBIX_HTTP`, `OBIX_HTTP_COMM_STACK`, `OBIX_SUCCESS`, `TRUE`, and `_Connection::type`.

Referenced by `obix_loadConfig()`.

7.7.3.2 static int connection_free (Connection * *connection*) [static]

Definition at line 243 of file obix_client.c.

References `_Connection::comm`, `device_free()`, `_Connection::devices`, `_Comm_Stack::freeConnection`, `_Connection::isConnected`, `log_error`, `OBIX_ERR_INVALID_STATE`, and `OBIX_SUCCESS`.

Referenced by `connection_create()`, and `obix_dispose()`.

7.7.3.3 int connection_get (int *connectionId*, BOOL *isConnected*, Connection ** *connection*)

Definition at line 408 of file obix_client.c.

References *_connectionCount*, *_Connection::isConnected*, OBIX_ERR_INVALID_ARGUMENT, OBIX_ERR_INVALID_STATE, and OBIX_SUCCESS.

Referenced by *obix_batch_create()*, *obix_closeConnection()*, *obix_openConnection()*, *obix_read()*, *obix_readValue()*, *obix_registerDevice()*, *obix_registerListener()*, *obix_unregisterDevice()*, *obix_unregisterListener()*, and *obix_writeValue()*.

7.7.3.4 static void device_free (Device * *device*) [static]

Definition at line 125 of file obix_client.c.

References *_Device::listeners*.

Referenced by *connection_free()*, and *device_unregister()*.

7.7.3.5 int device_get (Connection * *connection*, int *deviceId*, Device ** *device*)

Definition at line 227 of file obix_client.c.

References *_Connection::devices*, *_Connection::maxDevices*, OBIX_ERR_INVALID_ARGUMENT, and OBIX_SUCCESS.

Referenced by *obix_batch_addCommand()*, *obix_read()*, *obix_readValue()*, *obix_registerListener()*, *obix_unregisterDevice()*, *obix_unregisterListener()*, and *obix_writeValue()*.

7.7.3.6 static int device_register (Connection * *connection*, int *deviceId*, const char * *data*) [static]

Definition at line 134 of file obix_client.c.

References *_Connection::comm*, *_Connection::deviceCount*, *_Connection::devices*, *_Device::id*, *_Device::listenerCount*, *_Device::listeners*, *log_error*, *_Connection::maxListeners*, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, and *_Comm_Stack::registerDevice*.

Referenced by *connection_create()*, and *obix_registerDevice()*.

7.7.3.7 static int device_unregister (Connection * *connection*, int *deviceId*) [static]

Definition at line 201 of file obix_client.c.

References *_Connection::comm*, *device_free()*, *device_unregisterAllListeners()*, *_Connection::deviceCount*, *_Connection::devices*, OBIX_SUCCESS, and *_Comm_Stack::unregisterDevice*.

Referenced by *obix_closeConnection()*, and *obix_unregisterDevice()*.

**7.7.3.8 static int device_unregisterAllListeners (Connection * *connection*, int *deviceId*)
[static]**

Definition at line 176 of file obix_client.c.

References _Connection::devices, listener_unregister(), _Device::listenerCount, _Device::listeners, _Connection::maxListeners, and OBIX_SUCCESS.

Referenced by device_unregister(), and obix_closeConnection().

7.7.3.9 static void listener_free (Listener * *listener*) [static]

Definition at line 51 of file obix_client.c.

References _Listener::paramUri.

Referenced by listener_register(), and listener_unregister().

7.7.3.10 static int listener_register (Connection * *connection*, Device * *device*, int *listenerId*, const char * *paramUri*, obix_update_listener *callback*) [static]

Definition at line 57 of file obix_client.c.

References _Listener::callback, _Connection::comm, _Listener::connectionId, _Listener::deviceId, _Connection::id, _Device::id, _Listener::id, listener_free(), _Device::listenerCount, _Device::listeners, log_error, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, _Listener::paramUri, and _Comm_Stack::registerListener.

Referenced by obix_registerListener().

**7.7.3.11 static int listener_unregister (Connection * *connection*, Device * *device*, int *listenerId*)
[static]**

Definition at line 101 of file obix_client.c.

References _Connection::comm, _Device::id, listener_free(), _Device::listenerCount, _Device::listeners, and _Comm_Stack::unregisterListener.

Referenced by device_unregisterAllListeners(), and obix_unregisterListener().

7.7.3.12 int obix_closeConnection (int *connectionId*)

Closes specified connection releasing all used resources.

Also unregisters all devices and listeners which were registered using this connection.

Parameters:

connectionId ID of the connection which should be closed.

Returns:

[OBIX_SUCCESS](#) on success, error code otherwise.

Definition at line 534 of file obix_client.c.

References `_Comm_Stack::closeConnection`, `_Connection::comm`, `connection_get()`, `device_unregister()`, `device_unregisterAllListeners()`, `_Connection::deviceCount`, `_Connection::devices`, `FALSE`, `_Connection::isConnected`, `_Connection::maxDevices`, [OBIX_SUCCESS](#), and `TRUE`.

Referenced by `obix_dispose()`.

7.7.3.13 int obix_dispose ()

Releases all resources allocated by library.

All registered listeners and devices are unregistered, all open connections are closed.

Returns:

[OBIX_SUCCESS](#) if operation is completed successfully, error code otherwise.

Definition at line 800 of file obix_client.c.

References `_connectionCount`, `connection_free()`, `http_dispose()`, `obix_closeConnection()`, and [OBIX_SUCCESS](#).

Referenced by `main()`, `obix_loadConfig()`, and `testConnectionAndDevices()`.

7.7.3.14 int obix_loadConfig (IXML_Element * *config*)

Initializes library and loads connection setting from provided DOM structure.

Unlike `obix_loadConfigFile()` it doesn't configure log system of the library. It can be configured manually using `config_log()` or `log_utils.h` functions. By default, all messages (including debug ones) are written to `stdout`. The format of the configuration file can be found at `example_timer_config.xml`

Parameters:

config DOM structure representing a configuration XML.

Returns:

[OBIX_SUCCESS](#) if the library is initialized successfully, error code otherwise.

Definition at line 457 of file obix_client.c.

References `_connectionCount`, `connection_create()`, `CT_CONNECTION`, `ixmlNode_convertToElement()`, `log_error`, `obix_dispose()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_ERR_NO_MEMORY`, and [OBIX_SUCCESS](#).

Referenced by `loadSettings()`, and `obix_loadConfigFile()`.

7.7.3.15 int obix_loadConfigFile (const char **fileName*)

Initializes library and loads connection setting from XML file.

Also sets up the logging system of the library. The format of the configuration file can be found at example_timer_config.xml

Parameters:

fileName Name of the configuration file.

Returns:

OBIX_SUCCESS if the library initialized successfully, error code otherwise.

Definition at line 427 of file obix_client.c.

References config_finishInit(), config_loadFile(), config_log(), FALSE, OBIX_ERR_INVALID_ARGUMENT, obix_loadConfig(), OBIX_SUCCESS, and TRUE.

Referenced by main(), and testObixLoadConfigFile().

7.7.3.16 int obix_openConnection (int *connectionId*)

Opens connection to the oBIX server.

Parameters:

connectionId Connection ID which was specified in the loaded configuration file.

Returns:

OBIX_SUCCESS if connection is opened successfully, error code otherwise.

Definition at line 515 of file obix_client.c.

References _Connection::comm, connection_get(), FALSE, _Connection::isConnected, OBIX_SUCCESS, _Comm_Stack::openConnection, and TRUE.

Referenced by main(), and testConnectionAndDevices().

7.7.3.17 int obix_read (int *connectionId*, int *deviceId*, const char **paramUri*, IXML_Element *output*)**

Reads the whole oBIX object from the server and returns it as a DOM structure.

This function can be used to read any object on the server. It can be some object which was registered either by the same client, or by any other device driver. It can be also any server's own object.

Example: Server contains the following information:

```
<obj name="device1" href="/obix/device1/">
  <obj name="configure" href="conf/" />
```

```

<str name="name" href="name" val="My Device"/>
</obj>
</obj>
```

There are two options to read the "*configure*" object of "*device1*":

- If this device was registered by the same client earlier, than the client should use device ID assigned to this device and provide parameter URI relative to the root of device data (in current example it is "*conf*").
- In case if that device was registered by someone else, than *0* should be used instead device ID + full URI of the required object (in this example it is "*/obix/device1/conf*").

If the whole object "*device1*" should be read and it was previously published by the same client, then the client should provide device ID assigned to this device and *NULL* as *paramUri*.

Note:

Although there is no need for this function in the normal workflow (see [Usage](#)) it still can be used, for instance, during initialization phase for obtaining some data from the server. Usage of this function for periodical reading of some object is not efficient and should be avoided. Use [obix_registerListener](#) instead.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device whose data should be read or *0* if the object doesn't belong to devices registered by this client.

paramUri URI of the object. It should be either relative to the device record like it was provided during device registration, or relative to the server root if the object doesn't belong to devices registered by this client.

output If read command executed successfully the DOM representation of the read object is stored here.

Returns:

[OBIX_SUCCESS](#) on success, negative error code otherwise.

Definition at line 735 of file obix_client.c.

References `_Connection::comm`, `connection_get()`, `device_get()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_SUCCESS`, `_Comm_Stack::read`, and `TRUE`.

Referenced by `checkEventsTask()`.

7.7.3.18 `int obix_readValue (int connectionId, int deviceId, const char * paramUri, char ** output)`

Reads value of the specified parameter from the oBIX server.

This function can be used to read value of any object on the server: It can be some parameter of device which was registered either by the same client, or by any other device driver. It can be also any value of the server settings.

Example: Server contains the following information:

```
<obj name="device1" href="/obix/device1/">
  <obj name="configure" href="conf/" />
    <str name="name" href="name" val="My Device"/>
  </obj>
</obj>
```

There are two ways to read the value of "name" string of "device1":

- If this device was registered by the same client earlier, than the client should use device ID assigned to this device and provide parameter URI relative to the root of device data (in current example it is "conf/name").
- In case if that device was registered by someone else, than 0 should be used instead device ID + full URI of the required parameter (in this example it is "/obix/device1/conf/name").

Note:

This method is used to read only *val* attribute of some object, but not the whole object itself. If you need to read the whole oBIX object then use [obix_read](#) instead.

Although there is no need for this function in the normal workflow (see [Usage](#)), it still can be used, for instance, during initialization phase for obtaining some data from the server. Usage of this function for periodical reading of some object is not efficient and should be avoided. Use [obix_registerListener](#) instead.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device whose parameter should be read or 0 if the parameter doesn't belong to devices registered by this client.

paramUri URI of the parameter. It should be either relative to the device record like it was provided during device registration, or relative to the server root if the parameter doesn't belong to devices registered by this client.

output If read command executed successfully the attribute's value is stored here.

Returns:

[OBIX_SUCCESS](#) on success, negative error code otherwise.

Definition at line 702 of file obix_client.c.

References [_Connection::comm](#), [connection_get\(\)](#), [device_get\(\)](#), [OBIX_ERR_INVALID_ARGUMENT](#), [OBIX_SUCCESS](#), [_Comm_Stack::readValue](#), and [TRUE](#).

7.7.3.19 int obix_registerDevice (int *connectionId*, const char * *obixData*)

Posts the provided device information to the oBIX server.

Note:

Input data is not tested to conform with oBIX specification. Is is strongly recommended to provide *displayName* attribute to every object. Also attributes *href* and *writable* are obligatory for all device parameters which are going to be changed by the device driver or external oBIX server users:

- *writable* attribute should be set to *true*.

- *href* attribute of the parent object should be a valid URI, relative to the server root (start with "/").
- *href* attributes of all child objects should have a valid URIs, relative to the parent object.

Example:

```
<obj name="light" href="/kitchen/light/" displayName="Kitchen lights">
  <bool name="switch"
    href="switch/"
    displayName="Switch"
    is="obix:bool"
    val="true"
    writable="true"/>
</obj>
```

Note:

Parent object should specify *href* attribute but the oBIX server is free to modify it (for instance, add prefix of the device storage), thus the URI can't be used to refer to the device record. Use the assigned device ID instead.

Parameters:

- connectionId** ID of the connection which should be used.
obixData oBIX object representing a new device.

Returns:

- >0 ID of the created device record;
- <0 error code.

Definition at line 576 of file obix_client.c.

References `connection_get()`, `device_register()`, `_Connection::deviceCount`, `_Connection::devices`, `log_error`, `_Connection::maxDevices`, `OBIX_ERR_LIMIT_REACHED`, `OBIX_ERR_UNKNOWN_BUG`, `OBIX_SUCCESS`, and `TRUE`.

Referenced by `main()`, and `testConnectionAndDevices()`.

7.7.3.20 int obix_registerListener (int *connectionId*, int *deviceId*, const char * *paramUri*, obix_update_listener *listener*)

Registers listener for device parameter updates.

Overwrites existing listener if it is called twice for the same parameter.

This method can be also used to subscribe for the updates of any other objects stored at the oBIX server. In that case *0* should be provided as *deviceId* and *paramUri* should be relative to the server root.

Parameters:

- connectionId** ID of the connection which should be used.
deviceId ID of the device whose parameter should be monitored or *0* if the parameter doesn't belong to devices registered by this client.
paramUri URI of the parameter which should be monitored. It should be either relative to the device record like it was provided during device registration, or relative to the server root if the parameter doesn't belong to devices registered by this client.

listener Pointer to the listener function which would be invoked every time when the subscribed parameter is changed.

Note:

listener method should be quick. Slow listener (especially if it waits for some resource) will block subsequent calls to all listeners.

Returns:

- ≥ 0 ID of the created listener;
- < 0 error code.

Definition at line 627 of file obix_client.c.

References connection_get(), device_get(), listener_register(), _Device::listenerCount, _Device::listeners, log_error, _Connection::maxListeners, OBIX_ERR_LIMIT_REACHED, OBIX_ERR_UNKNOWN_BUG, OBIX_SUCCESS, and TRUE.

Referenced by main().

7.7.3.21 int obix_unregisterDevice (int *connectionId*, int *deviceId*)

Removes device record from the oBIX server.

Also removes all listeners which were registered for this device.

Parameters:

- connectionId** ID of the connection which should be used.
deviceId ID of the device which should be removed.

Returns:

OBIX_SUCCESS if the device record is removed successfully, error code otherwise.

Definition at line 609 of file obix_client.c.

References connection_get(), device_get(), device_unregister(), OBIX_SUCCESS, and TRUE.

Referenced by testConnectionAndDevices().

7.7.3.22 int obix_unregisterListener (int *connectionId*, int *deviceId*, int *listenerId*)

Unregisters listener of device parameter updates.

Parameters:

- connectionId** ID of the connection which should be used.
deviceId ID of the device whose parameter is now monitored or 0 if the parameter doesn't belong to devices registered by this client.
listenerId ID of listener to be removed.

Returns:

[OBIX_SUCCESS](#) if the listener is removed successfully, error code otherwise.

Definition at line 671 of file obix_client.c.

References `connection_get()`, `device_get()`, `listener_unregister()`, `_Device::listeners`, `_Connection::maxListeners`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_SUCCESS`, and `TRUE`.

Referenced by `main()`.

7.7.3.23 int obix_writeValue (int *connectionId*, int *deviceId*, const char * *paramUri*, const char * *newValue*, OBIX_DATA_TYPE *dataType*)

Overwrites value of the specified device parameter at the oBIX server.

This function can be also used to change a value of any *writable* object at the oBIX server.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device whose parameter should be changed or *0* if the parameter doesn't belong to devices registered by this client.

paramUri URI of the parameter. It should be either relative to the device record like it was provided during device registration, or relative to the server root if changing parameter doesn't belong to devices registered by this client.

newValue Text representation of the new value to be written. It should be a new value for the *val* attribute of the oBIX object on the server, not the whole object.

Note:

Only value of an object (*val* attribute) can be written using this method. It's not possible to overwrite a whole oBIX object on the server.

Parameters:

dataType Type of data which is written to the server.

Returns:

[OBIX_SUCCESS](#) on success, negative error code otherwise.

See also:

[obix_readValue\(\)](#) for the usage example.

Definition at line 767 of file obix_client.c.

References `_Connection::comm`, `connection_get()`, `device_get()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_SUCCESS`, `TRUE`, and `_Comm_Stack::writeValue`.

Referenced by `sendFallEventUpdate()`, and `timerTask()`.

7.7.4 Variable Documentation

7.7.4.1 int _connectionCount [static]

Definition at line 49 of file obix_client.c.

Referenced by connection_create(), connection_get(), obix_dispose(), and obix_loadConfig().

7.7.4.2 Connection** _connections [static]

Internal storage of connections.

Definition at line 48 of file obix_client.c.

7.7.4.3 const char* CT_CONNECTION = "connection" [static]

Definition at line 40 of file obix_client.c.

Referenced by connection_create(), and obix_loadConfig().

7.7.4.4 const char* CT_MAX_DEVICES = "max-devices" [static]

Definition at line 44 of file obix_client.c.

Referenced by connection_create().

7.7.4.5 const char* CT_MAX_LISTENERS = "max-listeners" [static]

Definition at line 45 of file obix_client.c.

Referenced by connection_create().

7.7.4.6 const char* CTA_CONNECTION_ID = "id" [static]

Definition at line 41 of file obix_client.c.

Referenced by connection_create().

7.7.4.7 const char* CTA_CONNECTION_TYPE = "type" [static]

Definition at line 42 of file obix_client.c.

Referenced by connection_create().

7.7.4.8 const char* CTAV_CONNECTION_TYPE_HTTP = "http" [static]

Definition at line 43 of file obix_client.c.

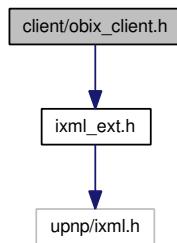
Referenced by connection_create().

7.8 client/obix_client.h File Reference

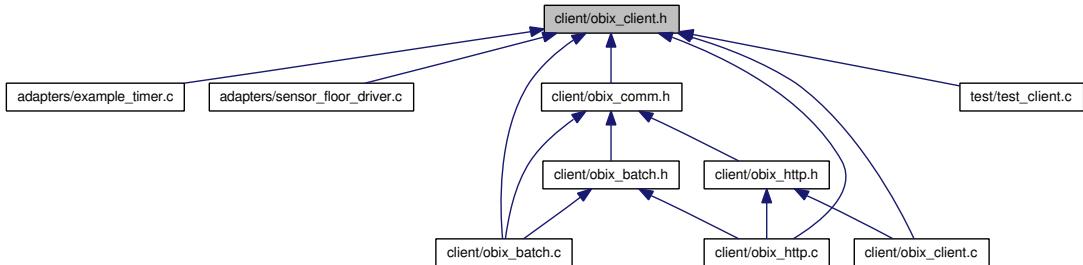
oBIX Client API.

```
#include <ixml_ext.h>
```

Include dependency graph for obix_client.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [_oBIX_BatchResult](#)

Contains outputs of the command, which was executed in a [Batch](#).

TypeDefs

- typedef struct [_oBIX_Batch](#) [oBIX_Batch](#)

Represents a Batch object.

- typedef struct [_oBIX_BatchResult](#) [oBIX_BatchResult](#)

Contains outputs of the command, which was executed in a [Batch](#).

- `typedef int(* obix_update_listener)(int connectionId, int deviceId, int listenerId, const char *newValue)`

Callback function, which is invoked when subscribed value is changed at the oBIX server.

Enumerations

- `enum OBIX_DATA_TYPE {
 OBIX_T_BOOL, OBIX_T_INT, OBIX_T_REAL, OBIX_T_STR,
 OBIX_T_ENUM, OBIX_T_ABSTIME, OBIX_T_RELTIME, OBIX_T_URI }`

Standard oBIX data types.
- `enum OBIX_ERRORCODE {
 OBIX_SUCCESS = 0, OBIX_ERR_INVALID_ARGUMENT = -1, OBIX_ERR_NO_MEMORY = -2, OBIX_ERR_INVALID_STATE = -3,
 OBIX_ERR_LIMIT_REACHED = -4, OBIX_ERR_BAD_CONNECTION = -5, OBIX_ERR_UNKNOWN_BUG = -100, OBIX_ERR_HTTP_LIB = -6,
 OBIX_ERR_SERVER_ERROR = -7 }`

Error codes which are returned by library functions.

Functions

- `int obix_closeConnection (int connectionId)`

Closes specified connection releasing all used resources.
- `int obix_dispose ()`

Releases all resources allocated by library.
- `int obix_loadConfig (IXML_Element *config)`

Initializes library and loads connection setting from provided DOM structure.
- `int obix_loadConfigFile (const char *fileName)`

Initializes library and loads connection setting from XML file.
- `int obix_openConnection (int connectionId)`

Opens connection to the oBIX server.
- `int obix_read (int connectionId, int deviceId, const char *paramUri, IXML_Element **output)`

Reads the whole oBIX object from the server and returns it as a DOM structure.
- `int obix_readValue (int connectionId, int deviceId, const char *paramUri, char **output)`

Reads value of the specified parameter from the oBIX server.
- `int obix_registerDevice (int connectionId, const char *obixData)`

Posts the provided device information to the oBIX server.
- `int obix_registerListener (int connectionId, int deviceId, const char *paramUri, obix_update_listener listener)`

Registers listener for device parameter updates.

- int [obix_unregisterDevice](#) (int connectionId, int deviceId)
Removes device record from the oBIX server.
- int [obix_unregisterListener](#) (int connectionId, int deviceId, int listenerId)
Unregisters listener of device parameter updates.
- int [obix_writeValue](#) (int connectionId, int deviceId, const char *paramUri, const char *newValue, [OBIX_DATA_TYPE](#) dataType)
Overwrites value of the specified device parameter at the oBIX server.

oBIX Batch operations

- [oBIX_Batch * obix_batch_create](#) (int connectionId)
Creates a new Batch instance.
- void [obix_batch_free](#) ([oBIX_Batch](#) *batch)
Releases memory allocated for the provided Batch object.
- const [oBIX_BatchResult * obix_batch_getResult](#) ([oBIX_Batch](#) *batch, int commandId)
Returns execution results of the command from specified Batch object.
- int [obix_batch_read](#) ([oBIX_Batch](#) *batch, int deviceId, const char *paramUri)
Adds read operation to the provided Batch.
- int [obix_batch_readValue](#) ([oBIX_Batch](#) *batch, int deviceId, const char *paramUri)
Adds readValue operation to the provided Batch.
- int [obix_batch_removeCommand](#) ([oBIX_Batch](#) *batch, int commandId)
Removes command with specified ID from the Batch object.
- int [obix_batch_send](#) ([oBIX_Batch](#) *batch)
Sends the Batch object to the oBIX server.
- int [obix_batch_writeValue](#) ([oBIX_Batch](#) *batch, int deviceId, const char *paramUri, const char *newValue, [OBIX_DATA_TYPE](#) dataType)
Adds writeValue operation to the provided Batch.

7.8.1 Detailed Description

oBIX Client API.

oBIX Client API simplifies implementing oBIX client applications, such as device drivers. Library hides all network calls and allows accessing data at the oBIX server without dealing with oBIX request formats. There is also a possibility to subscribe for data updates on the server, which is performed by library using oBIX Watch engine.

7.8.2 Usage

The typical usage of the library in device adapter (see example at [example_timer.c](#)):

- Include `obix_client.h` header.
- Initialize library during program startup. It can be done either by calling `obix_loadConfigFile()` which will load settings from configuration file, or by `obix_loadConfig()` with own generated XML settings structure.
- Open configured connection to oBIX server by calling `obix_openConnection()`.
- Generate an oBIX object for each device (for instance, one adapter can handle several devices of the same type) and register them at the server by calling `obix_registerDevice()`.
- If oBIX object, generated for the device, contains controlling values which can be changed outside (e.g. enabling/disabling boolean switch), register listener for these values by calling `obix_registerListener()`. Library starts polling changes of subscribed values and calls corresponding `obix_update_listener()` every time when receives a new value.
- When some update of state variable is received by the driver from device, post the new value to the oBIX server by calling `obix_writeValue()`.
- Call `obix_unregisterDevice()` when driver detects that the device is not available any more (unplugged, connection broken, etc.).
- When device driver is going down call `obix_dispose()` in order to close all connections and release resources reserved for communication with oBIX server(s).

Note:

In order to register a new device (using `obix_registerDevice()`), server should support *signUp* feature which is not in the oBIX specification. Currently *signUp* operation is supported by C oBIX Server included into this distribution and oFMS (<http://www.stok.fi/eng/ofms/index.html>). All other functions should work with any proper oBIX server implementation. If not, please report the found error to the author of this distribution.

Author:

Andrey Litvinov

Version:

0.0.0

Definition in file `obix_client.h`.

7.8.3 Typedef Documentation

7.8.3.1 `typedef struct _oBIX_Batch oBIX_Batch`

Represents a Batch object.

oBIX Batch allows combining several commands in one request to the server, thus reducing response time and network load.

General Usage:

- Create new Batch instance using [obix_batch_create\(\)](#);
- Add commands to batch using [obix_batch_read\(\)](#), [obix_batch_readValue\(\)](#) or [obix_batch_writeValue\(\)](#);
- Send Batch object to the server by calling [obix_batch_send\(\)](#);
- An instance of [oBIX_BatchResult](#) will be generated for each command in Batch, containing execution results. These results can be obtained using [obix_batch_getResult\(\)](#);
- Free Batch object with [obix_batch_free\(\)](#).

Definition at line 461 of file obix_client.h.

7.8.3.2 [typedef struct _oBIX_BatchResult oBIX_BatchResult](#)

Contains outputs of the command, which was executed in a [Batch](#).

7.8.3.3 [typedef int\(* obix_update_listener\)\(int connectionId, int deviceId, int listenerId, const char *newValue\)](#)

Callback function, which is invoked when subscribed value is changed at the oBIX server.

ID arguments of the listener can be used to define which parameter was updated in case when one function is registered to handle updates of several parameters.

See also:

[obix_registerListener\(\)](#)

Parameters:

connectionId ID of the connection from which the update is received.

deviceId ID of the device whose parameter was changed. If the parameter doesn't belong to any device which was registered by current client, than 0 will be passed.

listenerId ID of the listener which receives the event.

newValue New value of the parameter.

Returns:

The listener should return [OBIX_SUCCESS](#) if the event was handled properly. Any other returned value will be considered by library as an error.

Definition at line 155 of file obix_client.h.

7.8.4 Enumeration Type Documentation

7.8.4.1 [enum OBIX_DATA_TYPE](#)

Standard oBIX data types.

Used in [obix_writeValue\(\)](#).

Enumerator:

OBIX_T_BOOL Boolean data type (bool).

Possible values: "true" or "false".

OBIX_T_INT Integer data type (int).

Possible values are defined by *xs:long*.

OBIX_T_REAL Real data type (real).

Possible values are defined by *xs:double*.

OBIX_T_STR String data type (str).

OBIX_T_ENUM Enumeration data type (enum).

Possible values are defined by associated *range* object.

OBIX_T_ABSTIME Time data type (abstime).

Represents an absolute point in time. Value format is defined by *xs:dateTime*.

OBIX_T_RELTIME Time data type (reltime).

Represents time interval. Value format is defined by *xs:duration*.

OBIX_T_URI URI data type (uri).

Almost like a string, but contains valid URI.

Definition at line 112 of file obix_client.h.

7.8.4.2 enum OBIX_ERRORCODE

Error codes which are returned by library functions.

Enumerator:

OBIX_SUCCESS Operation is completed successfully.

OBIX_ERR_INVALID_ARGUMENT Function received wrong input argument.

OBIX_ERR_NO_MEMORY Not enough memory to complete the operation.

OBIX_ERR_INVALID_STATE Library has invalid state.

OBIX_ERR_LIMIT_REACHED Allocated buffer for devices or listeners is full.

[Todo](#)

Remove this error and enlarge arrays when needed.

OBIX_ERR_BAD_CONNECTION Error in communication with server.

OBIX_ERR_UNKNOWN_BUG Reserved for uncaught errors.

If such error occurs, this is a bug.

OBIX_ERR_HTTP_LIB Error inside HTTP communication module.

OBIX_ERR_SERVER_ERROR oBIX server returned an error object.

Definition at line 86 of file obix_client.h.

7.8.5 Function Documentation

7.8.5.1 `oBIX_Batch* obix_batch_create (int connectionId)`

Creates a new Batch instance.

oBIX Batch contains several operations which can be executed by one request to the server. Commands are executed at the server in the order they were added to the Batch.

Parameters:

connectionId ID of the connection for which batch is created.

Returns:

New Batch instance or *NULL* on error.

Definition at line 112 of file obix_batch.c.

References `_oBIX_Batch::command`, `_oBIX_Batch::commandCounter`, `_oBIX_Batch::connection`, `connection_get()`, `log_error`, `OBIX_SUCCESS`, `_oBIX_Batch::result`, and `TRUE`.

Referenced by `resetListener()`, `target_sendUpdate()`, and `testBatch()`.

7.8.5.2 `void obix_batch_free (oBIX_Batch *batch)`

Releases memory allocated for the provided Batch object.

Batch object will become unusable after calling this method.

Parameters:

batch Batch object to be deleted.

Definition at line 330 of file obix_batch.c.

References `_oBIX_Batch::command`, `_oBIX_BatchCmd::next`, `obix_batch_commandFree()`, and `_oBIX_Batch::result`.

Referenced by `target_sendUpdate()`, and `testBatch()`.

7.8.5.3 `const oBIX_BatchResult* obix_batch_getResult (oBIX_Batch *batch, int commandId)`

Returns execution results of the command from specified Batch object.

Parameters:

batch Batch object in which the command was executed.

commandId ID of the command whose results should be returned.

Returns:

An instance of `oBIX_BatchResult`, containing returned error code of the executed command and other return values if any.

Note:

Do not free the returned values. They will be freed automatically by next execution of [obix_batch_send\(\)](#) or [obix_batch_free\(\)](#).

Definition at line 312 of file obix_batch.c.

References OBIX_BATCH_EMPTY_RESULT, _oBIX_Batch::result, and _oBIX_BatchResult::status.

Referenced by testBatch().

7.8.5.4 int obix_batch_read (oBIX_Batch * *batch*, int *deviceId*, const char * *paramUri*)

Adds read operation to the provided Batch.

When Batch is executed, this operation will act like [obix_read\(\)](#). Read object will be stored at the corresponding [oBIX_BatchResult::obj](#).

Parameters:

batch Batch object where read operation should be added to.

deviceId ID of the device whose parameter should be read.

paramUri Uri of the parameter which should be read.

Returns:

- >0 - ID of the added command. IDs are assigned according to the order of adding commands to the Batch. Thus ID of the first added command will be 1, ID of the second - 2, and so on.
- <0 - Error code indicating that adding command to the Batch failed. **Note** that this is not the return code of the execution of read command: It will be stored in the corresponding [oBIX_BatchResult::status](#) after the whole Batch is executed.

Note:

Results of the previous execution of the Batch will become unavailable after calling this method.

See also:

[obix_read\(\)](#)

Definition at line 233 of file obix_batch.c.

References obix_batch_addCommand(), OBIX_BATCH_READ, and OBIX_T_BOOL.

Referenced by testBatch().

7.8.5.5 int obix_batch_readValue (oBIX_Batch * *batch*, int *deviceId*, const char * *paramUri*)

Adds readValue operation to the provided Batch.

When Batch is executed, this operation will act like [obix_readValue\(\)](#). Read value will be stored at the corresponding [oBIX_BatchResult::value](#).

Parameters:

batch Batch object where readValue operation should be added to.

deviceId ID of the device whose parameter should be read.

paramUri Uri of the parameter which should be read.

Returns:

- >0 - ID of the added command. IDs are assigned according to the order of adding commands to the Batch. Thus ID of the first added command will be 1, ID of the second - 2, and so on.
- <0 - Error code indicating that adding command to the Batch failed. **Note** that this is not the return code of the execution of read command: It will be stored in the corresponding [oBIX_BatchResult::status](#) after the whole Batch is executed.

Note:

Results of the previous execution of the Batch will become unavailable after calling this method.

See also:

[obix_readValue\(\)](#)

Definition at line 222 of file obix_batch.c.

References [obix_batch_addCommand\(\)](#), [OBIX_BATCH_READ_VALUE](#), and [OBIX_T_BOOL](#).

Referenced by [testBatch\(\)](#).

7.8.5.6 int obix_batch_removeCommand ([oBIX_Batch](#) * *batch*, int *commandId*)

Removes command with specified ID from the Batch object.

Parameters:

batch Batch object, from which the command should be removed.

commandId ID of the command which should be removed.

Returns:

- [OBIX_SUCCESS](#) if the command is removed successfully;
- [OBIX_ERR_INVALID_ARGUMENT](#) if batch is *NULL*;
- [OBIX_ERR_INVALID_STATE](#) if no command with specified ID is found in the Batch.

Note:

Results of the previous execution of the Batch will become unavailable after calling this method.

Definition at line 258 of file obix_batch.c.

References [_oBIX_Batch::command](#), [_oBIX_BatchCmd::id](#), [_oBIX_BatchCmd::next](#), [obix_batch_commandFree\(\)](#), [obix_batch_resultClear\(\)](#), [OBIX_ERR_INVALID_ARGUMENT](#), [OBIX_ERR_INVALID_STATE](#), [OBIX_SUCCESS](#), and [TRUE](#).

7.8.5.7 int obix_batch_send (oBIX_Batch * batch)

Sends the Batch object to the oBIX server.

After successful execution, a set of [oBIX_BatchResult](#) objects will be generated containing execution results for each command in Batch. Same Batch object can be sent to the server several times. In that case previous results will be freed and new one will be generated.

Parameters:

batch Batch object to be sent.

Returns:

[OBIX_SUCCESS](#) if the Batch object is sent successfully and server returned no errors. Error code is returned in case if at least one command in batch caused error.

Definition at line 296 of file obix_batch.c.

References `_Connection::comm`, `_oBIX_Batch::connection`, `obix_batch_resultInit()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_SUCCESS`, and `_Comm_Stack::sendBatch`.

Referenced by `resetListener()`, `target_sendUpdate()`, and `testBatch()`.

7.8.5.8 int obix_batch_writeValue (oBIX_Batch * batch, int deviceId, const char * paramUri, const char * newValue, OBIX_DATA_TYPE dataType)

Adds writeValue operation to the provided Batch.

When Batch is executed, this operation will act like [obix_writeValue\(\)](#).

Parameters:

batch Batch object where writeValue operation should be added to.

deviceId ID of the device whose parameter's value should be written.

paramUri Uri of the parameter which should be written.

newValue Text representation of the new value to be written. It should be a new value for the *val* attribute of the oBIX object on the server, not the whole object.

dataType Type of data which is written to the server.

Returns:

- >0 - ID of the added command. IDs are assigned according to the order of adding commands to the Batch. Thus ID of the first added command will be 1, ID of the second - 2, and so on.
- <0 - Error code indicating that adding command to the Batch failed. **Note** that this is not the return code of the execution of write command: It will be stored in the corresponding [oBIX_BatchResult::status](#) after the whole Batch is executed.

Note:

Only value of an object (*val* attribute) can be written using this method. It's not possible to overwrite a whole oBIX object on the server.

Results of the previous execution of the Batch will become unavailable after calling this method.

See also:

[obix_writeValue\(\)](#)

Definition at line 244 of file obix_batch.c.

References obix_batch_addCommand(), and OBIX_BATCH_WRITE_VALUE.

Referenced by resetListener(), target_sendUpdate(), and testBatch().

7.8.5.9 int obix_closeConnection (int *connectionId*)

Closes specified connection releasing all used resources.

Also unregisters all devices and listeners which were registered using this connection.

Parameters:

connectionId ID of the connection which should be closed.

Returns:

[OBIX_SUCCESS](#) on success, error code otherwise.

Definition at line 534 of file obix_client.c.

References _Comm_Stack::closeConnection, _Connection::comm, connection_get(), device_unregister(), device_unregisterAllListeners(), _Connection::deviceCount, _Connection::devices, FALSE, _Connection::isConnected, _Connection::maxDevices, OBIX_SUCCESS, and TRUE.

Referenced by obix_dispose().

7.8.5.10 int obix_dispose ()

Releases all resources allocated by library.

All registered listeners and devices are unregistered, all open connections are closed.

Returns:

[OBIX_SUCCESS](#) if operation is completed successfully, error code otherwise.

Definition at line 800 of file obix_client.c.

References _connectionCount, connection_free(), http_dispose(), obix_closeConnection(), and OBIX_SUCCESS.

Referenced by main(), obix_loadConfig(), and testConnectionAndDevices().

7.8.5.11 int obix_loadConfig (IXML_Element * *config*)

Initializes library and loads connection setting from provided DOM structure.

Unlike [obix_loadConfigFile\(\)](#) it doesn't configure log system of the library. It can be configured manually using [config_log\(\)](#) or [log_utils.h](#) functions. By default, all messages (including debug ones) are written to *stdout*. The format of the configuration file can be found at `example_timer_config.xml`

Parameters:

config DOM structure representing a configuration XML.

Returns:

[OBIX_SUCCESS](#) if the library is initialized successfully, error code otherwise.

Definition at line 457 of file `obix_client.c`.

References `_connectionCount`, `connection_create()`, `CT_CONNECTION`, `ixmlNode_convertToElement()`, `log_error`, `obix_dispose()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_ERR_NO_MEMORY`, and `OBIX_SUCCESS`.

Referenced by `loadSettings()`, and `obix_loadConfigFile()`.

7.8.5.12 int obix_loadConfigFile (const char **fileName*)

Initializes library and loads connection setting from XML file.

Also sets up the logging system of the library. The format of the configuration file can be found at `example_timer_config.xml`

Parameters:

fileName Name of the configuration file.

Returns:

[OBIX_SUCCESS](#) if the library initialized successfully, error code otherwise.

Definition at line 427 of file `obix_client.c`.

References `config_finishInit()`, `config_loadFile()`, `config_log()`, `FALSE`, `OBIX_ERR_INVALID_ARGUMENT`, `obix_loadConfig()`, `OBIX_SUCCESS`, and `TRUE`.

Referenced by `main()`, and `testObixLoadConfigFile()`.

7.8.5.13 int obix_openConnection (int *connectionId*)

Opens connection to the oBIX server.

Parameters:

connectionId Connection ID which was specified in the loaded configuration file.

Returns:

[OBIX_SUCCESS](#) if connection is opened successfully, error code otherwise.

Definition at line 515 of file obix_client.c.

References `_Connection::comm`, `connection_get()`, `FALSE`, `_Connection::isConnected`, `OBIX_SUCCESS`, `_Comm_Stack::openConnection`, and `TRUE`.

Referenced by `main()`, and `testConnectionAndDevices()`.

7.8.5.14 int obix_read (int *connectionId*, int *deviceId*, const char **paramUri*, IXML_Element *output*)**

Reads the whole oBIX object from the server and returns it as a DOM structure.

This function can be used to read any object on the server. It can be some object which was registered either by the same client, or by any other device driver. It can be also any server's own object.

Example: Server contains the following information:

```
<obj name="device1" href="/obix/device1/">
  <obj name="configure" href="conf/" />
    <str name="name" href="name" val="My Device"/>
  </obj>
</obj>
```

There are two options to read the "*configure*" object of "*device1*":

- If this device was registered by the same client earlier, than the client should use device ID assigned to this device and provide parameter URI relative to the root of device data (in current example it is "`conf/`").
- In case if that device was registered by someone else, than `0` should be used instead device ID + full URI of the required object (in this example it is `"/obix/device1/conf/"`).

If the whole object "*device1*" should be read and it was previously published by the same client, then the client should provide device ID assigned to this device and `NULL` as *paramUri*.

Note:

Although there is no need for this function in the normal workflow (see [Usage](#)) it still can be used, for instance, during initialization phase for obtaining some data from the server. Usage of this function for periodical reading of some object is not efficient and should be avoided. Use [obix_registerListener](#) instead.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device whose data should be read or `0` if the object doesn't belong to devices registered by this client.

paramUri URI of the object. It should be either relative to the device record like it was provided during device registration, or relative to the server root if the object doesn't belong to devices registered by this client.

output If read command executed successfully the DOM representation of the read object is stored here.

Returns:

[OBIX_SUCCESS](#) on success, negative error code otherwise.

Definition at line 735 of file obix_client.c.

References `_Connection::comm`, `connection_get()`, `device_get()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_SUCCESS`, `_Comm_Stack::read`, and `TRUE`.

Referenced by `checkEventsTask()`.

7.8.5.15 int obix_readValue (int *connectionId*, int *deviceId*, const char * *paramUri*, char ** *output*)

Reads value of the specified parameter from the oBIX server.

This function can be used to read value of any object on the server: It can be some parameter of device which was registered either by the same client, or by any other device driver. It can be also any value of the server settings.

Example: Server contains the following information:

```
<obj name="device1" href="/obix/device1/">
  <obj name="configure" href="conf/" />
    <str name="name" href="name" val="My Device"/>
  </obj>
</obj>
```

There are two ways to read the value of "name" string of "device1":

- If this device was registered by the same client earlier, than the client should use device ID assigned to this device and provide parameter URI relative to the root of device data (in current example it is "`conf/name`").
- In case if that device was registered by someone else, than *0* should be used instead device ID + full URI of the required parameter (in this example it is "`/obix/device1/conf/name`").

Note:

This method is used to read only *val* attribute of some object, but not the whole object itself. If you need to read the whole oBIX object then use [obix_read](#) instead.

Although there is no need for this function in the normal workflow (see [Usage](#)), it still can be used, for instance, during initialization phase for obtaining some data from the server. Usage of this function for periodical reading of some object is not efficient and should be avoided. Use [obix_registerListener](#) instead.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device whose parameter should be read or *0* if the parameter doesn't belong to devices registered by this client.

paramUri URI of the parameter. It should be either relative to the device record like it was provided during device registration, or relative to the server root if the parameter doesn't belong to devices registered by this client.

output If read command executed successfully the attribute's value is stored here.

Returns:

[OBIX_SUCCESS](#) on success, negative error code otherwise.

Definition at line 702 of file obix_client.c.

References `_Connection::comm`, `connection_get()`, `device_get()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_SUCCESS`, `_Comm_Stack::readValue`, and `TRUE`.

7.8.5.16 int obix_registerDevice (int *connectionId*, const char * *obixData*)

Posts the provided device information to the oBIX server.

Note:

Input data is not tested to conform with oBIX specification. It is strongly recommended to provide `displayName` attribute to every object. Also attributes `href` and `writable` are obligatory for all device parameters which are going to be changed by the device driver or external oBIX server users:

- `writable` attribute should be set to `true`.
- `href` attribute of the parent object should be a valid URI, relative to the server root (start with "/").
- `href` attributes of all child objects should have a valid URIs, relative to the parent object.

Example:

```
<obj name="light" href="/kitchen/light/" displayName="Kitchen lights">
    <bool name="switch"
        href="switch/"
        displayName="Switch"
        is="obix:bool"
        val="true"
        writable="true"/>
</obj>
```

Note:

Parent object should specify `href` attribute but the oBIX server is free to modify it (for instance, add prefix of the device storage), thus the URI can't be used to refer to the device record. Use the assigned device ID instead.

Parameters:

connectionId ID of the connection which should be used.

obixData oBIX object representing a new device.

Returns:

- `>0` ID of the created device record;
- `<0` error code.

Definition at line 576 of file obix_client.c.

References `connection_get()`, `device_register()`, `_Connection::deviceCount`, `_Connection::devices`, `log_error`, `_Connection::maxDevices`, `OBIX_ERR_LIMIT_REACHED`, `OBIX_ERR_UNKNOWN_BUG`, `OBIX_SUCCESS`, and `TRUE`.

Referenced by `main()`, and `testConnectionAndDevices()`.

7.8.5.17 int obix_registerListener (int *connectionId*, int *deviceId*, const char * *paramUri*, obix_update_listener *listener*)

Registers listener for device parameter updates.

Overwrites existing listener if it is called twice for the same parameter.

This method can be also used to subscribe for the updates of any other objects stored at the oBIX server. In that case *O* should be provided as *deviceId* and *paramUri* should be relative to the server root.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device whose parameter should be monitored or *O* if the parameter doesn't belong to devices registered by this client.

paramUri URI of the parameter which should be monitored. It should be either relative to the device record like it was provided during device registration, or relative to the server root if the parameter doesn't belong to devices registered by this client.

listener Pointer to the listener function which would be invoked every time when the subscribed parameter is changed.

Note:

listener method should be quick. Slow listener (especially if it waits for some resource) will block subsequent calls to all listeners.

Returns:

- ≥ 0 ID of the created listener;
- < 0 error code.

Definition at line 627 of file obix_client.c.

References `connection_get()`, `device_get()`, `listener_register()`, `_Device::listenerCount`, `_Device::listeners`, `log_error`, `_Connection::maxListeners`, `OBIX_ERR_LIMIT_REACHED`, `OBIX_ERR_UNKNOWN_-BUG`, `OBIX_SUCCESS`, and `TRUE`.

Referenced by `main()`.

7.8.5.18 int obix_unregisterDevice (int *connectionId*, int *deviceId*)

Removes device record from the oBIX server.

Also removes all listeners which were registered for this device.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device which should be removed.

Returns:

`OBIX_SUCCESS` if the device record is removed successfully, error code otherwise.

Definition at line 609 of file obix_client.c.

References connection_get(), device_get(), device_unregister(), OBIX_SUCCESS, and TRUE.

Referenced by testConnectionAndDevices().

7.8.5.19 int obix_unregisterListener (int *connectionId*, int *deviceId*, int *listenerId*)

Unregisters listener of device parameter updates.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device whose parameter is now monitored or *0* if the parameter doesn't belong to devices registered by this client.

listenerId ID of listener to be removed.

Returns:

OBIX_SUCCESS if the listener is removed successfully, error code otherwise.

Definition at line 671 of file obix_client.c.

References connection_get(), device_get(), listener_unregister(), _Device::listeners, _Connection::maxListeners, OBIX_ERR_INVALID_ARGUMENT, OBIX_SUCCESS, and TRUE.

Referenced by main().

7.8.5.20 int obix_writeValue (int *connectionId*, int *deviceId*, const char * *paramUri*, const char * *newValue*, OBIX_DATA_TYPE *dataType*)

Overwrites value of the specified device parameter at the oBIX server.

This function can be also used to change a value of any *writable* object at the oBIX server.

Parameters:

connectionId ID of the connection which should be used.

deviceId ID of the device whose parameter should be changed or *0* if the parameter doesn't belong to devices registered by this client.

paramUri URI of the parameter. It should be either relative to the device record like it was provided during device registration, or relative to the server root if changing parameter doesn't belong to devices registered by this client.

newValue Text representation of the new value to be written. It should be a new value for the *val* attribute of the oBIX object on the server, not the whole object.

Note:

Only value of an object (*val* attribute) can be written using this method. It's not possible to overwrite a whole oBIX object on the server.

Parameters:

dataType Type of data which is written to the server.

Returns:

OBIX_SUCCESS on success, negative error code otherwise.

See also:

[obix_readValue\(\)](#) for the usage example.

Definition at line 767 of file obix_client.c.

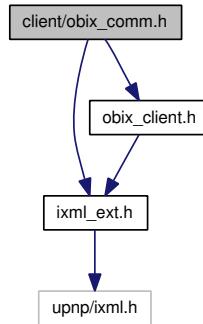
References `_Connection::comm`, `connection_get()`, `device_get()`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_SUCCESS`, `TRUE`, and `_Comm_Stack::writeValue`.

Referenced by `sendFallEventUpdate()`, and `timerTask()`.

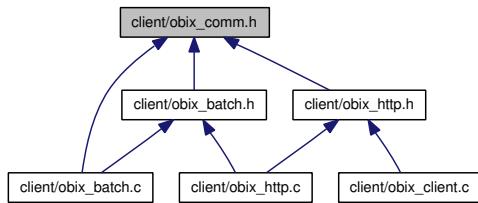
7.9 client/obix_comm.h File Reference

```
#include <ixml_ext.h>
#include <obix_client.h>
```

Include dependency graph for obix_comm.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [_Comm_Stack](#)
- struct [_Connection](#)
- struct [_Device](#)
- struct [_Listener](#)

Typedefs

- `typedef int(* comm_closeConnection)(Connection *connection)`
- `typedef void(* comm_freeConnection)(Connection *connection)`
- `typedef int(* comm_initConnection)(IXML_Element *connItem, Connection **connection)`
- `typedef int(* comm_openConnection)(Connection *connection)`
- `typedef int(* comm_read)(Connection *connection, Device *device, const char *paramUri, IXML_Element **output)`
- `typedef int(* comm_readValue)(Connection *connection, Device *device, const char *paramUri, char **output)`
- `typedef int(* comm_registerDevice)(Connection *connection, Device **device, const char *data)`
- `typedef int(* comm_registerListener)(Connection *connection, Device *device, Listener **listener)`
- `typedef int(* comm_sendBatch)(oBIX_Batch *batch)`
- `typedef struct _Comm_Stack Comm_Stack`
- `typedef int(* comm_unregisterDevice)(Connection *connection, Device *device)`
- `typedef int(* comm_unregisterListener)(Connection *connection, Device *device, Listener *listener)`
- `typedef int(* comm_writeValue)(Connection *connection, Device *device, const char *paramUri, const char *newValue, OBIX_DATA_TYPE dataType)`
- `typedef struct _Connection Connection`
- `typedef struct _Device Device`
- `typedef struct _Listener Listener`

Enumerations

- `enum Connection_Type { OBIX_HTTP }`

Functions

- `int connection_get (int connectionId, BOOL isConnected, Connection **connection)`
- `int device_get (Connection *connection, int deviceId, Device **device)`

7.9.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [obix_comm.h](#).

7.9.2 Typedef Documentation

7.9.2.1 `typedef int(* comm_closeConnection)(Connection *connection)`

Definition at line 47 of file obix_comm.h.

7.9.2.2 `typedef void(* comm_freeConnection)(Connection *connection)`

Definition at line 48 of file obix_comm.h.

7.9.2.3 `typedef int(* comm_initConnection)(IXML_Element *connItem, Connection **connection)`

Definition at line 44 of file obix_comm.h.

7.9.2.4 `typedef int(* comm_openConnection)(Connection *connection)`

Definition at line 46 of file obix_comm.h.

7.9.2.5 `typedef int(* comm_read)(Connection *connection, Device *device, const char *paramUri, IXML_Element **output)`

Definition at line 63 of file obix_comm.h.

7.9.2.6 `typedef int(* comm_readValue)(Connection *connection, Device *device, const char *paramUri, char **output)`

Definition at line 59 of file obix_comm.h.

7.9.2.7 `typedef int(* comm_registerDevice)(Connection *connection, Device **device, const char *data)`

Definition at line 49 of file obix_comm.h.

7.9.2.8 `typedef int(* comm_registerListener)(Connection *connection, Device *device, Listener **listener)`

Definition at line 53 of file obix_comm.h.

7.9.2.9 `typedef int(* comm_sendBatch)(oBIX_Batch *batch)`

Definition at line 72 of file obix_comm.h.

7.9.2.10 `typedef struct _Comm_Stack Comm_Stack`**7.9.2.11 `typedef int(* comm_unregisterDevice)(Connection *connection, Device *device)`**

Definition at line 52 of file obix_comm.h.

7.9.2.12 `typedef int(* comm_unregisterListener)(Connection *connection, Device *device, Listener *listener)`

Definition at line 56 of file obix_comm.h.

7.9.2.13 `typedef int(* comm_writeValue)(Connection *connection, Device *device, const char *paramUri, const char *newValue, OBIX_DATA_TYPE dataType)`

Definition at line 67 of file obix_comm.h.

7.9.2.14 `typedef struct _Connection Connection`

Definition at line 40 of file obix_comm.h.

7.9.2.15 `typedef struct _Device Device`

Definition at line 41 of file obix_comm.h.

7.9.2.16 `typedef struct _Listener Listener`

Definition at line 42 of file obix_comm.h.

7.9.3 Enumeration Type Documentation**7.9.3.1 `enum Connection_Type`****Enumerator:**

OBIX_HTTP

Definition at line 35 of file obix_comm.h.

7.9.4 Function Documentation

7.9.4.1 int connection_get (int *connectionId*, BOOL *isConnected*, Connection ** *connection*)

Definition at line 408 of file obix_client.c.

References _connectionCount, _Connection::isConnected, OBIX_ERR_INVALID_ARGUMENT, OBIX_ERR_INVALID_STATE, and OBIX_SUCCESS.

Referenced by obix_batch_create(), obix_closeConnection(), obix_openConnection(), obix_read(), obix_readValue(), obix_registerDevice(), obix_registerListener(), obix_unregisterDevice(), obix_unregisterListener(), and obix_writeValue().

7.9.4.2 int device_get (Connection * *connection*, int *deviceId*, Device ** *device*)

Definition at line 227 of file obix_client.c.

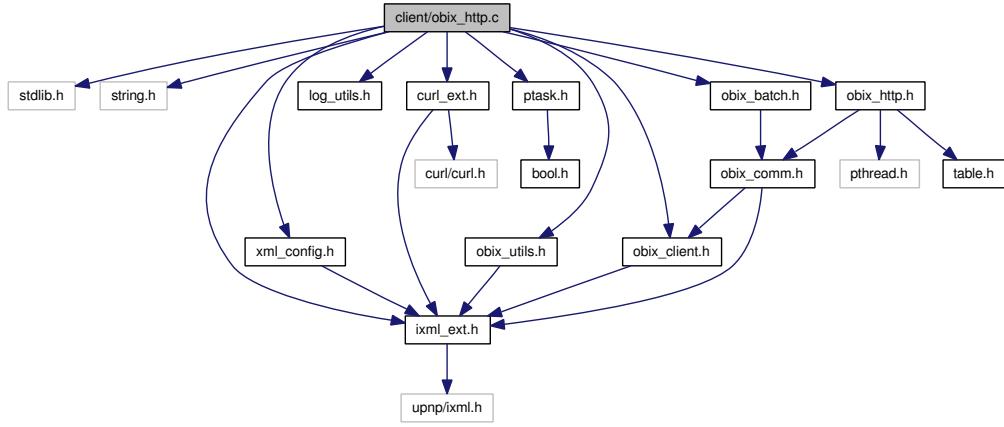
References _Connection::devices, _Connection::maxDevices, OBIX_ERR_INVALID_ARGUMENT, and OBIX_SUCCESS.

Referenced by obix_batch_addCommand(), obix_read(), obix_readValue(), obix_registerListener(), obix_unregisterDevice(), obix_unregisterListener(), and obix_writeValue().

7.10 client/obix_http.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <ixml_ext.h>
#include <xml_config.h>
#include <log_utils.h>
#include <curl_ext.h>
#include <ptask.h>
#include <obix_utils.h>
#include "obix_client.h"
#include "obix_batch.h"
#include "obix_http.h"
```

Include dependency graph for obix_http.c:



Defines

- #define **DEFAULT_POLLING_INTERVAL** 500
- #define **DEFAULT_WATCH_LEASE_PADDING** 20000
- #define **OBIX_BATCH_TEMPLATE_CMD_READ** "<uri is=\\"obix:Read\\" val=\\"%s\\" />\r\n"
- #define **OBIX_BATCH_TEMPLATE_CMD_READ_LENGTH** 32
- #define **OBIX_BATCH_TEMPLATE_CMD_WRITE**
- #define **OBIX_BATCH_TEMPLATE_CMD_WRITE_LENGTH** 65
- #define **OBIX_BATCH_TEMPLATE_FOOTER** "</list>"
- #define **OBIX_BATCH_TEMPLATE_FOOTER_LENGTH** 7
- #define **OBIX_BATCH_TEMPLATE_HEADER** "<list is=\\"obix:BatchIn\\" of=\\"obix:uri\\\">\r\n"
- #define **OBIX_BATCH_TEMPLATE_HEADER_LENGTH** 40
- #define **OBIX_WATCH_IN_TEMPLATE_FOOTER**
- #define **OBIX_WATCH_IN_TEMPLATE_FOOTER_LENGTH** 17
- #define **OBIX_WATCH_IN_TEMPLATE_HEADER**
- #define **OBIX_WATCH_IN_TEMPLATE_HEADER_LENGTH** 62
- #define **OBIX_WATCH_IN_TEMPLATE_URI** "<uri val=\\"%s\\" />\r\n"
- #define **OBIX_WATCH_IN_TEMPLATE_URI_LENGTH** 19
- #define **OBIX_WRITE_REQUEST_TEMPLATE** "<%s href=\\"%s\\" val=\\"%s\\" />"
- #define **OBIX_WRITE_REQUEST_TEMPLATE_LENGTH** 18

Functions

- static int **addListener** (Http_Connection *c, const char *paramUri, Listener *listener)
- static int **addWatchItem** (Http_Connection *c, const char **paramUri, int count, IXML_Document **response, CURL_EXT *curlHandle)
- static int **checkResponseDoc** (IXML_Document *respDoc, IXML_Element **respElem)
- static int **checkResponseElement** (IXML_Element *element)
- static int **createWatch** (Http_Connection *c, CURL_EXT *curlHandle)
- static char * **getAbsUri** (Http_Connection *connection, Device *device, const char *paramUri)

Combines server's URI, device's URI and parameter's URI (one of the two last can be empty), thus obtaining full URL.

- static `Http_Connection * getHttpConnection (Connection *connection)`
- static `Http_Device * getHttpDevice (Device *device)`
- static `char * getObjectUri (IXML_Document *doc, const char *objName, Http_Connection *c, BOOL full)`
- static `char * getRelUri (Device *device, const char *paramUri)`

Combines device's URI and parameter's URI (one of them can be empty), thus obtaining URI relative to the server root.
- static `char * getStrBatch (oBIX_Batch *batch)`
- static `char * getStrWatchIn (const char **paramUri, int count)`
- int `http_closeConnection (Connection *connection)`
- int `http_dispose ()`

Calls when HTTP stack is not initialized have no effect.
- void `http_freeConnection (Connection *connection)`
- int `http_init ()`

Subsequent calls have no effect.
- int `http_initConnection (IXML_Element *connItem, Connection **connection)`
- int `http_openConnection (Connection *connection)`
- int `http_read (Connection *connection, Device *device, const char *paramUri, IXML_Element **output)`
- int `http_readValue (Connection *connection, Device *device, const char *paramUri, char **output)`
- int `http_registerDevice (Connection *connection, Device **device, const char *data)`
- int `http_registerListener (Connection *connection, Device *device, Listener **listener)`
- int `http_sendBatch (oBIX_Batch *batch)`
- int `http_unregisterDevice (Connection *connection, Device *device)`
- int `http_unregisterListener (Connection *connection, Device *device, Listener *listener)`
- int `http_writeValue (Connection *connection, Device *device, const char *paramUri, const char *newValue, OBIX_DATA_TYPE dataType)`
- static `char * ixmLElement_getFullHref (IXML_Element *element, Http_Connection *c, BOOL full)`
- static `int parseElementValue (IXML_Element *element, char **output)`
- static `int parseWatchOut (IXML_Document *doc, Http_Connection *c, CURL_EXT *curlHandle)`
- static `int recreateWatch (Http_Connection *c, IXML_Document **response, CURL_EXT *curlHandle)`

Creates new Watch object after a failure of previous one.
- static `int removeListener (Http_Connection *c, const char *paramUri)`
- static `const char * removeServerAddress (const char *uri, Http_Connection *c)`
- static `int removeWatch (Http_Connection *c)`
- static `void resetWatchUris (Http_Connection *c)`
- static `int setWatchLeaseTime (Http_Connection *c, CURL_EXT *curlHandle, IXML_Document *watchXml)`
- static `int setWatchPollWaitTime (Http_Connection *c, CURL_EXT *curlHandle, IXML_Document *watchXml)`
- static `int setWatchTimeParam (Http_Connection *c, CURL_EXT *curlHandle, IXML_Document *watchXml, const char *paramName, long paramValue)`
- static `void watchPollTask (void *arg)`
- static `int writeValue (const char *paramUri, const char *newValue, OBIX_DATA_TYPE dataType, CURL_EXT *curlHandle)`

Variables

- static `CURL_EXT * _curl_handle`
- static `CURL_EXT * _curl_watch_handle`
- static `BOOL _initialized`
- static `Task_Thread * _watchThread`
- static const char * `CT_LONG_POLL` = "long-poll"
- static const char * `CT_LONG_POLL_MAX` = "max-interval"
- static const char * `CT_LONG_POLL_MIN` = "min-interval"
- static const char * `CT_POLL_INTERVAL` = "poll-interval"
- static const char * `CT_SERVER_ADDRESS` = "server-address"
- static const char * `CT_WATCHLEASE` = "watch-lease"
- static const char * `CTA_LOBBY` = "lobby"
- static const char * `OBIX_CONTRACT_LONG_POLL_WATCH` = "LongPollWatch"
- static const char ** `OBIX_DATA_TYPE_NAMES` []
- const `Comm_Stack OBIX_HTTP_COMM_STACK`
- static const char * `OBIX_WATCH_OUT_VALUES` = "values"

7.10.1 Detailed Description

[Todo](#)

add description here

Author:

Andrey Litvinov

Version:

1.1

Definition in file [obix_http.c](#).

7.10.2 Define Documentation

7.10.2.1 `#define DEFAULT_POLLING_INTERVAL 500`

Definition at line 42 of file obix_http.c.

Referenced by `http_initConnection()`.

7.10.2.2 `#define DEFAULT_WATCHLEASE_PADDING 20000`

Definition at line 43 of file obix_http.c.

Referenced by `http_initConnection()`.

7.10.2.3 #define OBIX_BATCH_TEMPLATE_CMD_READ " <uri is=\\"obix:Read\\" val=\\"%s\\" />\r\n"

Definition at line 1829 of file obix_http.c.

Referenced by getStrBatch().

7.10.2.4 #define OBIX_BATCH_TEMPLATE_CMD_READ_LENGTH 32

Definition at line 1830 of file obix_http.c.

Referenced by getStrBatch().

7.10.2.5 #define OBIX_BATCH_TEMPLATE_CMD_WRITE

Value:

```
(" <uri is=\\"obix:Write\\" val=\\"%s\\" >\r\n" \
    "   <%s name=\\"in\\" val=\\"%s\\" />\r\n" \
    "   </uri>\r\n")
```

Definition at line 1831 of file obix_http.c.

Referenced by getStrBatch().

7.10.2.6 #define OBIX_BATCH_TEMPLATE_CMD_WRITE_LENGTH 65

Definition at line 1834 of file obix_http.c.

Referenced by getStrBatch().

7.10.2.7 #define OBIX_BATCH_TEMPLATE_FOOTER "</list>"

Definition at line 1827 of file obix_http.c.

Referenced by getStrBatch().

7.10.2.8 #define OBIX_BATCH_TEMPLATE_FOOTER_LENGTH 7

Definition at line 1828 of file obix_http.c.

Referenced by getStrBatch().

7.10.2.9 #define OBIX_BATCH_TEMPLATE_HEADER "<list is=\"obix:BatchIn\" of=\"obix:uri\">\r\n"

Definition at line 1825 of file obix_http.c.

Referenced by getStrBatch().

7.10.2.10 #define OBIX_BATCH_TEMPLATE_HEADER_LENGTH 40

Definition at line 1826 of file obix_http.c.

Referenced by getStrBatch().

7.10.2.11 #define OBIX_WATCH_IN_TEMPLATE_FOOTER

Value:

```
("  </list>\r\n" \
    "</obj>")
```

Definition at line 52 of file obix_http.c.

Referenced by getStrWatchIn().

7.10.2.12 #define OBIX_WATCH_IN_TEMPLATE_FOOTER_LENGTH 17

Definition at line 54 of file obix_http.c.

Referenced by getStrWatchIn().

7.10.2.13 #define OBIX_WATCH_IN_TEMPLATE_HEADER

Value:

```
("<obj is=\"obix:WatchIn\">\r\n" \
    "  <list name=\"refs\" of=\"obix:Uri\">\r\n")
```

Definition at line 45 of file obix_http.c.

Referenced by getStrWatchIn().

7.10.2.14 #define OBIX_WATCH_IN_TEMPLATE_HEADER_LENGTH 62

Definition at line 47 of file obix_http.c.

Referenced by getStrWatchIn().

7.10.2.15 #define OBIX_WATCH_IN_TEMPLATE_URI "*<uri val=%s/>\r\n"*

Definition at line 49 of file obix_http.c.

Referenced by getStrWatchIn().

7.10.2.16 #define OBIX_WATCH_IN_TEMPLATE_URI_LENGTH 19

Definition at line 50 of file obix_http.c.

Referenced by getStrWatchIn().

7.10.2.17 #define OBIX_WRITE_REQUEST_TEMPLATE "<%s href=\"%s\" val=\"%s\"/>"

Definition at line 56 of file obix_http.c.

Referenced by writeValue().

7.10.2.18 #define OBIX_WRITE_REQUEST_TEMPLATE_LENGTH 18

Definition at line 57 of file obix_http.c.

Referenced by writeValue().

7.10.3 Function Documentation**7.10.3.1 static int addListener (Http_Connection * *c*, const char * *paramUri*, Listener * *listener*) [static]**

Definition at line 945 of file obix_http.c.

References createWatch(), EXECUTE_INDEFINITE, log_error, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, _Http_Connection::pollInterval, _Http_Connection::pollWaitMax, ptask_schedule(), table_put(), _Http_Connection::watchAddUri, _Http_Connection::watchMutex, watchPollTask(), _Http_Connection::watchPollTaskId, and _Http_Connection::watchTable.

Referenced by http_registerListener().

7.10.3.2 static int addWatchItem (Http_Connection * *c*, const char ** *paramUri*, int *count*, IXML_Document ** *response*, CURL_EXT * *curlHandle*) [static]

Definition at line 377 of file obix_http.c.

References `curl_ext_postDOM()`, `getStrWatchIn()`, `log_error`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_ERR_NO_MEMORY`, `OBIX_SUCCESS`, `_CURL_EXT::outputBuffer`, `_Http_Connection::serverUri`, `_Http_Connection::serverUriLength`, and `_Http_Connection::watchAddUri`.

Referenced by `http_registerListener()`, and `recreateWatch()`.

7.10.3.3 static int checkResponseDoc (IXML_Document * *respDoc*, IXML_Element ** *respElem*) [static]

Definition at line 622 of file obix_http.c.

References `checkResponseElement()`, `ixmlDocument_getRootElement()`, `log_error`, and `OBIX_ERR_BAD_CONNECTION`.

Referenced by `http.openConnection()`, `http_read()`, `http_registerDevice()`, `http_sendBatch()`, `http_unregisterListener()`, `parseWatchOut()`, and `removeWatch()`.

7.10.3.4 static int checkResponseElement (IXML_Element * *element*) [static]

Definition at line 608 of file obix_http.c.

References `ixmlelement_getNode()`, `log_error`, `OBIX_ERR_SERVER_ERROR`, `OBIX_OBJ_ERR`, and `OBIX_SUCCESS`.

Referenced by `checkResponseDoc()`, and `http_sendBatch()`.

7.10.3.5 static int createWatch (Http_Connection * *c*, CURL_EXT * *curlHandle*) [static]

Definition at line 504 of file obix_http.c.

References `curl_ext_postDOM()`, `FALSE`, `getObjectUri()`, `_CURL_EXT::inputBuffer`, `log_error`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_NAME_WATCH_ADD`, `OBIX_NAME_WATCH_DELETE`, `OBIX_NAME_WATCH_POLLCHANGES`, `OBIX_NAME_WATCH_REMOVE`, `OBIX_SUCCESS`, `_CURL_EXT::outputBuffer`, `_Http_Connection::serverUri`, `_Http_Connection::serverUriLength`, `setWatchLeaseTime()`, `setWatchPollWaitTime()`, `TRUE`, `_Http_Connection::watchAddUri`, `_Http_Connection::watchDeleteUri`, `_Http_Connection::watchMakeUri`, `_Http_Connection::watchPollChangesFullUri`, and `_Http_Connection::watchRemoveUri`.

Referenced by `addListener()`, and `recreateWatch()`.

7.10.3.6 static char* getAbsUri (Http_Connection * *connection*, Device * *device*, const char * *paramUri*) [static]

Combines server's URI, device's URI and parameter's URI (one of the two last can be empty), thus obtaining full URL.

Definition at line 1045 of file obix_http.c.

References `getHttpDevice()`, `_Http_Connection::serverUri`, `_Http_Connection::serverUriLength`, `_Http_Device::uri`, and `_Http_Device::uriLength`.

Referenced by http_read(), and http_writeValue().

7.10.3.7 static **Http_Connection*** getHttpConnection (**Connection * connection**) [static]

Definition at line 105 of file obix_http.c.

Referenced by http_closeConnection(), http_freeConnection(), http_openConnection(), http_read(), http_registerDevice(), http_registerListener(), http_sendBatch(), http_unregisterDevice(), http_unregisterListener(), and http_writeValue().

7.10.3.8 static **Http_Device*** getHttpDevice (**Device * device**) [static]

Definition at line 111 of file obix_http.c.

Referenced by getAbsUri(), getRelUri(), and http_unregisterDevice().

7.10.3.9 static **char*** getObjectUri (**IXML_Document * doc**, **const char * objName**, **Http_Connection * c**, **BOOL full**) [static]

Definition at line 245 of file obix_http.c.

References ixmldocument_getElementByAttrValue(), ixmlelement_getFullHref(), log_error, and OBIX_ATTR_NAME.

Referenced by createWatch(), http_openConnection(), and setWatchTimeParam().

7.10.3.10 static **char*** getRelUri (**Device * device**, **const char * paramUri**) [static]

Combines device's URI and parameter's URI (one of them can be empty), thus obtaining URI relative to the server root.

Definition at line 1007 of file obix_http.c.

References getHttpDevice(), _Http_Device::uri, and _Http_Device::uriLength.

Referenced by getStrBatch(), http_registerListener(), and http_unregisterListener().

7.10.3.11 static **char*** getStrBatch (**oBIX_Batch * batch**) [static]

Definition at line 1836 of file obix_http.c.

References _oBIX_Batch::command, _oBIX_Batch::commandCounter, _oBIX_BatchCmd::dataType, _oBIX_BatchCmd::device, getRelUri(), _oBIX_BatchCmd::id, _oBIX_BatchCmd::newValue, _oBIX_BatchCmd::next, OBIX_BATCH_TEMPLATE_CMD_READ, OBIX_BATCH_TEMPLATE_CMD_READ_LENGTH, OBIX_BATCH_TEMPLATE_CMD_WRITE, OBIX_BATCH_TEMPLATE_CMD_WRITE_LENGTH, OBIX_BATCH_TEMPLATE_FOOTER, OBIX_BATCH_TEMPLATE_FOOTER_LENGTH, OBIX_BATCH_TEMPLATE_HEADER, OBIX_BATCH_TEMPLATE_HEADER_LENGTH,

OBIX_BATCH_WRITE_VALUE, OBIX_DATA_TYPE_NAMES, _oBIX_BatchCmd::paramUri, and _oBIX_BatchCmd::type.

Referenced by http_sendBatch().

7.10.3.12 static char* getStrWatchIn (const char ***paramUri*, int *count*) [static]

Definition at line 294 of file obix_http.c.

References OBIX_WATCH_IN_TEMPLATE_FOOTER, OBIX_WATCH_IN_TEMPLATE_FOOTER_LENGTH, OBIX_WATCH_IN_TEMPLATE_HEADER, OBIX_WATCH_IN_TEMPLATE_HEADER_LENGTH, OBIX_WATCH_IN_TEMPLATE_URI, and OBIX_WATCH_IN_TEMPLATE_URI_LENGTH.

Referenced by addWatchItem(), and http_unregisterListener().

7.10.3.13 int http_closeConnection (Connection **connection*)

Definition at line 1473 of file obix_http.c.

References getHttpConnection(), log_debug, OBIX_SUCCESS, removeWatch(), _Http_Connection::serverUri, and _Http_Connection::watchDeleteUri.

7.10.3.14 int http_dispose ()

Calls when HTTP stack is not initialized have no effect.

Definition at line 1158 of file obix_http.c.

References _initialized, curl_ext_Dispose(), curl_ext_Free(), FALSE, OBIX_SUCCESS, ptask_Dispose(), and TRUE.

Referenced by obix_Dispose().

7.10.3.15 void http_freeConnection (Connection **connection*)

Definition at line 1366 of file obix_http.c.

References _Http_Connection::batchUri, getHttpConnection(), _Http_Connection::lobbyUri, resetWatchUris(), _Http_Connection::serverUri, _Http_Connection::signUpUri, table_Free(), _Http_Connection::watchMakeUri, _Http_Connection::watchMutex, and _Http_Connection::watchTable.

7.10.3.16 int http_init ()

Subsequent calls have no effect.

Definition at line 1106 of file obix_http.c.

References _initialized, curl_ext_create(), curl_ext_init(), OBIX_ERR_HTTP_LIB, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, ptask_init(), and TRUE.

Referenced by connection_create().

7.10.3.17 int http_initConnection (IXML_Element * *connItem*, Connection ** *connection*)

Definition at line 1176 of file obix_http.c.

References _Http_Connection::batchUri, _Http_Connection::c, config_getChildTag(), config_getTagAttributeValue(), config_getTagAttrLongValue(), CT_LONG_POLL, CT_LONG_POLL_MAX, CT_LONG_POLL_MIN, CT_POLL_INTERVAL, CT_SERVER_ADDRESS, CT_WATCHLEASE, CTA_LOBBY, CTA_VALUE, DEFAULT_POLLING_INTERVAL, DEFAULT_WATCHLEASE_PADDING, FALSE, _Http_Connection::lobbyUri, log_error, OBIX_ATTR_VAL, OBIX_ERR_HTTP_LIB, OBIX_ERR_INVALID_ARGUMENT, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, _Http_Connection::pollInterval, _Http_Connection::pollWaitMax, _Http_Connection::pollWaitMin, _Http_Connection::serverUri, _Http_Connection::serverUriLength, _Http_Connection::signUpUri, table_create(), TRUE, _Http_Connection::watchAddUri, _Http_Connection::watchDeleteUri, _Http_Connection::watchLease, _Http_Connection::watchMakeUri, _Http_Connection::watchMutex, _Http_Connection::watchPollChangesFullUri, _Http_Connection::watchRemoveUri, and _Http_Connection::watchTable.

7.10.3.18 int http_openConnection (Connection * *connection*)

Definition at line 1388 of file obix_http.c.

References _Http_Connection::batchUri, checkResponseDoc(), curl_ext_getDOM(), FALSE, getHttpConnection(), getObjectUri(), _Http_Connection::lobbyUri, log_debug, log_error, OBIX_ERR_BAD_CONNECTION, OBIX_NAME_BATCH, OBIX_NAME_SIGN_UP, OBIX_NAME_WATCH_SERVICE, OBIX_NAME_WATCH_SERVICE_MAKE, OBIX_SUCCESS, _Http_Connection::serverUri, _Http_Connection::signUpUri, TRUE, and _Http_Connection::watchMakeUri.

7.10.3.19 int http_read (Connection * *connection*, Device * *device*, const char * *paramUri*, IXML_Element ** *output*)

Definition at line 1741 of file obix_http.c.

References checkResponseDoc(), curl_ext_getDOM(), getAbsUri(), getHttpConnection(), xmlDoc_getRootElement(), log_error, OBIX_ERR_BAD_CONNECTION, OBIX_ERR_NO_MEMORY, and OBIX_SUCCESS.

Referenced by http_readValue().

7.10.3.20 int http_readValue (Connection * *connection*, Device * *device*, const char * *paramUri*, char ** *output*)

Definition at line 1790 of file obix_http.c.

References http_read(), OBIX_SUCCESS, and parseElementValue().

7.10.3.21 int http_registerDevice (Connection * *connection*, Device ** *device*, const char * *data*)

Definition at line 1488 of file obix_http.c.

References checkResponseDoc(), curl_ext_getDOM(), curl_ext_postDOM(), _Http_Device::d, getHttpConnection(), _CURL_EXT::inputBuffer, log_debug, log_error, log_warning, OBIX_ATTR_HREF, OBIX_ERR_BAD_CONNECTION, OBIX_ERR_INVALID_STATE, OBIX_ERR_NO_MEMORY, OBIX_ERR_SERVER_ERROR, OBIX_SUCCESS, _CURL_EXT::outputBuffer, removeServerAddress(), _Http_Connection::serverUri, _Http_Connection::serverUriLength, _Http_Connection::signUpUri, _Http_Device::uri, and _Http_Device::uriLength.

7.10.3.22 int http_registerListener (Connection * *connection*, Device * *device*, Listener ** *listener*)

Definition at line 1620 of file obix_http.c.

References addListener(), addWatchItem(), getHttpConnection(), getRelUri(), log_debug, log_error, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, parseWatchOut(), removeListener(), and _Http_Connection::serverUri.

7.10.3.23 int http_sendBatch (oBIX_Batch * *batch*)

Definition at line 1931 of file obix_http.c.

References _Http_Connection::batchUri, checkResponseDoc(), checkResponseElement(), _oBIX_Batch::command, _oBIX_Batch::connection, curl_ext_postDOM(), getHttpConnection(), getStrBatch(), _oBIX_BatchCmd::id, ixmlelement_cloneWithLog(), ixmlelement_getNode(), ixmlelement_convertToElement(), log_error, _oBIX_BatchCmd::next, OBIX_BATCH_READ, OBIX_BATCH_READ_VALUE, OBIX_ERR_BAD_CONNECTION, OBIX_ERR_NO_MEMORY, OBIX_ERR_UNKNOWN_BUG, OBIX_SUCCESS, _oBIX_BatchResult::obj, _CURL_EXT::outputBuffer, parseElementValue(), _oBIX_Batch::result, _Http_Connection::serverUri, _Http_Connection::serverUriLength, _oBIX_BatchResult::status, _oBIX_BatchCmd::type, and _oBIX_BatchResult::value.

7.10.3.24 int http_unregisterDevice (Connection * *connection*, Device * *device*)

Definition at line 1608 of file obix_http.c.

References getHttpConnection(), getHttpDevice(), log_debug, log_warning, and _Http_Device::uri.

7.10.3.25 int http_unregisterListener (Connection * *connection*, Device * *device*, Listener * *listener*)

Definition at line 1667 of file obix_http.c.

References `checkResponseDoc()`, `curl_ext_postDOM()`, `getHttpConnection()`, `getRelUri()`, `getStrWatchIn()`, `log_debug`, `log_error`, `log_warning`, `OBIX_CONTRACT_ERR_BAD_URI`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_ERR_NO_MEMORY`, `obix_obj_implementsContract()`, `OBIX_SUCCESS`, `_CURL_EXT::outputBuffer`, `_Listener::paramUri`, `removeListener()`, `_Http_Connection::serverUri`, `_Http_Connection::serverUriLength`, and `_Http_Connection::watchRemoveUri`.

7.10.3.26 int http_writeValue (Connection * *connection*, Device * *device*, const char * *paramUri*, const char * *newValue*, OBIX_DATA_TYPE *dataType*)

Definition at line 1805 of file obix_http.c.

References `getAbsUri()`, `getHttpConnection()`, `log_debug`, `log_error`, `OBIX_ERR_NO_MEMORY`, and `writeValue()`.

7.10.3.27 static char* ixmlelement_getFullHref (IXML_Element * *element*, Http_Connection * *c*, BOOL *full*) [static]

Definition at line 129 of file obix_http.c.

References `ixmlelement_getNode()`, `ixmlNode_convertToElement()`, `log_error`, `OBIX_ATTR_HREF`, `_Http_Connection::serverUri`, and `_Http_Connection::serverUriLength`.

Referenced by `getObjectUri()`.

7.10.3.28 static int parseElementValue (IXML_Element * *element*, char ** *output*) [static]

Definition at line 1081 of file obix_http.c.

References `ixmlelement_freeOwnerDocument()`, `ixmlelement_getNode()`, `log_error`, `log_warning`, `OBIX_ATTR_VAL`, `OBIX_ERR_INVALID_ARGUMENT`, `OBIX_ERR_NO_MEMORY`, and `OBIX_SUCCESS`.

Referenced by `http_readValue()`, and `http_sendBatch()`.

7.10.3.29 static int parseWatchOut (IXML_Document * *doc*, Http_Connection * *c*, CURL_EXT * *curlHandle*) [static]

Definition at line 646 of file obix_http.c.

References `_Listener::callback`, `checkResponseDoc()`, `_Listener::connectionId`, `_Listener::deviceId`, `FALSE`, `_Listener::id`, `ixmlDocument_getElementByAttrValue()`, `ixmlelement_getNode()`, `ixmlNode_convertToElement()`, `log_error`, `log_warning`, `OBIX_ATTR_HREF`, `OBIX_ATTR_NAME`, `OBIX_ATTR_VAL`, `OBIX_CONTRACT_ERR_BAD_URI`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_ERR_SERVER_ERROR`, `OBIX_OBJ_ERR`, `obix_obj_implementsContract()`, `OBIX_OBJ_LIST`, `OBIX_SUCCESS`, `OBIX_WATCH_OUT_VALUES`, `recreateWatch()`, `_Http_Connection::serverUri`, `_Http_Connection::serverUriLength`, `table_get()`, `TRUE`, and `_Http_Connection::watchTable`.

Referenced by `http_registerListener()`, and `watchPollTask()`.

7.10.3.30 static int recreateWatch (Http_Connection** * *c*, **IXML_Document** ** *response*,
CURL_EXT * *curlHandle*) [static]**

Creates new Watch object after a failure of previous one.

Definition at line 584 of file obix_http.c.

References addWatchItem(), createWatch(), CT_POLL_INTERVAL, CT_WATCHLEASE, log_warning, OBIX_SUCCESS, resetWatchUris(), table_getKeys(), and _Http_Connection::watchTable.

Referenced by parseWatchOut().

7.10.3.31 static int removeListener (Http_Connection** * *c*, const char * *paramUri*) [static]**

Definition at line 983 of file obix_http.c.

References _Table::count, OBIX_SUCCESS, removeWatch(), table_remove(), _Http_Connection::watchMutex, and _Http_Connection::watchTable.

Referenced by http_registerListener(), and http_unregisterListener().

7.10.3.32 static const char* removeServerAddress (const char * *uri*, **Http_Connection * *c*)
[static]**

Definition at line 116 of file obix_http.c.

References _Http_Connection::serverUri, and _Http_Connection::serverUriLength.

Referenced by http_registerDevice().

7.10.3.33 static int removeWatch (Http_Connection** * *c*) [static]**

Definition at line 862 of file obix_http.c.

References checkResponseDoc(), _Table::count, curl_ext_postDOM(), log_error, log_warning, OBIX_CONTRACT_ERR_BAD_URI, OBIX_ERR_BAD_CONNECTION, obix_obj_implementsContract(), OBIX_SUCCESS, _CURL_EXT::outputBuffer, ptask_cancel(), ptask_reschedule(), resetWatchUris(), _Http_Connection::serverUri, _Http_Connection::serverUriLength, TRUE, _Http_Connection::watchDeleteUri, _Http_Connection::watchMutex, _Http_Connection::watchPollTaskId, and _Http_Connection::watchTable.

Referenced by http_closeConnection(), and removeListener().

7.10.3.34 static void resetWatchUris (Http_Connection** * *c*) [static]**

Definition at line 270 of file obix_http.c.

References _Http_Connection::watchAddUri, _Http_Connection::watchDeleteUri, _Http_Connection::watchPollChangesFullUri, and _Http_Connection::watchRemoveUri.

Referenced by http_freeConnection(), recreateWatch(), and removeWatch().

7.10.3.35 static int setWatchLeaseTime (Http_Connection * *c*, CURL_EXT * *curlHandle*, IXML_Document * *watchXml*) [static]

Definition at line 484 of file obix_http.c.

References OBIX_ERR_BAD_CONNECTION, OBIX_NAME_WATCHLEASE, OBIX_SUCCESS, setWatchTimeParam(), and _Http_Connection::watchLease.

Referenced by createWatch().

7.10.3.36 static int setWatchPollWaitTime (Http_Connection * *c*, CURL_EXT * *curlHandle*, IXML_Document * *watchXml*) [static]

Definition at line 442 of file obix_http.c.

References ixmldocument_getRootElement(), log_warning, OBIX_CONTRACT_LONG_POLL_WATCH, OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MAX, OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MIN, obix_obj_implementsContract(), OBIX_SUCCESS, _Http_Connection::pollWaitMax, _Http_Connection::pollWaitMin, and setWatchTimeParam().

Referenced by createWatch().

7.10.3.37 static int setWatchTimeParam (Http_Connection * *c*, CURL_EXT * *curlHandle*, IXML_Document * *watchXml*, const char * *paramName*, long *paramValue*) [static]

Returns:

1 Parameter uri is not found;

Definition at line 407 of file obix_http.c.

References getObjectUri(), _CURL_EXT::inputBuffer, log_error, log_warning, OBIX_ERR_BAD_CONNECTION, OBIX_ERR_UNKNOWN_BUG, obix_reltimes_fromLong(), OBIX_T_RELTIME, RELTIME_SEC, _Http_Connection::serverUri, TRUE, and writeValue().

Referenced by setWatchLeaseTime(), and setWatchPollWaitTime().

7.10.3.38 static void watchPollTask (void * *arg*) [static]

Definition at line 812 of file obix_http.c.

References curl_ext_postDOM(), FALSE, log_debug, log_error, OBIX_SUCCESS, _CURL_EXT::outputBuffer, parseWatchOut(), ptask_cancel(), _Http_Connection::watchMutex, _Http_Connection::watchPollChangesFullUri, and _Http_Connection::watchPollTaskId.

Referenced by addListener().

7.10.3.39 static int writeValue (const char * *paramUri*, const char * *newValue*, OBIX_DATA_TYPE *dataType*, CURL_EXT * *curlHandle*) [static]

Definition at line 325 of file obix_http.c.

References curl_ext_put(), _CURL_EXT::inputBuffer, log_error, log_warning, OBIX_DATA_TYPE_NAMES, OBIX_ERR_BAD_CONNECTION, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, OBIX_WRITE_REQUEST_TEMPLATE, OBIX_WRITE_REQUEST_TEMPLATE_LENGTH, and _CURL_EXT::outputBuffer.

Referenced by http_writeValue(), and setWatchTimeParam().

7.10.4 Variable Documentation

7.10.4.1 CURL_EXT* _curl_handle [static]

Definition at line 100 of file obix_http.c.

7.10.4.2 CURL_EXT* _curl_watch_handle [static]

Definition at line 101 of file obix_http.c.

7.10.4.3 BOOL _initialized [static]

Definition at line 99 of file obix_http.c.

Referenced by http_dispose(), and http_init().

7.10.4.4 Task_Thread* _watchThread [static]

Definition at line 103 of file obix_http.c.

7.10.4.5 const char* CT_LONG_POLL = "long-poll" [static]

Definition at line 94 of file obix_http.c.

Referenced by http_initConnection().

7.10.4.6 const char* CT_LONG_POLL_MAX = "max-interval" [static]

Definition at line 96 of file obix_http.c.

Referenced by http_initConnection().

7.10.4.7 const char* CT_LONG_POLL_MIN = "min-interval" [static]

Definition at line 95 of file obix_http.c.

Referenced by http_initConnection().

7.10.4.8 const char* CT_POLL_INTERVAL = "poll-interval" [static]

Definition at line 92 of file obix_http.c.

Referenced by http_initConnection(), and recreateWatch().

7.10.4.9 const char* CT_SERVER_ADDRESS = "server-address" [static]

Definition at line 91 of file obix_http.c.

Referenced by http_initConnection(), and obix_server_init().

7.10.4.10 const char* CT_WATCHLEASE = "watch-lease" [static]

Definition at line 93 of file obix_http.c.

Referenced by http_initConnection(), and recreateWatch().

7.10.4.11 const char* CTA_LOBBY = "lobby" [static]

Definition at line 97 of file obix_http.c.

Referenced by http_initConnection().

7.10.4.12 const char* OBIX_CONTRACT_LONG_POLL_WATCH = "LongPollWatch" [static]

Definition at line 73 of file obix_http.c.

Referenced by setWatchPollWaitTime().

7.10.4.13 const char OBIX_DATA_TYPE_NAMES[] [static]**

Initial value:

```
{
    &OBIX_OBJ_BOOL,
    &OBIX_OBJ_INT,
    &OBIX_OBJ_REAL,
    &OBIX_OBJ_STR,
    &OBIX_OBJ_ENUM,
    &OBIX_OBJ_ABSTIME,
    &OBIX_OBJ_RELTIME,
    &OBIX_OBJ_URI
}
```

Definition at line 59 of file obix_http.c.

Referenced by getStrBatch(), and writeValue().

7.10.4.14 const Comm_Stack OBIX_HTTP_COMM_STACK

Initial value:

```
{
    &http_initConnection,
    &http_openConnection,
    &http_closeConnection,
    &http_freeConnection,
    &http_registerDevice,
    &http_unregisterDevice,
    &http_registerListener,
    &http_unregisterListener,
    &http_read,
    &http_readValue,
    &http_writeValue,
    &http_sendBatch
}
```

Definition at line 75 of file obix_http.c.

Referenced by connection_create().

7.10.4.15 const char* OBIX_WATCH_OUT_VALUES = "values" [static]

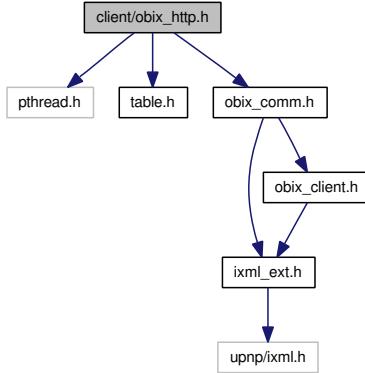
Definition at line 71 of file obix_http.c.

Referenced by parseWatchOut().

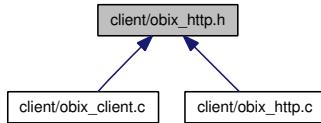
7.11 client/obix_http.h File Reference

```
#include <pthread.h>
#include <table.h>
#include <obix_comm.h>
```

Include dependency graph for obix_http.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `_Http_Connection`
- struct `_Http_Device`

Typedefs

- typedef struct `_Http_Connection` `Http_Connection`
- typedef struct `_Http_Device` `Http_Device`

Functions

- int `http_closeConnection` (`Connection` *connection)

Calls when HTTP stack is not initialized have no effect.
- int `http_dispose` ()

Subsequent calls have no effect.
- void `http_freeConnection` (`Connection` *connection)
- int `http_init` ()

Subsequent calls have no effect.
- int `http_initConnection` (`IXML_Element` *connItem, `Connection` **connection)
- int `http_openConnection` (`Connection` *connection)
- int `http_read` (`Connection` *connection, `Device` *device, const char *paramUri, `IXML_Element` **output)
- int `http_readValue` (`Connection` *connection, `Device` *device, const char *paramUri, char **output)
- int `http_registerDevice` (`Connection` *connection, `Device` **device, const char *data)

- int http_registerListener (Connection *connection, Device *device, Listener **listener)
- int http_sendBatch (oBIX_Batch *batch)
- int http_unregisterDevice (Connection *connection, Device *device)
- int http_unregisterListener (Connection *connection, Device *device, Listener *listener)
- int http_writeValue (Connection *connection, Device *device, const char *paramUri, const char *newValue, OBIX_DATA_TYPE dataType)

Variables

- const Comm_Stack OBIX_HTTP_COMM_STACK

7.11.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [obix_http.h](#).

7.11.2 Typedef Documentation

7.11.2.1 `typedef struct _Http_Connection Http_Connection`

7.11.2.2 `typedef struct _Http_Device Http_Device`

7.11.3 Function Documentation

7.11.3.1 `int http_closeConnection (Connection * connection)`

Definition at line 1473 of file obix_http.c.

References `getHttpConnection()`, `log_debug`, `OBIX_SUCCESS`, `removeWatch()`, `_Http_Connection::serverUri`, and `_Http_Connection::watchDeleteUri`.

7.11.3.2 int http_dispose ()

Calls when HTTP stack is not initialized have no effect.

Definition at line 1158 of file obix_http.c.

References _initialized, curl_ext_dispose(), curl_ext_free(), FALSE, OBIX_SUCCESS, ptask_dispose(), and TRUE.

Referenced by obix_dispose().

7.11.3.3 void http_freeConnection (Connection * *connection*)

Definition at line 1366 of file obix_http.c.

References _Http_Connection::batchUri, getHttpConnection(), _Http_Connection::lobbyUri, resetWatchUris(), _Http_Connection::serverUri, _Http_Connection::signUpUri, table_free(), _Http_Connection::watchMakeUri, _Http_Connection::watchMutex, and _Http_Connection::watchTable.

7.11.3.4 int http_init ()

Subsequent calls have no effect.

Definition at line 1106 of file obix_http.c.

References _initialized, curl_ext_create(), curl_ext_init(), OBIX_ERR_HTTP_LIB, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, ptask_init(), and TRUE.

Referenced by connection_create().

7.11.3.5 int http_initConnection (IXML_Element * *connItem*, Connection ** *connection*)

Definition at line 1176 of file obix_http.c.

References _Http_Connection::batchUri, _Http_Connection::c, config_getChildTag(), config_getTagAttributeValue(), config_getTagAttrLongValue(), CT_LONG_POLL, CT_LONG_POLL_MAX, CT_LONG_POLL_MIN, CT_POLL_INTERVAL, CT_SERVER_ADDRESS, CT_WATCHLEASE, CTA_LOBBY, CTA_VALUE, DEFAULT_POLLING_INTERVAL, DEFAULT_WATCHLEASE_PADDING, FALSE, _Http_Connection::lobbyUri, log_error, OBIX_ATTR_VAL, OBIX_ERR_HTTP_LIB, OBIX_ERR_INVALID_ARGUMENT, OBIX_ERR_NO_MEMORY, OBIX_SUCCESS, _Http_Connection::pollInterval, _Http_Connection::pollWaitMax, _Http_Connection::pollWaitMin, _Http_Connection::serverUri, _Http_Connection::serverUriLength, _Http_Connection::signUpUri, table_create(), TRUE, _Http_Connection::watchAddUri, _Http_Connection::watchDeleteUri, _Http_Connection::watchLease, _Http_Connection::watchMakeUri, _Http_Connection::watchMutex, _Http_Connection::watchPollChangesFullUri, _Http_Connection::watchRemoveUri, and _Http_Connection::watchTable.

7.11.3.6 int http_openConnection (Connection * *connection*)

Definition at line 1388 of file obix_http.c.

References `_Http_Connection::batchUri`, `checkResponseDoc()`, `curl_ext_getDOM()`, `FALSE`, `getHttpConnection()`, `getObjectUri()`, `_Http_Connection::lobbyUri`, `log_debug`, `log_error`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_NAME_BATCH`, `OBIX_NAME_SIGN_UP`, `OBIX_NAME_WATCH_SERVICE`, `OBIX_NAME_WATCH_SERVICE_MAKE`, `OBIX_SUCCESS`, `_Http_Connection::serverUri`, `_Http_Connection::signUpUri`, `TRUE`, and `_Http_Connection::watchMakeUri`.

7.11.3.7 int http_read (Connection * *connection*, Device * *device*, const char * *paramUri*, IXML_Element ** *output*)

Definition at line 1741 of file obix_http.c.

References `checkResponseDoc()`, `curl_ext_getDOM()`, `getAbsUri()`, `getHttpConnection()`, `ixmlDocument_getRootElement()`, `log_error`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_ERR_NO_MEMORY`, and `OBIX_SUCCESS`.

Referenced by `http_readValue()`.

7.11.3.8 int http_readValue (Connection * *connection*, Device * *device*, const char * *paramUri*, char ** *output*)

Definition at line 1790 of file obix_http.c.

References `http_read()`, `OBIX_SUCCESS`, and `parseElementValue()`.

7.11.3.9 int http_registerDevice (Connection * *connection*, Device ** *device*, const char * *data*)

Definition at line 1488 of file obix_http.c.

References `checkResponseDoc()`, `curl_ext_getDOM()`, `curl_ext_postDOM()`, `_Http_Device::d`, `getHttpConnection()`, `_CURL_EXT::inputBuffer`, `log_debug`, `log_error`, `log_warning`, `OBIX_ATTR_HREF`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_ERR_INVALID_STATE`, `OBIX_ERR_NO_MEMORY`, `OBIX_ERR_SERVER_ERROR`, `OBIX_SUCCESS`, `_CURL_EXT::outputBuffer`, `removeServerAddress()`, `_Http_Connection::serverUri`, `_Http_Connection::serverUriLength`, `_Http_Connection::signUpUri`, `_Http_Device::uri`, and `_Http_Device::uriLength`.

7.11.3.10 int http_registerListener (Connection * *connection*, Device * *device*, Listener ** *listener*)

Definition at line 1620 of file obix_http.c.

References `addListener()`, `addWatchItem()`, `getHttpConnection()`, `getRelUri()`, `log_debug`, `log_error`, `OBIX_ERR_NO_MEMORY`, `OBIX_SUCCESS`, `parseWatchOut()`, `removeListener()`, and `_Http_Connection::serverUri`.

7.11.3.11 int http_sendBatch (oBIX_Batch * batch)

Definition at line 1931 of file obix_http.c.

References `_Http_Connection::batchUri`, `checkResponseDoc()`, `checkResponseElement()`, `_oBIX_Batch::command`, `_oBIX_Batch::connection`, `curl_ext_postDOM()`, `getHttpConnection()`, `getStrBatch()`, `_oBIX_BatchCmd::id`, `ixmlElement_cloneWithLog()`, `ixmlElement_getNode()`, `ixmlNode_convertToElement()`, `log_error`, `_oBIX_BatchCmd::next`, `OBIX_BATCH_READ`, `OBIX_BATCH_READ_VALUE`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_ERR_NO_MEMORY`, `OBIX_ERR_UNKNOWN_BUG`, `OBIX_SUCCESS`, `_oBIX_BatchResult::obj`, `_CURL_EXT::outputBuffer`, `parseElementValue()`, `_oBIX_Batch::result`, `_Http_Connection::serverUri`, `_Http_Connection::serverUriLength`, `_oBIX_BatchResult::status`, `_oBIX_BatchCmd::type`, and `_oBIX_BatchResult::value`.

7.11.3.12 int http_unregisterDevice (Connection * connection, Device * device)

Definition at line 1608 of file obix_http.c.

References `getHttpConnection()`, `getHttpDevice()`, `log_debug`, `log_warning`, and `_Http_Device::uri`.

7.11.3.13 int http_unregisterListener (Connection * connection, Device * device, Listener * listener)

Definition at line 1667 of file obix_http.c.

References `checkResponseDoc()`, `curl_ext_postDOM()`, `getHttpConnection()`, `getRelUri()`, `getStrWatchIn()`, `log_debug`, `log_error`, `log_warning`, `OBIX_CONTRACT_ERR_BAD_URI`, `OBIX_ERR_BAD_CONNECTION`, `OBIX_ERR_NO_MEMORY`, `obix_obj_implementsContract()`, `OBIX_SUCCESS`, `_CURL_EXT::outputBuffer`, `_Listener::paramUri`, `removeListener()`, `_Http_Connection::serverUri`, `_Http_Connection::serverUriLength`, and `_Http_Connection::watchRemoveUri`.

7.11.3.14 int http_writeValue (Connection * connection, Device * device, const char * paramUri, const char * newValue, OBIX_DATA_TYPE dataType)

Definition at line 1805 of file obix_http.c.

References `getAbsUri()`, `getHttpConnection()`, `log_debug`, `log_error`, `OBIX_ERR_NO_MEMORY`, and `writeValue()`.

7.11.4 Variable Documentation**7.11.4.1 const Comm_Stack OBIX_HTTP_COMM_STACK**

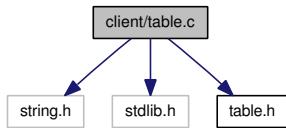
Definition at line 75 of file obix_http.c.

Referenced by `connection_create()`.

7.12 client/table.c File Reference

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

Include dependency graph for table.c:



Functions

- `Table * table_create (int initialSize)`
- static int `table_extend (Table *table)`
- void `table_free (Table *table)`
- void * `table_get (Table *table, const char *key)`
- int `table_getKeys (Table *table, const char ***keys)`
- int `table_getValues (Table *table, const void ***values)`
- int `table_put (Table *table, const char *key, void *value)`
- int `table_remove (Table *table, const char *key)`

7.12.1 Detailed Description

Todo

It supposed to be a hashtable, but now it is quite slow.

Author:

Andrey Litvinov

Version:

1.0

Definition in file [table.c](#).

7.12.2 Function Documentation

7.12.2.1 `Table* table_create (int initialSize)`

Definition at line 33 of file [table.c](#).

References `_Table::count`, `_Table::keys`, `_Table::size`, and `_Table::values`.

Referenced by `http_initConnection()`, and `test_table()`.

7.12.2.2 static int table_extend (Table * *table*) [static]

Definition at line 55 of file table.c.

References _Table::keys, _Table::size, and _Table::values.

Referenced by table_put().

7.12.2.3 void table_free (Table * *table*)

Definition at line 145 of file table.c.

References _Table::keys, _Table::size, and _Table::values.

Referenced by http_freeConnection(), and test_table().

7.12.2.4 void* table_get (Table * *table*, const char * *key*)

Definition at line 110 of file table.c.

References _Table::keys, _Table::size, and _Table::values.

Referenced by parseWatchOut(), test_table(), and testTableGet().

7.12.2.5 int table_getKeys (Table * *table*, const char * *keys*)**

Definition at line 163 of file table.c.

References _Table::count, and _Table::keys.

Referenced by recreateWatch().

7.12.2.6 int table_getValues (Table * *table*, const void * *values*)**

Definition at line 169 of file table.c.

References _Table::count, and _Table::values.

7.12.2.7 int table_put (Table * *table*, const char * *key*, void * *value*)

Definition at line 78 of file table.c.

References _Table::count, _Table::keys, _Table::size, table_extend(), and _Table::values.

Referenced by addListener(), and test_table().

7.12.2.8 int table_remove (Table * *table*, const char * *key*)

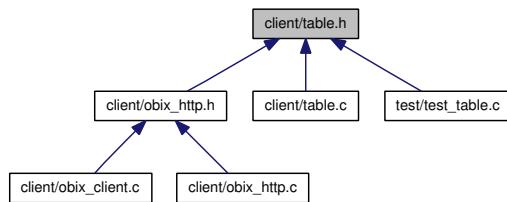
Definition at line 126 of file table.c.

References _Table::count, _Table::keys, and _Table::size.

Referenced by removeListener(), and test_table().

7.13 client/table.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct [_Table](#)

TypeDefs

- typedef struct [_Table](#) [Table](#)

Functions

- [Table * table_create](#) (int initialSize)
- void [table_free](#) (Table *table)
- void * [table_get](#) (Table *table, const char *key)
- int [table_getKeys](#) (Table *table, const char ***keys)
- int [table_getValues](#) (Table *table, const void ***values)
- int [table_put](#) (Table *table, const char *key, void *value)
- int [table_remove](#) (Table *table, const char *key)

7.13.1 Detailed Description

[Todo](#)

add description here

[Author:](#)

Andrey Litvinov

[Version:](#)

1.0

Definition in file [table.h](#).

7.13.2 Typedef Documentation

7.13.2.1 `typedef struct _Table Table`

7.13.3 Function Documentation

7.13.3.1 `Table* table_create (int initialSize)`

Definition at line 33 of file table.c.

References `_Table::count`, `_Table::keys`, `_Table::size`, and `_Table::values`.

Referenced by `http_initConnection()`, and `test_table()`.

7.13.3.2 `void table_free (Table * table)`

Definition at line 145 of file table.c.

References `_Table::keys`, `_Table::size`, and `_Table::values`.

Referenced by `http_freeConnection()`, and `test_table()`.

7.13.3.3 `void* table_get (Table * table, const char * key)`

Definition at line 110 of file table.c.

References `_Table::keys`, `_Table::size`, and `_Table::values`.

Referenced by `parseWatchOut()`, `test_table()`, and `testTableGet()`.

7.13.3.4 `int table_getKeys (Table * table, const char *** keys)`

Definition at line 163 of file table.c.

References `_Table::count`, and `_Table::keys`.

Referenced by `recreateWatch()`.

7.13.3.5 `int table_getValues (Table * table, const void *** values)`

Definition at line 169 of file table.c.

References `_Table::count`, and `_Table::values`.

7.13.3.6 int table_put (Table * *table*, const char * *key*, void * *value*)

Definition at line 78 of file table.c.

References _Table::count, _Table::keys, _Table::size, table_extend(), and _Table::values.

Referenced by addListener(), and test_table().

7.13.3.7 int table_remove (Table * *table*, const char * *key*)

Definition at line 126 of file table.c.

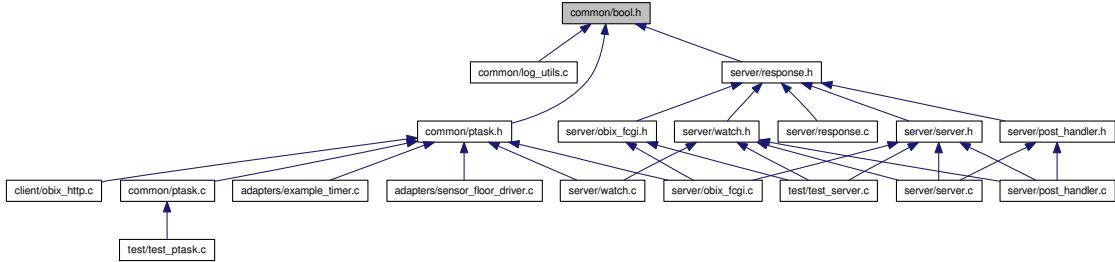
References _Table::count, _Table::keys, and _Table::size.

Referenced by removeListener(), and test_table().

7.14 common/bool.h File Reference

Defines boolean data type.

This graph shows which files directly or indirectly include this file:



Defines

- #define FALSE 0

This is false.

- #define TRUE 1

That's true.

TypeDefs

- typedef int BOOL

Boolean data type which is so natural for all programmers.

7.14.1 Detailed Description

Defines boolean data type.

Just in case if nobody defined it before.

Author:

Andrey Litvinov

Version:

1.0

Definition in file [bool.h](#).

7.14.2 Define Documentation

7.14.2.1 #define FALSE 0

This is *false*.

Definition at line 43 of file bool.h.

Referenced by compare_uri(), config_getTagAttrBoolValue(), config_log(), connection_create(), createWatch(), handlerWatchAdd(), handlerWatchPollHelper(), handlerWatchPollRefresh(), handlerWatchServiceMake(), http_dispose(), http_initConnection(), http_openConnection(), isResponseSent(), loadSettings(), log_usePrintf(), main(), obix_batch_resultInit(), obix_closeConnection(), obix_fcgi_dumpEnvironment(), obix_fcgi_init(), obix_fcgi_loadConfig(), obix_loadConfigFile(), obix_obj_implementsContract(), obix_openConnection(), obix_server_generateObixErrorMessage(), obix_server_generateResponse(), obix_server_read(), obix_server_write(), obixRequest_getHead(), obixResponse_add(), obixResponse_create(), obixResponse_isHead(), obixWatch_create(), obixWatch_deleteHelper(), obixWatch_isLongPoll(), obixWatch_isWatchUri(), obixWatch_notifyPollTask(), obixWatch_resetLeaseTimer(), obixWatchItem_isUpdated(), parseWatchOut(), periodicTask_create(), periodicTask_execute(), pollWatchItemIterator(), ptask_isScheduled(), putDeviceReference(), sendFallEventUpdate(), target_sendUpdate(), targetRemoveTask(), test_ptask(), test_server(), test_table(), testConnectionAndDevices(), testCurlExt(), testCurlExtRequest(), testDelete(), testGenerateResponse(), testGetClosestTask(), testObix_reltimes_fromLong(), testObix_reltimes_parse(), testObixLoadConfigFile(), testPeriodicTask(), testPtaskReschedule(), testPutHandler(), testResponse_setRightUri(), testSearch(), testSignUp(), testSignUpHelper(), testTableGet(), testWatch(), testWatchPollChanges(), testWatchRemove(), testWriteToDatabase(), threadCycle(), watchPollTask(), and xmldb_loadFile().

7.14.2.2 #define TRUE 1

That's *true*.

Definition at line 38 of file bool.h.

Referenced by checkTargets(), compare_uri(), completeWatchPollResponse(), config_getChildTag(), config_getTagAttrBoolValue(), config_getTagAttributeValue(), config_loadFile(), config_log(), connection_create(), createWatch(), dummyResponseListener(), handlerBatch(), handlerSignUp(), handlerWatchAdd(), handlerWatchDelete(), handlerWatchLongPoll(), handlerWatchPollChanges(),

handlerWatchPollHelper(), handlerWatchRemove(), handlerWatchServiceMake(), http_dispose(), http_init(), http_initConnection(), http_openConnection(), isResponseSent(), ixmlelement_cloneWithLog(), loadDeviceData(), loadSettings(), log_useSyslog(), main(), obix_batch_addCommand(), obix_batch_create(), obix_batch_removeCommand(), obix_closeConnection(), obix_fcgi_dumpEnvironment(), obix_loadConfigFile(), obix_obj_implementsContract(), obix_openConnection(), obix_read(), obix_readValue(), obix_registerDevice(), obix_registerListener(), obix_server_generateObixErrorMessage(), obix_server_init(), obix_server_read(), obix_server_write(), obix_unregisterDevice(), obix_unregisterListener(), obix_writeValue(), obixRequest_free(), obixRequest_getHead(), obixResponse_isError(), obixResponse_isHead(), obixResponse_setError(), obixWatch_delete(), obixWatch_dispose(), obixWatch_isLongPoll(), obixWatch_isWatchUri(), obixWatch_notifyPollTask(), obixWatchItem_isUpdated(), parseWatchOut(), periodicTask_execute(), pollWatchItemIterator(), ptask_cancel(), ptask_isScheduled(), putDeviceReference(), removeWatch(), sendFallEventUpdate(), setWatchTimeParam(), target_get(), target_sendUpdate(), target_setX(), target_setY(), targetRemoveTask(), taskStopThread(), test_ptask(), test_server(), test_table(), testConnectionAndDevices(), testCurlExt(), testCurlExtRequest(), testDelete(), testGenerateResponse(), testGetClosestTask(), testLog(), testObix_reltimes_fromLong(), testObix_reltimes_parse(), testObixLoadConfigFile(), testPeriodicTask(), testPtaskReschedule(), testPutHandler(), testResponse_setRightUri(), testSearch(), testSignUp(), testSignUpHelper(), testWatch(), testWatchPollChanges(), testWatchRemove(), testWriteToDatabase(), xmldb_put(), xmldb_putDOM(), and xmldb_putDOMHelper().

7.14.3 Typedef Documentation

7.14.3.1 `typedef int BOOL`

Boolean data type which is so natural for all programmers.

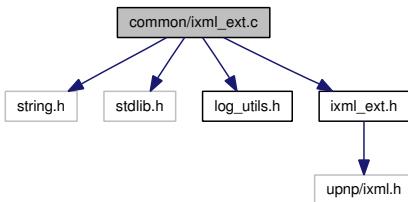
Definition at line 34 of file bool.h.

7.15 common/ixml_ext.c File Reference

Implements utility methods for work with XML DOM structure.

```
#include <string.h>
#include <stdlib.h>
#include "log_utils.h"
#include "ixml_ext.h"
```

Include dependency graph for ixml_ext.c:



Functions

- `IXML_Node * ixmleAttr_getNode (IXML_Attr *attr)`

Returns node which represents provided attribute.

- **IXML_Element * [ixmlAttr_getOwnerElement](#) (IXML_Attr *attr)**
Returns element which the provided attribute belongs to.
- **IXML_Element * [ixmlDocument_getElementByAttributeValue](#) (IXML_Document *doc, const char *attrName, const char *attrValue)**
Returns first element in the documents with provided attribute value.
- **IXML_Node * [ixmlDocument_getNode](#) (IXML_Document *doc)**
Returns node which represents provided document.
- **IXML_Element * [ixmlDocument_getRootElement](#) (IXML_Document *doc)**
Returns root element (root tag) of the XML document.
- **IXML_Element * [ixmlelement_cloneWithLog](#) (IXML_Element *source)**
Duplicates provided element.
- **int [ixmlelement_copyAttributeWithLog](#) (IXML_Element *source, IXML_Element *target, const char *attrName, BOOL obligatory)**
Copies attribute value from one element to another.
- **void [ixmlelement_freeOwnerDocument](#) (IXML_Element *element)**
Frees the IXML_Document which the provided element belongs to.
- **IXML_Node * [ixmlelement_getNode](#) (IXML_Element *element)**
Returns node which represents provided element.
- **IXML_Element * [ixmlelement_parseBuffer](#) (const char *data)**
Parses an XML text buffer and returns the parent element of the generated DOM structure.
- **int [ixmlelement_removeAttributeWithLog](#) (IXML_Element *element, const char *attrName)**
Removes attribute from the provided element.
- **int [ixmlelement_setAttributeWithLog](#) (IXML_Element *element, const char *attrName, const char *attrValue)**
Does the same with `ixmlDocument_getElementByHrefRecursive(IXML_Document, const char*, int)`.*
- **IXML_Attr * [ixmlNode_convertToAttr](#) (IXML_Node *node)**
Converts node to the attribute.
- **IXML_Element * [ixmlNode_convertToElement](#) (IXML_Node *node)**
Converts node to the element.
- **void [ixmlNode_freeOwnerDocument](#) (IXML_Node *node)**
Frees the IXML_Document which the provided node belongs to.
- **static IXML_Element * [ixmlNode_getElementByAttributeValue](#) (IXML_Node *node, const char *attrName, const char *attrValue)**
It is not good idea to make it public, because it searches for the required element also among neighbors of provided node (not only among it's children).

- IXML_Node * [ixmlNode_parseBuffer](#) (const char *data)

Parses an XML text buffer and returns the parent node of the generated DOM structure.

7.15.1 Detailed Description

Implements utility methods for work with XML DOM structure.

Definition in file [ixml_ext.c](#).

7.15.2 Function Documentation

7.15.2.1 IXML_Node* [ixmlAttr_getNode](#) (IXML_Attr *attr)

Returns node which represents provided attribute.

Parameters:

attr Attribute whose node representation is needed.

Returns:

Node corresponding to the provided attribute.

Definition at line 72 of file [ixml_ext.c](#).

Referenced by [xmldb_putMeta\(\)](#).

7.15.2.2 IXML_Element* [ixmlAttr_getOwnerElement](#) (IXML_Attr *attr)

Returns element which the provided attribute belongs to.

Parameters:

attr Attribute whose owner element should be returned.

Returns:

Owner element.

Definition at line 282 of file [ixml_ext.c](#).

Referenced by [xmldb_deleteMeta\(\)](#).

7.15.2.3 IXML_Element* [ixmlDocument_getElementByAttrValue](#) (IXML_Document *doc, const char *attrName, const char *attrValue)

Returns first element in the documents with provided attribute value.

Parameters:

doc Document where to search.
attrName Name of the attribute to check.
attrValue Attribute value which should be found

Returns:

A pointer to the element with matching attribute; *NULL* if no such element found.

Definition at line 336 of file ixml_ext.c.

References ixmldocument_getNode(), and ixmldocument_getElementByAttrValue().

Referenced by getObjectUri(), parseWatchOut(), and sendFallEventUpdate().

7.15.2.4 IXML_Node* ixmldocument_getNode (IXML_Document * *doc*)

Returns node which represents provided document.

Parameters:

doc Document whose node representation is needed.

Returns:

Node corresponding to the provided document.

Definition at line 67 of file ixml_ext.c.

Referenced by getNodeByHref(), ixmldocument_getElementByAttrValue(), ixmldocument_getRootElement(), ixmlelement_cloneWithLog(), ixmldocument_parseBuffer(), xmldb_getDump(), xmldb_printDump(), and xmldb_putDOMHelper().

7.15.2.5 IXML_Element* ixmldocument_getRootElement (IXML_Document * *doc*)

Returns root element (root tag) of the XML document.

Parameters:

doc Document, whose root element should be retrieved.

Returns:

Root element or *NULL* if the document is empty or other error occurred.

Definition at line 77 of file ixml_ext.c.

References ixmldocument_getNode(), and ixmldocument_convertToElement().

Referenced by checkResponseDoc(), config_loadFile(), http_read(), and setWatchPollWaitTime().

7.15.2.6 IXML_Element* ixmlelement_cloneWithLog (IXML_Element * *source*)

Duplicates provided element.

Creates new instance of *IXML_Document* and copies entire element including all its children to that document. Also writes message to log (using [log_utils.h](#)) on error.

Note:

Don't forget to free owner document of the clone after usage.

See also:

XmlNode_getOwnerDocument() at *ixml.h*

Parameters:

source Element to be copied.

Returns:

NULL on error, otherwise a pointer to the new copy of the source element.

Definition at line 214 of file *ixml_ext.c*.

References *ixmlDocument_getNode()*, *ixmlelement_getNode()*, *ixmlNode_convertToElement()*, *log_error*, and *TRUE*.

Referenced by *http_sendBatch()*, *loadDeviceData()*, *loadSettings()*, *normalizeObixDocument()*, and *xmldb_getObixSysObject()*.

7.15.2.7 int ixmlelement_copyAttributeWithLog (IXML_Element * *source*, IXML_Element * *target*, const char * *attrName*, BOOL *obligatory*)

Copies attribute value from one element to another.

If the attribute with provided name doesn't exist in the target node, it is created. Method also writes error messages using [log_utils.h](#) facilities.

Parameters:

source Element where the attribute will be copied from.

target Element which the attribute will be copied to.

attrName Name of the attribute to be copied.

obligatory Tells whether the attribute should necessarily present in the source tag. If *TRUE* and the attribute is missing than the error message will be logged. If it is *False* than only an error code will be returned in the same situation.

Returns:

IXML_SUCCESS if everything went well, or one of *ixml* error codes.

Definition at line 256 of file *ixml_ext.c*.

References *log_error*.

Referenced by *putDeviceReference()*.

7.15.2.8 void ixmlelement_freeOwnerDocument (IXML_Element * *element*)

Frees the *IXML_Document* which the provided element belongs to.

Note:

As long as the whole document is freed, all other nodes which belongs to the same document are also freed.

Parameters:

element Element which should be freed together with it's owner document.

Definition at line 251 of file ixmlext.c.

References ixmlelement_getNode().

Referenced by checkEventsTask(), config_finishInit(), handlerWatchServiceMake(), normalizeObixDocument(), obix_batch_resultClear(), obix_fcg_loadConfig(), obix_server_generateObixErrorMessage(), obix_server_handlePOST(), obix_server_handlePUT(), obixWatch_create(), parseElementValue(), and testWriteToDatabase().

7.15.2.9 IXML_Node* ixmlelement_getNode (IXML_Element * *element*)

Returns node which represents provided element.

Parameters:

element Element whose node representation is needed.

Returns:

Node corresponding to the provided element.

Definition at line 62 of file ixmlext.c.

Referenced by checkEventsTask(), checkResponseElement(), getLeaseTime(), getMetaInfo(), handlerBatch(), http_sendBatch(), ixmlelement_cloneWithLog(), ixmlelement_freeOwnerDocument(), ixmlelement_getFullHref(), loadDeviceData(), loadSettings(), normalizeObixDocument(), obix_server_write(), obixWatch_create(), obixWatch_updateMeta(), parseElementValue(), parseWatchOut(), putDeviceReference(), testBatch(), xmldb_deleteMeta(), xmldb_putDOMHelper(), and xmldb_putMeta().

7.15.2.10 IXML_Element* ixmlelement_parseBuffer (const char * *data*)

Parses an XML text buffer and returns the parent element of the generated DOM structure.

Note:

Don't forget to free memory allocated for the parsed document, not only the element (e.g. using [ixmlelement_freeOwnerDocument\(\)](#)).

Parameters:

data Text buffer to be parsed.

Returns:

Element representing parent tag of the parsed XML.

Definition at line 368 of file ixml_ext.c.

References `ixmlNode_convertToElement()`, and `ixmlNode_parseBuffer()`.

Referenced by `obix_server_handlePOST()`, `obix_server_handlePUT()`, and `testWriteToDatabase()`.

7.15.2.11 int ixmlelement_removeAttributeWithLog (IXML_Element * *element*, const char * *attrName*)

Removes attribute from the provided element.

Unlike `ixmlelement_removeAttribute()` the attribute node is removed totally, not only value. Also writes warning message to log (using [log_utils.h](#)) on error.

Parameters:

element Element from which the attribute should be removed.

attrName Name of the attribute to be removed.

Returns:

0 on success or 1 on error.

Definition at line 194 of file ixml_ext.c.

References `log_warning`.

7.15.2.12 int ixmlelement_setAttributeWithLog (IXML_Element * *element*, const char * *attrName*, const char * *attrValue*)

Does the same with `ixmlDocument_getElementByHrefRecursive(IXML_Document*,const char*,int)`.

Adds new attribute to the element.

The difference is that this implementation uses utility from upnp library to resolve URI. Such solution helps to resolve relative URI starting with '/'.

Definition at line 182 of file ixml_ext.c.

References `log_error`.

Referenced by `normalizeObixDocument()`, `obix_server_generateObixErrorMessage()`, `obixWatch_create()`, and `xmldb_updateDOM()`.

7.15.2.13 IXML_Attr* ixmlNode_convertToAttr (IXML_Node * *node*)

Converts node to the attribute.

Parameters:

node Node which should be converted.

Returns:

NULL if node is not an attribute.

Definition at line 47 of file ixml_ext.c.

Referenced by xmldb_deleteMeta().

7.15.2.14 IXML_Element* ixmlNode_convertToElement (IXML_Node * *node*)

Converts node to the element.

Parameters:

node Node which should be converted.

Returns:

NULL if node is not an element (i.e. tag).

Definition at line 31 of file ixml_ext.c.

Referenced by checkNode(), config_getChildTag(), getMetaInfo(), getNodeByHrefRecursive(), getUriSet(), handlerBatch(), http_sendBatch(), insertServerAddress(), ixmldocument_getRootElement(), ixmlelement_cloneWithLog(), ixmlelement_getFullHref(), ixmlelement_parseBuffer(), ixmlNode_getElementByAttrValue(), loadDeviceData(), obix_loadConfig(), obixWatch_updateMeta(), parseTarget(), parseWatchOut(), updateMetaWatch(), xmldb_getDOM(), and xmldb_putHelper().

7.15.2.15 void ixmlNode_freeOwnerDocument (IXML_Node * *node*)

Frees the *IXML_Document* which the provided node belongs to.

Note:

As long as the whole document is freed, all other nodes which belongs to the same document are also freed.

Parameters:

node Node which should be freed together with it's owner document.

Definition at line 246 of file ixml_ext.c.

7.15.2.16 static IXML_Element* ixmlNode_getElementByAttrValue (IXML_Node * *node*, const char * *attrName*, const char * *attrValue*) [static]

It is not good idea to make it public, because it searches for the required element also among neighbors of provided node (not only among it's children).

Definition at line 298 of file ixml_ext.c.

References ixmlNode_convertToElement().

Referenced by ixmlDocument_getElementByAttrValue().

7.15.2.17 IXML_Node* ixmlNode_parseBuffer (const char * *data*)

Parses an XML text buffer and returns the parent node of the generated DOM structure.

Note:

Don't forget to free memory allocated for the parsed document, not only the node (e.g. using [ixmlelement_freeOwnerDocument\(\)](#)).

Parameters:

data Text buffer to be parsed.

Returns:

Node representing parent tag of the parsed XML.

Definition at line 347 of file ixml_ext.c.

References ixmlDocument_getNode(), and log_warning.

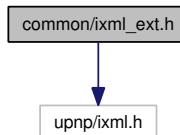
Referenced by ixmlelement_parseBuffer(), and xmldb_putHelper().

7.16 common/ixml_ext.h File Reference

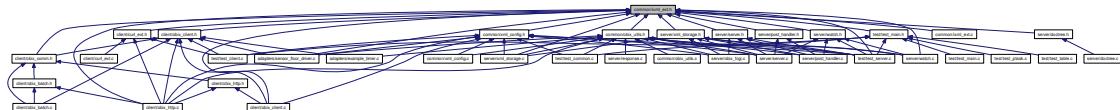
Defines utility methods for work with XML DOM structure.

```
#include <upnp/ixml.h>
```

Include dependency graph for ixml_ext.h:



This graph shows which files directly or indirectly include this file:



Defines

- `#define BOOL_H_`

xml.h already contains the definition of BOOL type, thus we consider that `bool.h` is already added.

Functions

- `IXML_Element * ixmlAttr_getOwnerElement (IXML_Attr *attr)`
Returns element which the provided attribute belongs to.
- `IXML_Element * ixmldocument_getElementByAttrValue (IXML_Document *doc, const char *attrName, const char *attrValue)`
Returns first element in the documents with provided attribute value.
- `IXML_Element * ixmldocument_getRootElement (IXML_Document *doc)`
Returns root element (root tag) of the XML document.
- `IXML_Element * ixmlelement_cloneWithLog (IXML_Element *source)`
Duplicates provided element.
- `int ixmlelement_copyAttributeWithLog (IXML_Element *source, IXML_Element *target, const char *attrName, BOOL obligatory)`
Copies attribute value from one element to another.
- `void ixmlelement_freeOwnerDocument (IXML_Element *element)`
Frees the IXML_Document which the provided element belongs to.
- `IXML_Element * ixmlelement_parseBuffer (const char *data)`
Parses an XML text buffer and returns the parent element of the generated DOM structure.
- `int ixmlelement_removeAttributeWithLog (IXML_Element *element, const char *attrName)`
Removes attribute from the provided element.
- `int ixmlelement_setAttributeWithLog (IXML_Element *element, const char *attrName, const char *attrValue)`
Adds new attribute to the element.
- `void ixmlelement_freeOwnerDocument (IXML_Node *node)`
Frees the IXML_Document which the provided node belongs to.
- `IXML_Node * ixmlelement_parseBuffer (const char *data)`
Parses an XML text buffer and returns the parent node of the generated DOM structure.

XML node types conversion

- `IXML_Node * ixmlelement_getNode (IXML_Attr *attr)`
Returns node which represents provided attribute.
- `IXML_Node * ixmldocument_getNode (IXML_Document *doc)`

Returns node which represents provided document.

- IXML_Node * [ixmlElement_getNode](#) (IXML_Element *element)
Returns node which represents provided element.
- IXML_Attr * [ixmlNode_convertToAttr](#) (IXML_Node *node)
Converts node to the attribute.
- IXML_Element * [ixmlNode_convertToElement](#) (IXML_Node *node)
Converts node to the element.

7.16.1 Detailed Description

Defines utility methods for work with XML DOM structure.

Expands functionality of *ixml* library which provides DOM XML parser functionality. *ixml* is distributed as a part of *libupnp* (<http://pupnp.sourceforge.net/>).

Definition in file [ixml_ext.h](#).

7.16.2 Define Documentation

7.16.2.1 #define BOOL_H_

ixml.h already contains the definition of **BOOL** type, thus we consider that [bool.h](#) is already added.

Definition at line 36 of file [ixml_ext.h](#).

7.16.3 Function Documentation

7.16.3.1 IXML_Node* ixmlAttr_getNode (IXML_Attr *attr)

Returns node which represents provided attribute.

Parameters:

attr Attribute whose node representation is needed.

Returns:

Node corresponding to the provided attribute.

Definition at line 72 of file [ixml_ext.c](#).

Referenced by [xmldb_putMeta\(\)](#).

7.16.3.2 IXML_Element* ixmlAttr_getOwnerElement (IXML_Attr *attr)

Returns element which the provided attribute belongs to.

Parameters:

attr Attribute whose owner element should be returned.

Returns:

Owner element.

Definition at line 282 of file ixml_ext.c.

Referenced by xmldb_deleteMeta().

7.16.3.3 IXML_Element* ixmlDocument_getElementByAttrValue (IXML_Document * *doc*, const char * *attrName*, const char * *attrValue*)

Returns first element in the documents with provided attribute value.

Parameters:

doc Document where to search.

attrName Name of the attribute to check.

attrValue Attribute value which should be found

Returns:

A pointer to the element with matching attribute; *NULL* if no such element found.

Definition at line 336 of file ixml_ext.c.

References ixmlDocument_getNode(), and ixmlNode_getElementByAttrValue().

Referenced by getObjectUri(), parseWatchOut(), and sendFallEventUpdate().

7.16.3.4 IXML_Node* ixmlDocument_getNode (IXML_Document * *doc*)

Returns node which represents provided document.

Parameters:

doc Document whose node representation is needed.

Returns:

Node corresponding to the provided document.

Definition at line 67 of file ixml_ext.c.

Referenced by getNodeByHref(), ixmlDocument_getElementByAttrValue(), ixmlDocument_getRootElement(), ixmlElement_cloneWithLog(), ixmlNode_parseBuffer(), xmldb_getDump(), xmldb_printDump(), and xmldb_putDOMHelper().

7.16.3.5 IXML_Element* ixmldocument_getRootElement (IXML_Document * doc)

Returns root element (root tag) of the XML document.

Parameters:

doc Document, whose root element should be retrieved.

Returns:

Root element or *NULL* if the document is empty or other error occurred.

Definition at line 77 of file ixml_ext.c.

References ixmldocument_getNode(), and ixmllnode_convertToElement().

Referenced by checkResponseDoc(), config_loadFile(), http_read(), and setWatchPollWaitTime().

7.16.3.6 IXML_Element* ixmlelement_cloneWithLog (IXML_Element * source)

Duplicates provided element.

Creates new instance of *IXML_Document* and copies entire element including all its children to that document. Also writes message to log (using [log_utils.h](#)) on error.

Note:

Don't forget to free owner document of the clone after usage.

See also:

ixmllnode_getOwnerDocument() at *ixml.h*

Parameters:

source Element to be copied.

Returns:

NULL on error, otherwise a pointer to the new copy of the source element.

Definition at line 214 of file ixml_ext.c.

References ixmldocument_getNode(), ixmlelement_getNode(), ixmllnode_convertToElement(), log_error, and TRUE.

Referenced by http_sendBatch(), loadDeviceData(), loadSettings(), normalizeObixDocument(), and xmldb_getObixSysObject().

7.16.3.7 int ixmlelement_copyAttributeWithLog (IXML_Element * source, IXML_Element * target, const char * attrName, BOOL obligatory)

Copies attribute value from one element to another.

If the attribute with provided name doesn't exist in the target node, it is created. Method also writes error messages using [log_utils.h](#) facilities.

Parameters:

source Element where the attribute will be copied from.

target Element which the attribute will be copied to.

attrName Name of the attribute to be copied.

obligatory Tells whether the attribute should necessarily present in the source tag. If *TRUE* and the attribute is missing than the error message will be logged. If it is *False* than only an error code will be returned in the same situation.

Returns:

IXML_SUCCESS if everything went well, or one of *ixml* error codes.

Definition at line 256 of file ixml_ext.c.

References [log_error](#).

Referenced by [putDeviceReference\(\)](#).

7.16.3.8 void **ixmlElement_freeOwnerDocument (IXML_Element * element)**

Frees the *IXML_Document* which the provided element belongs to.

Note:

As long as the whole document is freed, all other nodes which belongs to the same document are also freed.

Parameters:

element Element which should be freed together with it's owner document.

Definition at line 251 of file ixml_ext.c.

References [ixmlElement_getNode\(\)](#).

Referenced by [checkEventsTask\(\)](#), [config_finishInit\(\)](#), [handlerWatchServiceMake\(\)](#), [normalizeObixDocument\(\)](#), [obix_batch_resultClear\(\)](#), [obix_fcgi_loadConfig\(\)](#), [obix_server_generateObixErrorMessage\(\)](#), [obix_server_handlePOST\(\)](#), [obix_server_handlePUT\(\)](#), [obixWatch_create\(\)](#), [parseElementValue\(\)](#), and [testWriteToDatabase\(\)](#).

7.16.3.9 IXML_Node* **ixmlElement_getNode (IXML_Element * element)**

Returns node which represents provided element.

Parameters:

element Element whose node representation is needed.

Returns:

Node corresponding to the provided element.

Definition at line 62 of file ixml_ext.c.

Referenced by checkEventsTask(), checkResponseElement(), getLeaseTime(), getMetaInfo(), handlerBatch(), http_sendBatch(), ixmlelement_cloneWithLog(), ixmlelement_freeOwnerDocument(), ixmlelement_getFullHref(), loadDeviceData(), loadSettings(), normalizeObixDocument(), obix_server_write(), obixWatch_create(), obixWatch_updateMeta(), parseElementValue(), parseWatchOut(), putDeviceReference(), testBatch(), xmldb_deleteMeta(), xmldb_putDOMHelper(), and xmldb_putMeta().

7.16.3.10 IXML_Element* ixmlelement_parseBuffer (const char * *data*)

Parses an XML text buffer and returns the parent element of the generated DOM structure.

Note:

Don't forget to free memory allocated for the parsed document, not only the element (e.g. using [ixmlelement_freeOwnerDocument\(\)](#)).

Parameters:

data Text buffer to be parsed.

Returns:

Element representing parent tag of the parsed XML.

Definition at line 368 of file ixml_ext.c.

References ixmlelement_convertToElement(), and ixmlelement_parseBuffer().

Referenced by obix_server_handlePOST(), obix_server_handlePUT(), and testWriteToDatabase().

7.16.3.11 int ixmlelement_removeAttributeWithLog (IXML_Element * *element*, const char * *attrName*)

Removes attribute from the provided element.

Unlike *ixmlelement_removeAttribute()* the attribute node is removed totally, not only value. Also writes warning message to log (using [log_utils.h](#)) on error.

Parameters:

element Element from which the attribute should be removed.

attrName Name of the attribute to be removed.

Returns:

0 on success or 1 on error.

Definition at line 194 of file ixml_ext.c.

References [log_warning](#).

7.16.3.12 int ixmlelement_setAttributeWithLog (IXML_Element * *element*, const char * *attrName*, const char * *attrValue*)

Adds new attribute to the element.

If attribute with the same name already exists, it's value will be updated. Writes warning message to log (using [log_utils.h](#)) on error.

Parameters:

element Element to which the attribute should be added.

attrName Name of the attribute to be added.

attrValue Value of the attribute.

Returns:

0 on success or 1 on error.

Adds new attribute to the element.

The difference is that this implementation uses utility from upnp library to resolve URI. Such solution helps to resolve relative URI starting with '/'.

Definition at line 182 of file ixmlext.c.

References log_error.

Referenced by normalizeObixDocument(), obix_server_generateObixErrorMessage(), obixWatch_create(), and xmldb_updateDOM().

7.16.3.13 IXML_Attr* ixmlelement_convertToAttr (IXML_Node * *node*)

Converts node to the attribute.

Parameters:

node Node which should be converted.

Returns:

NULL if node is not an attribute.

Definition at line 47 of file ixmlext.c.

Referenced by xmldb_deleteMeta().

7.16.3.14 IXML_Element* ixmlelement_convertToElement (IXML_Node * *node*)

Converts node to the element.

Parameters:

node Node which should be converted.

Returns:

NULL if node is not an element (i.e. tag).

Definition at line 31 of file ixml_ext.c.

Referenced by checkNode(), config_getChildTag(), getMetaInfo(), getNodeByHrefRecursive(), getUriSet(), handlerBatch(), http_sendBatch(), insertServerAddress(), ixmldocument_getRootElement(), ixmlelement_cloneWithLog(), ixmlelement_getFullHref(), ixmlelement_parseBuffer(), ixmlelement_getElementByAttrValue(), loadDeviceData(), obix_loadConfig(), obixWatch_updateMeta(), parseTarget(), parseWatchOut(), updateMetaWatch(), xmldb_getDOM(), and xmldb_putHelper().

7.16.3.15 void ixmlelement_freeOwnerDocument (IXML_Node * *node*)

Frees the *IXML_Document* which the provided node belongs to.

Note:

As long as the whole document is freed, all other nodes which belongs to the same document are also freed.

Parameters:

node Node which should be freed together with it's owner document.

Definition at line 246 of file ixml_ext.c.

7.16.3.16 IXML_Node* ixmlelement_parseBuffer (const char * *data*)

Parses an XML text buffer and returns the parent node of the generated DOM structure.

Note:

Don't forget to free memory allocated for the parsed document, not only the node (e.g. using [ixmlelement_freeOwnerDocument\(\)](#)).

Parameters:

data Text buffer to be parsed.

Returns:

Node representing parent tag of the parsed XML.

Definition at line 347 of file ixml_ext.c.

References ixmldocument_getNode(), and log_warning.

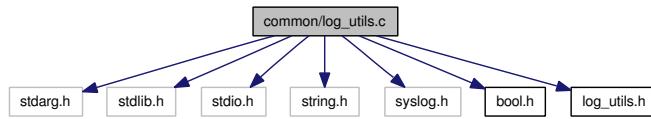
Referenced by ixmlelement_parseBuffer(), and xmldb_putHelper().

7.17 common/log_utils.c File Reference

Contains implementation of logging tools.

```
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <syslog.h>
#include "bool.h"
#include "log_utils.h"
```

Include dependency graph for log_utils.c:



Functions

- static void [log_nothing](#) (char *fmt,...)
- void [log_setLevel](#) (LOG_LEVEL level)

Sets the minimum priority level of the messages which will be processed.
- void [log_usePrintf](#) ()

Switches library to use printf for handling messages.
- void [log_useSyslog](#) (int facility)

Switches library to use syslog for handling messages.
- static void [setPrintf](#) ()
 • static void [setSyslog](#) ()

Logging to @a stdout

- static void [log_debugPrintf](#) (char *fmt,...)
- static void [log_errorPrintf](#) (char *fmt,...)
- static void [log_warningPrintf](#) (char *fmt,...)

Logging using @a syslog

- static void [log_debugSyslog](#) (char *fmt,...)
- static void [log_errorSyslog](#) (char *fmt,...)
- static void [log_warningSyslog](#) (char *fmt,...)

Variables

- static int [_log_level](#) = LOG_LEVEL_DEBUG
- static BOOL [_use_syslog](#) = FALSE

Log handlers.

- [log_function log_debugHandler](#) = &log_debugPrintf
Contains link to the current handler of debug log messages.
- [log_function log_errorHandler](#) = &log_errorPrintf
Contains link to the current handler of error log messages.
- [log_function log_warningHandler](#) = &log_warningPrintf
Contains link to the current handler of warning log messages.

7.17.1 Detailed Description

Contains implementation of logging tools.

Author:

Andrey Litvinov

Version:

2.0

Definition in file [log_utils.c](#).

7.17.2 Function Documentation

7.17.2.1 static void log_debugPrintf (char *fmt, ...) [static]

Definition at line 62 of file [log_utils.c](#).

Referenced by [setPrintf\(\)](#).

7.17.2.2 static void log_debugSyslog (char *fmt, ...) [static]

Definition at line 92 of file [log_utils.c](#).

Referenced by [setSyslog\(\)](#).

7.17.2.3 static void log_errorPrintf (char *fmt, ...) [static]

Definition at line 82 of file [log_utils.c](#).

Referenced by [setPrintf\(\)](#).

7.17.2.4 static void log_errorSyslog (char **fmt*, ...) [static]

Definition at line 108 of file log_utils.c.

Referenced by setSyslog().

7.17.2.5 static void log_nothing (char **fmt*, ...) [static]

Definition at line 116 of file log_utils.c.

Referenced by setPrintf(), and setSyslog().

7.17.2.6 void log_setLevel (LOG_LEVEL *level*)

Sets the minimum priority level of the messages which will be processed.

Parameters:

level Priority level:

- [LOG_LEVEL_DEBUG](#) - All messages will be printed;
- [LOG_LEVEL_WARNING](#) - Only warning and error messages will be printed;
- [LOG_LEVEL_ERROR](#) - Only error messages are printed;
- [LOG_LEVEL_NO](#) - Nothing is printed at all.

Definition at line 178 of file log_utils.c.

References _log_level, _use_syslog, setPrintf(), and setSyslog().

Referenced by config_log().

7.17.2.7 void log_usePrintf ()

Switches library to use *printf* for handling messages.

Definition at line 161 of file log_utils.c.

References _use_syslog, FALSE, and setPrintf().

Referenced by config_log(), and testLog().

7.17.2.8 void log_useSyslog (int *facility*)

Switches library to use *syslog* for handling messages.

Parameters:

facility Facility tells syslog who issued the message. See documentation of *syslog* for more information.

Definition at line 171 of file log_utils.c.

References _use_syslog, setSyslog(), and TRUE.

Referenced by config_log(), and parseArguments().

7.17.2.9 static void log_warningPrintf (char *fmt, ...) [static]

Definition at line 72 of file log_utils.c.

Referenced by setPrintf().

7.17.2.10 static void log_warningSyslog (char *fmt, ...) [static]

Definition at line 100 of file log_utils.c.

Referenced by setSyslog().

7.17.2.11 static void setPrintf () [static]

Definition at line 121 of file log_utils.c.

References _log_level, log_debugHandler, log_debugPrintf(), log_errorHandler, log_errorPrintf(), LOG_LEVEL_DEBUG, LOG_LEVEL_ERROR, LOG_LEVEL_WARNING, log_nothing(), log_warningHandler, and log_warningPrintf().

Referenced by log_setLevel(), and log_usePrintf().

7.17.2.12 static void setSyslog () [static]

Definition at line 141 of file log_utils.c.

References _log_level, log_debugHandler, log_debugSyslog(), log_errorHandler, log_errorSyslog(), LOG_LEVEL_DEBUG, LOG_LEVEL_ERROR, LOG_LEVEL_WARNING, log_nothing(), log_warningHandler, and log_warningSyslog().

Referenced by log_setLevel(), and log_useSyslog().

7.17.3 Variable Documentation

7.17.3.1 int _log_level = LOG_LEVEL_DEBUG [static]

Definition at line 58 of file log_utils.c.

Referenced by log_setLevel(), setPrintf(), and setSyslog().

7.17.3.2 BOOL _use_syslog = FALSE [static]

Definition at line 60 of file log_utils.c.

Referenced by log_setLevel(), log_usePrintf(), and log_useSyslog().

7.17.3.3 log_function log_debugHandler = &log_debugPrintf

Contains link to the current handler of *debug* log messages.

Normally [log_debug\(\)](#) should be used instead.

Definition at line 53 of file log_utils.c.

Referenced by setPrintf(), and setSyslog().

7.17.3.4 log_function log_errorHandler = &log_errorPrintf

Contains link to the current handler of *error* log messages.

Normally [log_error\(\)](#) should be used instead.

Definition at line 55 of file log_utils.c.

Referenced by setPrintf(), and setSyslog().

7.17.3.5 log_function log_warningHandler = &log_warningPrintf

Contains link to the current handler of *warning* log messages.

Normally [log_warning\(\)](#) should be used instead.

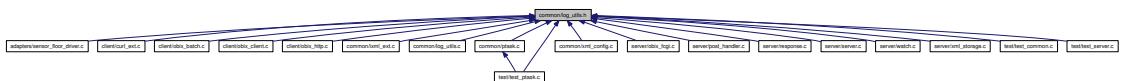
Definition at line 54 of file log_utils.c.

Referenced by setPrintf(), and setSyslog().

7.18 common/log_utils.h File Reference

Definitions of logging tools.

This graph shows which files directly or indirectly include this file:

**Defines****Logging utilities**

- #define `log_debug`(fmt,...)
Prints debug message to the configured output.
- #define `log_error`(fmt,...)
Prints error message to the configured output.
- #define `log_warning`(fmt,...)
Prints warning message to the configured output.

Typedefs

- typedef void(* `log_function`)(char *fmt,...)
This is a prototype of log handler function.

Enumerations

- enum `LOG_LEVEL` { `LOG_LEVEL_DEBUG`, `LOG_LEVEL_WARNING`, `LOG_LEVEL_ERROR`, `LOG_LEVEL_NO` }
Defines possible log levels.

Functions

- void `log_setLevel` (`LOG_LEVEL` level)
Sets the minimum priority level of the messages which will be processed.
- void `log_usePrintf` ()
Switches library to use printf for handling messages.
- void `log_useSyslog` (int facility)
Switches library to use syslog for handling messages.

Variables

Log handlers

Contain links to the current log handlers.

Normally these links should not be used directly.

- `log_function log_debugHandler`
Contains link to the current handler of debug log messages.
- `log_function log_errorHandler`
Contains link to the current handler of error log messages.
- `log_function log_warningHandler`
Contains link to the current handler of warning log messages.

7.18.1 Detailed Description

Definitions of logging tools.

Log system has two modes:

- *syslog* - when all log messages are forwarded to syslog;
- *printf* - all messages are printed using *printf* utility, thus in most cases are shown on a console.

The default mode is *printf*, but it can be switched at any time.

Library provides three simple methods for logging messages with different priority levels:

- [log_debug\(\)](#)
- [log_warning\(\)](#)
- [log_error\(\)](#)

Author:

Andrey Litvinov

Version:

2.0

Definition in file [log_utils.h](#).

7.18.2 Define Documentation

7.18.2.1 #define log_debug(fmt, ...)

Value:

```
(*log_debugHandler) ("%s(%d): " fmt, __FILE__, \
__LINE__, ## __VA_ARGS__)
```

Prints *debug* message to the configured output.

Automatically adds filename and string number of the place from where the log was written.

Parameters:

fmt Message format (used in the same way as with *printf()*).

Definition at line 85 of file [log_utils.h](#).

Referenced by `checkEventsTask()`, `config_finishInit()`, `config_getTagAttrBoolValue()`, `config_getTagAttrLongValue()`, `config_log()`, `config_setResourceDir()`, `curl_ext_get()`, `curl_ext_post()`, `curl_ext_put()`, `eventFeedListener()`, `handlerError()`, `handlerSignUp()`, `handlerWatchAdd()`, `handlerWatchDelete()`, `handlerWatchLongPoll()`, `handlerWatchRemove()`, `handlerWatchServiceMake()`, `http_closeConnection()`, `http_openConnection()`, `http_registerDevice()`, `http_registerListener()`, `http_unregisterDevice()`, `http_unregisterListener()`, `http_writeValue()`, `inputWriter()`, `main()`, `obix_fcgi_dumpEnvironment()`, `obix_fcgi_readRequestInput()`, `obix_server_invoke()`, `obix_server_shutdown()`, `obixWatch_holdPoll()`, `obixWatch_init()`, `outputReader()`, `parseArguments()`, `printXMLContents()`, `removeMetaInfo()`, `sendRequest()`, `stopTask()`, `taskDeleteWatch()`, `testLog()`, `threadCycle()`, `watchPollTask()`, and `xmldb_init()`.

7.18.2.2 #define log_error(fmt, ...)

Value:

```
(*log_errorHandler) ("%s(%d): " fmt, __FILE__, \
__LINE__, ## __VA_ARGS__)
```

Prints *error* message to the configured output.

Automatically adds filename and string number of the place from where the log was written.

Parameters:

fmt Message format (used in the same way as with *printf()*).

Definition at line 105 of file log_utils.h.

Referenced by addListener(), addResponsePart(), addWatchItem(), checkEventsTask(), checkResponseDoc(), checkResponseElement(), config_finishInit(), config_getChildTag(), config_getResFullPath(), config_getTagAttrBoolValue(), config_getTagAttributeValue(), config_getTagAttrIntValue(), config_getTagAttrLongValue(), config_loadFile(), config_log(), config_setResourceDir(), connection_create(), connection_free(), createWatch(), curl_ext_create(), curl_ext_get(), curl_ext_init(), curl_ext_post(), curl_ext_put(), device_register(), eventFeedListener(), generateWatchUri(), getLeaseTime(), getObjectUri(), http_initConnection(), http_openConnection(), http_read(), http_registerDevice(), http_registerListener(), http_sendBatch(), http_unregisterListener(), http_writeValue(), inputWriter(), ixmlelement_cloneWithLog(), ixmlelement_copyAttributeWithLog(), ixmlelement_getFullHref(), ixmlelement_setAttributeWithLog(), listener_register(), main(), normalizeObixDocument(), normalizeUri(), obix_batch_addCommand(), obix_batch_create(), obix_batch_resultInit(), obix_fcgi_dumpEnvironment(), obix_fcgi_handleRequest(), obix_fcgi_init(), obix_fcgi_readRequestInput(), obix_fcgi_sendResponse(), obix_loadConfig(), obix_registerDevice(), obix_registerListener(), obix_server_generateObixErrorMessage(), obix_server_generateResponse(), obix_server_init(), obixRequest_create(), obixRequest_get(), obixResponse_setError(), obixResponse_setRightUri(), obixResponse_setText(), obixWatch_appendWatchItem(), obixWatch_create(), obixWatch_createWatchItem(), obixWatch_delete(), obixWatch_deleteHelper(), obixWatch_deleteWatchItem(), obixWatch_getWatchItem(), obixWatch_holdPoll(), obixWatch_init(), obixWatch_notifyPollTask(), obixWatch_resetLeaseTimer(), obixWatch_updateMeta(), obixWatchItem_free(), parseElementValue(), parseTarget(), parseWatchOut(), parseXmlInput(), periodicTask_create(), ptask_init(), putDeviceReference(), removeWatch(), sendFallEventUpdate(), sendRequest(), setWatchTimeParam(), targetRemoveTask(), targetsListener(), taskDeleteWatch(), testLog(), threadCycle(), watchPollTask(), writeValue(), xmldb_deleteMeta(), xmldb_getFullUri(), xmldb_init(), xmldb_loadFile(), xmldb_putHelper(), and xmldb_putMeta().

7.18.2.3 #define log_warning(fmt, ...)

Value:

```
(*log_warningHandler) ("%s(%d): " fmt, __FILE__, \
__LINE__, ## __VA_ARGS__)
```

Prints *warning* message to the configured output.

Automatically adds filename and string number of the place from where the log was written.

Parameters:

fmt Message format (used in the same way as with *printf()*).

Definition at line 95 of file log_utils.h.

Referenced by checkNode(), config_getTagAttrBoolValue(), handlerWatchServiceMake(), http_registerDevice(), http_unregisterDevice(), http_unregisterListener(), inputWriter(), insertServerAddress(), ixmlelement_removeAttributeWithLog(), ixmlelement_parseBuffer(), main(), obix_fcgi_handleRequest(), obix_server_read(), obix_server_write(), obixWatch_create(), obixWatch_get(), obixWatch_getByUri(), obixWatch_init(), obixWatch_processTimeUpdates(), parseArguments(), parseElementValue(), parseWatchOut(), recreateWatch(), removeMetaInfo(), removeWatch(), sendErrorMessage(), sendFallEventUpdate(), setWatchPollWaitTime(), setWatchTimeParam(), targetsListener(), testCycle(), testLog(), threadCycle(), writeValue(), xmldb_delete(), xmldb_getFullUri(), xmldb_putDOMHelper(), and xmldb_updateDOM().

7.18.3 Typedef Documentation

7.18.3.1 **typedef void(* log_function)(char *fmt,...)**

This is a prototype of log handler function.

Parameters:

fmt Message format (used in the same way as with *printf()*).

Definition at line 49 of file log_utils.h.

7.18.4 Enumeration Type Documentation

7.18.4.1 **enum LOG_LEVEL**

Defines possible log levels.

Enumerator:

LOG_LEVEL_DEBUG Debug log level.

LOG_LEVEL_WARNING Warning log level.

LOG_LEVEL_ERROR Error log level.

LOG_LEVEL_NO 'No' log level.

Definition at line 112 of file log_utils.h.

7.18.5 Function Documentation

7.18.5.1 **void log_setLevel (LOG_LEVEL *level*)**

Sets the minimum priority level of the messages which will be processed.

Parameters:

level Priority level:

- [LOG_LEVEL_DEBUG](#) - All messages will be printed;
- [LOG_LEVEL_WARNING](#) - Only warning and error messages will be printed;
- [LOG_LEVEL_ERROR](#) - Only error messages are printed;
- [LOG_LEVEL_NO](#) - Nothing is printed at all.

Definition at line 178 of file log_utils.c.

References `_log_level`, `_use_syslog`, `setPrintf()`, and `setSyslog()`.

Referenced by `config_log()`.

7.18.5.2 void log_usePrintf()

Switches library to use *printf* for handling messages.

Definition at line 161 of file log_utils.c.

References `_use_syslog`, FALSE, and `setPrintf()`.

Referenced by `config_log()`, and `testLog()`.

7.18.5.3 void log_useSyslog (int *facility*)

Switches library to use *syslog* for handling messages.

Parameters:

facility Facility tells syslog who issued the message. See documentation of *syslog* for more information.

Definition at line 171 of file log_utils.c.

References `_use_syslog`, `setSyslog()`, and TRUE.

Referenced by `config_log()`, and `parseArguments()`.

7.18.6 Variable Documentation

7.18.6.1 log_function log_debugHandler

Contains link to the current handler of *debug* log messages.

Normally [log_debug\(\)](#) should be used instead.

Definition at line 53 of file log_utils.c.

Referenced by `setPrintf()`, and `setSyslog()`.

7.18.6.2 log_function log_errorHandler

Contains link to the current handler of *error* log messages.

Normally [log_error\(\)](#) should be used instead.

Definition at line 55 of file log_utils.c.

Referenced by `setPrintf()`, and `setSyslog()`.

7.18.6.3 log_function log_warningHandler

Contains link to the current handler of *warning* log messages.

Normally [log_warning\(\)](#) should be used instead.

Definition at line 54 of file log_utils.c.

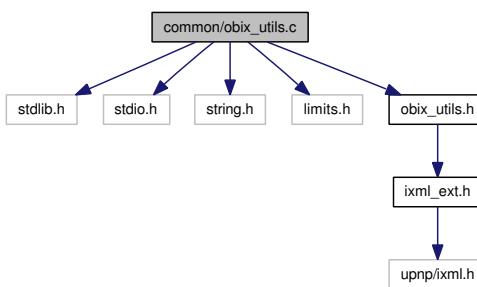
Referenced by `setPrintf()`, and `setSyslog()`.

7.19 common/obix_utils.c File Reference

Contains names of oBIX objects, contracts, facets, etc.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <limits.h>
#include "obix_utils.h"
```

Include dependency graph for obix_utils.c:



Functions

- [BOOL obix_obj_implementsContract \(IXML_Element *obj, const char *contract\)](#)
Checks whether oBIX object implements specified contract.
- [char * obix_reltIME_fromLong \(long millis, RELTIME_FORMAT format\)](#)
Generates reltime value from the provided time in milliseconds.

- int **obix_reltimes_parseToLong** (const char *str, long *duration)
Parses string value of reltime object and returns corresponding time in milliseconds.

Variables

- const char * **OBIX_ATTR_DISPLAY** = "display"
oBIX facet "display".
- const char * **OBIX_ATTR_DISPLAY_NAME** = "displayName"
oBIX facet "displayName".
- const char * **OBIX_ATTR_HREF** = "href"
Object attribute "href".
- const char * **OBIX_ATTR_IS** = "is"
Object attribute "is".
- const char * **OBIX_ATTR_NAME** = "name"
Object attribute "name".
- const char * **OBIX_ATTR_NULL** = "null"
Object attribute "null".
- const char * **OBIX_ATTR_VAL** = "val"
Object attribute "val".
- const char * **OBIX_ATTR_WRITABLE** = "writable"
oBIX facet "writable".
- const char * **OBIX_CONTRACT_ERR_BAD_URI** = "obix:BadUriErr"
URI of the BadUriErr error contract.
- const char * **OBIX_CONTRACT_ERR_PERMISSION** = "obix:PermissionErr"
URI of the PermissionErr error contract.
- const char * **OBIX_CONTRACT_ERR_UNSUPPORTED** = "obix:UnsupportedErr"
URI of the UnsupportedErr error contract.
- const char * **OBIX_NAME_BATCH** = "batch"
Name of batch operation in the Lobby object.
- const char * **OBIX_NAME_SIGN_UP** = "signUp"
Name of signUp operation in the Lobby object.
- const char * **OBIX_NAME_WATCH_ADD** = "add"
Name of the Watch.add operation.
- const char * **OBIX_NAME_WATCH_DELETE** = "delete"

Name of the Watch.delete operation.

- const char * **OBIX_NAME_WATCHLEASE** = "lease"

Name of the Watch.lease parameter.

- const char * **OBIX_NAME_WATCH_POLL_WAIT_INTERVAL** = "pollWaitInterval"

Name of the Watch.pollWaitInterval object.

- const char * **OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MAX** = "max"

Name of the Watch.pollWaitInterval.max parameter.

- const char * **OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MIN** = "min"

Name of the Watch.pollWaitInterval.min parameter.

- const char * **OBIX_NAME_WATCH_POLLCHANGES** = "pollChanges"

Name of the Watch.pollChanges operation.

- const char * **OBIX_NAME_WATCH_POLLREFRESH** = "pollRefresh"

Name of the Watch.pollRefresh operation.

- const char * **OBIX_NAME_WATCHREMOVE** = "remove"

Name of the Watch.remove operation.

- const char * **OBIX_NAME_WATCHSERVICE** = "watchService"

Name of the Watch Service in the Lobby object.

- const char * **OBIX_NAME_WATCHSERVICEMAKE** = "make"

Name of the watchService.make operation.

- const char * **OBIXOBJ** = "obj"

oBIX Object (obj)

- const char * **OBIXOBJABSTIME** = "abstime"

oBIX Absolute Time (abstime)

- const char * **OBIXOBJBOOL** = "bool"

oBIX Boolean (bool)

- const char * **OBIXOBJENUM** = "enum"

oBIX Enumeration (enum)

- const char * **OBIXOBJERR** = "err"

oBIX Error (err)

- const char * **OBIXOBJFEED** = "feed"

oBIX Feed (feed)

- const char * **OBIXOBJINT** = "int"

oBIX Integer (int)

- const char * **OBIX_OBJ_LIST** = "list"
oBIX List (list)
- const char * **OBIX_OBJ_NULL_TEMPLATE** = "<obj null=\\"true\\"/>"
String which represents oBIX NULL object.
- const char * **OBIX_OBJ_OP** = "op"
oBIX Operation (op)
- const char * **OBIX_OBJ_REAL** = "real"
oBIX Real (real)
- const char * **OBIX_OBJ_REF** = "ref"
oBIX Reference (ref)
- const char * **OBIX_OBJ_RELTIME** = "reltime"
oBIX Relative Duration of Time (reltime)
- const char * **OBIX_OBJ_STR** = "str"
oBIX String (str)
- const char * **OBIX_OBJ_URI** = "uri"
oBIX URI (uri)
- const char * **XML_FALSE** = "false"
String representation of boolean false value.
- const char * **XML_TRUE** = "true"
String representation of boolean true value.

7.19.1 Detailed Description

Contains names of oBIX objects, contracts, facets, etc.

(<http://obix.org/>)

Definition in file **obix_utils.c**.

7.19.2 Function Documentation

7.19.2.1 BOOL obix_obj_implementsContract (IXML_Element * *obj*, const char * *contract*)

Checks whether oBIX object implements specified contract.

Object implements a contract when contract's URI is listed in object's *is* attribute.

Parameters:

obj XML DOM structure representing an oBIX object.

contract URI of the contract which should be checked.

Returns:

TRUE if the object implements specified contract, **FALSE** otherwise.

Definition at line 422 of file obix_utils.c.

References FALSE, OBIX_ATTR_IS, and TRUE.

Referenced by handlerBatch(), http_unregisterListener(), parseTarget(), parseWatchOut(), removeWatch(), and setWatchPollWaitTime().

7.19.2.2 **char* obix_reltimes_fromLong (long duration, RELTIME_FORMAT format)**

Generates *reltime* value from the provided time in milliseconds.

Parameters:

duration Time in milliseconds which should be converted.

format Format of the generated *reltime* value. Specifies the maximum time component for the output value. For example, converting 2 minutes with *RELTIME_MIN* will result in "PT2M"; *RELTIME_SEC* - "PT120S".

Returns:

String, which represents provided time in *xs:duration* format, or *NULL* if memory allocation failed.

Definition at line 279 of file obix_utils.c.

References RELTIME_DAY, RELTIME_HOUR, and RELTIME_MIN.

Referenced by setWatchTimeParam(), testObix_reltimes_fromLong(), and timerTask().

7.19.2.3 **int obix_reltimes_parseToLong (const char * str, long * duration)**

Parses string value of *reltime* object and returns corresponding time in milliseconds.

Follows *xs:duration* format.

Note:

Durations which can overload *long* variable are not parsed.

Parameters:

str String value of a *reltime* object which should be parsed.

duration If parsing is successful than the parsed value will be written there.

Returns:

- *0* - Operation completed successfully;
- *-1* - Parsing error (provided string has bad format);

- -2 - Provided *reltime* value is bigger than or equal to 24 days (The maximum possible value is "P23DT23H59M59.999S"). Also this error code is returned when the input value is not normalized: If some of time components (e.g. hours) presents, than all smaller components (minutes and seconds) should represent less time than previous component. For example "PT60M" and "PT60S" are allowed, but "PT1H60M" and "PT1H60S" are not.

Parses positive integer. Does the same as `strtol(startPos, endPos, 10)`, but assumes that negative result is an error. Thus returns -1 instead of 0, when nothing is parsed. Also sets flag `parsedSomething` if parsing did not fail.

Definition at line 80 of file `obix_utils.c`.

Referenced by `getLeaseTime()`, `obixWatch_processTimeUpdates()`, and `testObix_reltimes_parseToLong()`.

7.19.3 Variable Documentation

7.19.3.1 const char* OBIX_ATTR_DISPLAY = "display"

oBIX facet "*display*".

Definition at line 74 of file `obix_utils.c`.

Referenced by `obix_server_generateObixErrorMessage()`, and `putDeviceReference()`.

7.19.3.2 const char* OBIX_ATTR_DISPLAY_NAME = "displayName"

oBIX facet "*displayName*".

Definition at line 75 of file `obix_utils.c`.

Referenced by `loadDeviceData()`, `obix_server_generateObixErrorMessage()`, and `putDeviceReference()`.

7.19.3.3 const char* OBIX_ATTR_HREF = "href"

Object attribute "*href*".

Definition at line 70 of file `obix_utils.c`.

Referenced by `checkNode()`, `getNodeByHrefRecursive()`, `handlerSignUp()`, `handlerWatchServiceMake()`, `http_registerDevice()`, `insertServerAddress()`, `ixmlElement_getFullHref()`, `loadDeviceData()`, `normalizeObixDocument()`, `normalizeUri()`, `obix_server_generateResponse()`, `obixWatch_create()`, `parseWatchOut()`, and `putDeviceReference()`.

7.19.3.4 const char* OBIX_ATTR_IS = "is"

Object attribute "*is*".

Definition at line 68 of file `obix_utils.c`.

Referenced by `obix_obj_implementsContract()`, and `obix_server_generateObixErrorMessage()`.

7.19.3.5 const char* OBIX_ATTR_NAME = "name"

Object attribute "*name*".

Definition at line 69 of file obix_utils.c.

Referenced by getObjectUri(), loadDeviceData(), parseTarget(), parseWatchOut(), and putDeviceReference().

7.19.3.6 const char* OBIX_ATTR_NULL = "null"

Object attribute "*null*".

Definition at line 72 of file obix_utils.c.

7.19.3.7 const char* OBIX_ATTR_VAL = "val"

Object attribute "*val*".

Definition at line 71 of file obix_utils.c.

Referenced by getLeaseTime(), getUriSet(), handlerBatch(), http_initConnection(), obixWatch_processTimeUpdates(), obixWatch_updateMeta(), parseElementValue(), parseTarget(), parseWatchOut(), sendFallEventUpdate(), xmldb_putMeta(), and xmldb_updateDOM().

7.19.3.8 const char* OBIX_ATTR_WRITABLE = "writable"

OBIX facet "*writable*".

Definition at line 73 of file obix_utils.c.

Referenced by xmldb_updateDOM().

7.19.3.9 const char* OBIX_CONTRACT_ERR_BAD_URI = "obix:BadUriErr"

URI of the *BadUriErr* error contract.

Definition at line 33 of file obix_utils.c.

Referenced by handlerWatchAdd(), http_unregisterListener(), obix_server_invoke(), obix_server_read(), obix_server_write(), parseWatchOut(), and removeWatch().

7.19.3.10 const char* OBIX_CONTRACT_ERR_PERMISSION = "obix:PermissionErr"

URI of the *PermissionErr* error contract.

Definition at line 35 of file obix_utils.c.

Referenced by obix_server_write().

7.19.3.11 const char* OBIX_CONTRACT_ERR_UNSUPPORTED = "obix:UnsupportedErr"

URI of the *UnsupportedErr* error contract.

Definition at line 34 of file obix_utils.c.

Referenced by handlerError(), and obix_fcgi_handleRequest().

7.19.3.12 const char* OBIX_NAME_BATCH = "batch"

Name of *batch* operation in the Lobby object.

Definition at line 53 of file obix_utils.c.

Referenced by http_openConnection().

7.19.3.13 const char* OBIX_NAME_SIGN_UP = "signUp"

Name of *signUp* operation in the Lobby object.

Definition at line 52 of file obix_utils.c.

Referenced by http_openConnection().

7.19.3.14 const char* OBIX_NAME_WATCH_ADD = "add"

Name of the *Watch.add* operation.

Definition at line 56 of file obix_utils.c.

Referenced by createWatch().

7.19.3.15 const char* OBIX_NAME_WATCH_DELETE = "delete"

Name of the *Watch.delete* operation.

Definition at line 60 of file obix_utils.c.

Referenced by createWatch().

7.19.3.16 const char* OBIX_NAME_WATCHLEASE = "lease"

Name of the *Watch.lease* parameter.

Definition at line 61 of file obix_utils.c.

Referenced by setWatchLeaseTime().

7.19.3.17 const char* OBIX_NAME_WATCH_POLL_WAIT_INTERVAL = "pollWaitInterval"

Name of the *Watch.pollWaitInterval* object.

Definition at line 62 of file obix_utils.c.

7.19.3.18 const char* OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MAX = "max"

Name of the *Watch.pollWaitInterval.max* parameter.

Definition at line 64 of file obix_utils.c.

Referenced by setWatchPollWaitTime().

7.19.3.19 const char* OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MIN = "min"

Name of the *Watch.pollWaitInterval.min* parameter.

Definition at line 63 of file obix_utils.c.

Referenced by setWatchPollWaitTime().

7.19.3.20 const char* OBIX_NAME_WATCH_POLLCHANGES = "pollChanges"

Name of the *Watch.pollChanges* operation.

Definition at line 58 of file obix_utils.c.

Referenced by createWatch().

7.19.3.21 const char* OBIX_NAME_WATCH_POLLREFRESH = "pollRefresh"

Name of the *Watch.pollRefresh* operation.

Definition at line 59 of file obix_utils.c.

7.19.3.22 const char* OBIX_NAME_WATCH_REMOVE = "remove"

Name of the *Watch.remove* operation.

Definition at line 57 of file obix_utils.c.

Referenced by createWatch().

7.19.3.23 const char* OBIX_NAME_WATCH_SERVICE = "watchService"

Name of the Watch Service in the Lobby object.

Definition at line 54 of file obix_utils.c.

Referenced by http_openConnection().

7.19.3.24 const char* OBIX_NAME_WATCH_SERVICE_MAKE = "make"

Name of the *watchService.make* operation.

Definition at line 55 of file obix_utils.c.

Referenced by http_openConnection().

7.19.3.25 const char* OBIX_OBJ = "obj"

oBIX Object (*obj*)

Definition at line 37 of file obix_utils.c.

Referenced by loadDeviceData(), sendFallEventUpdate(), and targetsListener().

7.19.3.26 const char* OBIX_OBJ_ABSTIME = "abstime"

oBIX Absolute Time (*abstime*)

Definition at line 47 of file obix_utils.c.

7.19.3.27 const char* OBIX_OBJ_BOOL = "bool"

oBIX Boolean (*bool*)

Definition at line 42 of file obix_utils.c.

7.19.3.28 const char* OBIX_OBJ_ENUM = "enum"

oBIX Enumeration (*enum*)

Definition at line 46 of file obix_utils.c.

7.19.3.29 const char* OBIX_OBJ_ERR = "err"

oBIX Error (*err*)

Definition at line 41 of file obix_utils.c.

Referenced by checkResponseElement(), and parseWatchOut().

7.19.3.30 const char* OBIX_OBJ_FEED = "feed"

oBIX Feed (*feed*)

Definition at line 50 of file obix_utils.c.

7.19.3.31 const char* OBIX_OBJ_INT = "int"

oBIX Integer (*int*)

Definition at line 43 of file obix_utils.c.

7.19.3.32 const char* OBIX_OBJ_LIST = "list"

oBIX List (*list*)

Definition at line 40 of file obix_utils.c.

Referenced by parseWatchOut(), and testCycle().

7.19.3.33 const char* OBIX_OBJ_NULL_TEMPLATE = "<obj null=\\"true\\"/>"

String which represents oBIX *NULL* object.

Definition at line 66 of file obix_utils.c.

Referenced by handlerWatchDelete(), and handlerWatchRemove().

7.19.3.34 const char* OBIX_OBJ_OP = "op"

oBIX Operation (*op*)

Definition at line 39 of file obix_utils.c.

Referenced by obix_server_invoke(), and obixWatch_createWatchItem().

7.19.3.35 const char* OBIX_OBJ_REAL = "real"

oBIX Real (*real*)

Definition at line 44 of file obix_utils.c.

7.19.3.36 const char* OBIX_OBJ_REF = "ref"

oBIX Reference (*ref*)

Definition at line 38 of file obix_utils.c.

Referenced by getNodeByHrefRecursive(), and putDeviceReference().

7.19.3.37 const char* OBIX_OBJ_RELTIME = "reltime"

oBIX Relative Duration of Time (*reltime*)

Definition at line 48 of file obix_utils.c.

Referenced by getLeaseTime().

7.19.3.38 const char* OBIX_OBJ_STR = "str"

oBIX String (*str*)

Definition at line 45 of file obix_utils.c.

7.19.3.39 const char* OBIX_OBJ_URI = "uri"

oBIX URI (*uri*)

Definition at line 49 of file obix_utils.c.

Referenced by processWatchIn().

7.19.3.40 const char* XML_FALSE = "false"

String representation of boolean *false* value.

Definition at line 78 of file obix_utils.c.

Referenced by config_getTagAttrBoolValue(), resetListener(), and target_sendUpdate().

7.19.3.41 const char* XML_TRUE = "true"

String representation of boolean *true* value.

Definition at line 77 of file obix_utils.c.

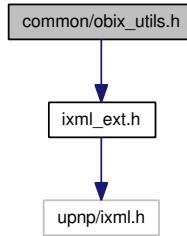
Referenced by config_getTagAttrBoolValue(), sendFallEventUpdate(), target_sendUpdate(), and xmldb_updateDOM().

7.20 common/obix_utils.h File Reference

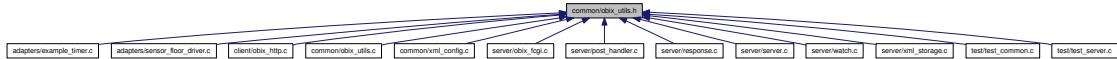
Contains oBIX keywords (object names, contracts, facets, etc) and some utility functions.

```
#include <ixml_ext.h>
```

Include dependency graph for obix_utils.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum RELTIME_FORMAT {
 RELTIME_SEC, RELTIME_MIN, RELTIME_HOUR, RELTIME_DAY,
 RELTIME_MONTH, RELTIME_YEAR }

Specifies a format of reltime value, generated by [obix_reltimes_fromLong](#).

Functions

- BOOL [obix_obj_implementsContract](#) (IXML_Element *obj, const char *contract)
Checks whether oBIX object implements specified contract.
- char * [obix_reltimes_fromLong](#) (long duration, [RELTIME_FORMAT](#) format)
Generates reltime value from the provided time in milliseconds.
- int [obix_reltimes_parseToLong](#) (const char *str, long *duration)
Parses string value of reltime object and returns corresponding time in milliseconds.

Variables

- const char * **OBIX_OBJ_NULL_TEMPLATE**
String which represents oBIX NULL object.
- const char * **XML_FALSE**
String representation of boolean false value.
- const char * **XML_TRUE**
String representation of boolean true value.

oBIX Object Attributes and Facets

- const char * **OBIX_ATTR_DISPLAY**
oBIX facet "display".
- const char * **OBIX_ATTR_DISPLAY_NAME**
oBIX facet "displayName".
- const char * **OBIX_ATTR_HREF**
Object attribute "href".
- const char * **OBIX_ATTR_IS**
Object attribute "is".
- const char * **OBIX_ATTR_NAME**
Object attribute "name".
- const char * **OBIX_ATTR_NULL**
Object attribute "null".
- const char * **OBIX_ATTR_VAL**
Object attribute "val".
- const char * **OBIX_ATTR_WRITABLE**
oBIX facet "writable".

oBIX Error Contracts' URIs

Can be used to define the error type returned by an oBIX server.

- const char * **OBIX_CONTRACT_ERR_BAD_URI**
URI of the BadUriErr error contract.
- const char * **OBIX_CONTRACT_ERR_PERMISSION**
URI of the PermissionErr error contract.
- const char * **OBIX_CONTRACT_ERR_UNSUPPORTED**
URI of the UnsupportedErr error contract.

oBIX Object Names

Object names which are used in oBIX contracts.

- const char * **OBIX_NAME_BATCH**
Name of batch operation in the Lobby object.
- const char * **OBIX_NAME_SIGN_UP**
Name of signUp operation in the Lobby object.
- const char * **OBIX_NAME_WATCH_ADD**
Name of the Watch.add operation.
- const char * **OBIX_NAME_WATCH_DELETE**
Name of the Watch.delete operation.
- const char * **OBIX_NAME_WATCHLEASE**
Name of the Watch.lease parameter.
- const char * **OBIX_NAME_WATCH_POLL_WAIT_INTERVAL**
Name of the Watch.pollWaitInterval object.
- const char * **OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MAX**
Name of the Watch.pollWaitInterval.max parameter.
- const char * **OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MIN**
Name of the Watch.pollWaitInterval.min parameter.
- const char * **OBIX_NAME_WATCH_POLLCHANGES**
Name of the Watch.pollChanges operation.
- const char * **OBIX_NAME_WATCH_POLLREFRESH**
Name of the Watch.pollRefresh operation.
- const char * **OBIX_NAME_WATCH_REMOVE**
Name of the Watch.remove operation.
- const char * **OBIX_NAME_WATCH_SERVICE**
Name of the Watch Service in the Lobby object.
- const char * **OBIX_NAME_WATCH_SERVICE_MAKE**
Name of the watchService.make operation.

oBIX Object Types (XML Element Types)

- const char * **OBIX_OBJ**
oBIX Object (obj)
- const char * **OBIX_OBJ_ABSTIME**
oBIX Absolute Time (abstime)
- const char * **OBIX_OBJ_BOOL**
oBIX Boolean (bool)
- const char * **OBIX_OBJ_ENUM**
oBIX Enumeration (enum)
- const char * **OBIX_OBJ_ERR**

oBIX Error (err)

- `const char * OBIX_OBJ_FEED`
oBIX Feed (feed)
- `const char * OBIX_OBJ_INT`
oBIX Integer (int)
- `const char * OBIX_OBJ_LIST`
oBIX List (list)
- `const char * OBIX_OBJ_OP`
oBIX Operation (op)
- `const char * OBIX_OBJ_REAL`
oBIX Real (real)
- `const char * OBIX_OBJ_REF`
oBIX Reference (ref)
- `const char * OBIX_OBJ_RELTIME`
oBIX Relative Duration of Time (reltime)
- `const char * OBIX_OBJ_STR`
oBIX String (str)
- `const char * OBIX_OBJ_URI`
oBIX URI (uri)

7.20.1 Detailed Description

Contains oBIX keywords (object names, contracts, facets, etc) and some utility functions.

The list of keywords is not complete and should be expanded every time when something new is needed. In most cases a common oBIX client application doesn't need to use any of those, because [oBIX Client API](#) hides all oBIX syntax (except case with registering new device data).

Read more about oBIX from <http://obix.org/>

Definition in file [obix_utils.h](#).

7.20.2 Enumeration Type Documentation

7.20.2.1 enum RELTIME_FORMAT

Specifies a format of *reltime* value, generated by [obix_reltimes_fromLong](#).

Enumerator:

`RELTIME_SEC`
`RELTIME_MIN`
`RELTIME_HOUR`

RELTIME_DAY
RELTIME_MONTH
RELTIME_YEAR

Definition at line 149 of file obix_utils.h.

7.20.3 Function Documentation

7.20.3.1 BOOL obix_obj_implementsContract (IXML_Element * *obj*, const char * *contract*)

Checks whether oBIX object implements specified contract.

Object implements a contract when contract's URI is listed in object's *is* attribute.

Parameters:

obj XML DOM structure representing an oBIX object.
contract URI of the contract which should be checked.

Returns:

TRUE if the object implements specified contract, FALSE otherwise.

Definition at line 422 of file obix_utils.c.

References FALSE, OBIX_ATTR_IS, and TRUE.

Referenced by handlerBatch(), http_unregisterListener(), parseTarget(), parseWatchOut(), removeWatch(), and setWatchPollWaitTime().

7.20.3.2 char* obix_reltimes_fromLong (long *duration*, RELTIME_FORMAT *format*)

Generates *reltime* value from the provided time in milliseconds.

Parameters:

duration Time in milliseconds which should be converted.
format Format of the generated *reltime* value. Specifies the maximum time component for the output value. For example, converting 2 minutes with *RELTIME_MIN* will result in "PT2M"; *RELTIME_SEC* - "PT120S".

Returns:

String, which represents provided time in xs:*duration* format, or NULL if memory allocation failed.

Definition at line 279 of file obix_utils.c.

References RELTIME_DAY, RELTIME_HOUR, and RELTIME_MIN.

Referenced by setWatchTimeParam(), testObix_reltimes_fromLong(), and timerTask().

7.20.3.3 int obix_reltimes_parseToLong (const char * str, long * duration)

Parses string value of *reltime* object and returns corresponding time in milliseconds.

Follows *xs:duration* format.

Note:

Durations which can overload *long* variable are not parsed.

Parameters:

str String value of a *reltime* object which should be parsed.

duration If parsing is successful than the parsed value will be written there.

Returns:

- 0 - Operation completed successfully;
- -1 - Parsing error (provided string has bad format);
- -2 - Provided *reltime* value is bigger than or equal to 24 days (The maximum possible value is "P23DT23H59M59.999S"). Also this error code is returned when the input value is not normalized: If some of time components (e.g. hours) presents, than all smaller components (minutes and seconds) should represent less time than previous component. For example "PT60M" and "PT60S" are allowed, but "PT1H60M" and "PT1H60S" are not.

Parses positive integer. Does the same as `strtol(startPos, endPos, 10)`, but assumes that negative result is an error. Thus returns -1 instead of 0, when nothing is parsed. Also sets flag `parsedSomething` if parsing did not fail.

Definition at line 80 of file `obix_utils.c`.

Referenced by `getLeaseTime()`, `obixWatch_processTimeUpdates()`, and `testObix_reltimes_parseToLong()`.

7.20.4 Variable Documentation

7.20.4.1 const char* OBIX_ATTR_DISPLAY

oBIX facet "*display*".

Definition at line 74 of file `obix_utils.c`.

Referenced by `obix_server_generateObixErrorMessage()`, and `putDeviceReference()`.

7.20.4.2 const char* OBIX_ATTR_DISPLAY_NAME

oBIX facet "*displayName*".

Definition at line 75 of file `obix_utils.c`.

Referenced by `loadDeviceData()`, `obix_server_generateObixErrorMessage()`, and `putDeviceReference()`.

7.20.4.3 const char* OBIX_ATTR_HREF

Object attribute "*href*".

Definition at line 70 of file obix_utils.c.

Referenced by checkNode(), getNodeByHrefRecursive(), handlerSignUp(), handlerWatchServiceMake(), http_registerDevice(), insertServerAddress(), ixmlelement_getFullHref(), loadDeviceData(), normalizeObixDocument(), normalizeUri(), obix_server_generateResponse(), obixWatch_create(), parseWatchOut(), and putDeviceReference().

7.20.4.4 const char* OBIX_ATTR_IS

Object attribute "*is*".

Definition at line 68 of file obix_utils.c.

Referenced by obix_obj_implementsContract(), and obix_server_generateObixErrorMessage().

7.20.4.5 const char* OBIX_ATTR_NAME

Object attribute "*name*".

Definition at line 69 of file obix_utils.c.

Referenced by getObjectUri(), loadDeviceData(), parseTarget(), parseWatchOut(), and putDeviceReference().

7.20.4.6 const char* OBIX_ATTR_NULL

Object attribute "*null*".

Definition at line 72 of file obix_utils.c.

7.20.4.7 const char* OBIX_ATTR_VAL

Object attribute "*val*".

Definition at line 71 of file obix_utils.c.

Referenced by getLeaseTime(), getUriSet(), handlerBatch(), http_initConnection(), obixWatch_processTimeUpdates(), obixWatch_updateMeta(), parseElementValue(), parseTarget(), parseWatchOut(), sendFallEventUpdate(), xmldb_putMeta(), and xmldb_updateDOM().

7.20.4.8 const char* OBIX_ATTR_WRITABLE

oBIX facet "*writable*".

Definition at line 73 of file obix_utils.c.

Referenced by xmlDb_updateDOM().

7.20.4.9 const char* OBIX_CONTRACT_ERR_BAD_URI

URI of the *BadUriErr* error contract.

Definition at line 33 of file obix_utils.c.

Referenced by handlerWatchAdd(), http_unregisterListener(), obix_server_invoke(), obix_server_read(), obix_server_write(), parseWatchOut(), and removeWatch().

7.20.4.10 const char* OBIX_CONTRACT_ERR_PERMISSION

URI of the *PermissionErr* error contract.

Definition at line 35 of file obix_utils.c.

Referenced by obix_server_write().

7.20.4.11 const char* OBIX_CONTRACT_ERR_UNSUPPORTED

URI of the *UnsupportedErr* error contract.

Definition at line 34 of file obix_utils.c.

Referenced by handlerError(), and obix_fcgi_handleRequest().

7.20.4.12 const char* OBIX_NAME_BATCH

Name of *batch* operation in the Lobby object.

Definition at line 53 of file obix_utils.c.

Referenced by http_openConnection().

7.20.4.13 const char* OBIX_NAME_SIGN_UP

Name of *signUp* operation in the Lobby object.

Definition at line 52 of file obix_utils.c.

Referenced by http_openConnection().

7.20.4.14 const char* OBIX_NAME_WATCH_ADD

Name of the *Watch.add* operation.

Definition at line 56 of file obix_utils.c.

Referenced by `createWatch()`.

7.20.4.15 const char* OBIX_NAME_WATCH_DELETE

Name of the *Watch.delete* operation.

Definition at line 60 of file obix_utils.c.

Referenced by `createWatch()`.

7.20.4.16 const char* OBIX_NAME_WATCHLEASE

Name of the *Watch.lease* parameter.

Definition at line 61 of file obix_utils.c.

Referenced by `setWatchLeaseTime()`.

7.20.4.17 const char* OBIX_NAME_WATCH_POLL_WAIT_INTERVAL

Name of the *Watch.pollWaitInterval* object.

Definition at line 62 of file obix_utils.c.

7.20.4.18 const char* OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MAX

Name of the *Watch.pollWaitInterval.max* parameter.

Definition at line 64 of file obix_utils.c.

Referenced by `setWatchPollWaitTime()`.

7.20.4.19 const char* OBIX_NAME_WATCH_POLL_WAIT_INTERVAL_MIN

Name of the *Watch.pollWaitInterval.min* parameter.

Definition at line 63 of file obix_utils.c.

Referenced by `setWatchPollWaitTime()`.

7.20.4.20 const char* OBIX_NAME_WATCH_POLLCHANGES

Name of the *Watch.pollChanges* operation.

Definition at line 58 of file obix_utils.c.

Referenced by `createWatch()`.

7.20.4.21 const char* OBIX_NAME_WATCH_POLLREFRESH

Name of the *Watch.pollRefresh* operation.

Definition at line 59 of file obix_utils.c.

7.20.4.22 const char* OBIX_NAME_WATCH_REMOVE

Name of the *Watch.remove* operation.

Definition at line 57 of file obix_utils.c.

Referenced by `createWatch()`.

7.20.4.23 const char* OBIX_NAME_WATCH_SERVICE

Name of the Watch Service in the Lobby object.

Definition at line 54 of file obix_utils.c.

Referenced by `http.openConnection()`.

7.20.4.24 const char* OBIX_NAME_WATCH_SERVICE_MAKE

Name of the *watchService.make* operation.

Definition at line 55 of file obix_utils.c.

Referenced by `http.openConnection()`.

7.20.4.25 const char* OBIX_OBJ

oBIX Object (*obj*)

Definition at line 37 of file obix_utils.c.

Referenced by `loadDeviceData()`, `sendFallEventUpdate()`, and `targetsListener()`.

7.20.4.26 const char* OBIX_OBJ_ABSTIME

oBIX Absolute Time (*abstime*)

Definition at line 47 of file obix_utils.c.

7.20.4.27 const char* OBIX_OBJ_BOOL

oBIX Boolean (*bool*)

Definition at line 42 of file obix_utils.c.

7.20.4.28 const char* OBIX_OBJ_ENUM

oBIX Enumeration (*enum*)

Definition at line 46 of file obix_utils.c.

7.20.4.29 const char* OBIX_OBJ_ERR

oBIX Error (*err*)

Definition at line 41 of file obix_utils.c.

Referenced by checkResponseElement(), and parseWatchOut().

7.20.4.30 const char* OBIX_OBJ_FEED

oBIX Feed (*feed*)

Definition at line 50 of file obix_utils.c.

7.20.4.31 const char* OBIX_OBJ_INT

oBIX Integer (*int*)

Definition at line 43 of file obix_utils.c.

7.20.4.32 const char* OBIX_OBJ_LIST

oBIX List (*list*)

Definition at line 40 of file obix_utils.c.

Referenced by parseWatchOut(), and testCycle().

7.20.4.33 const char* OBIX_OBJ_NULL_TEMPLATE

String which represents oBIX *NULL* object.

Definition at line 66 of file obix_utils.c.

Referenced by handlerWatchDelete(), and handlerWatchRemove().

7.20.4.34 const char* OBIX_OBJ_OP

oBIX Operation (*op*)

Definition at line 39 of file obix_utils.c.

Referenced by obix_server_invoke(), and obixWatch_createWatchItem().

7.20.4.35 const char* OBIX_OBJ_REAL

oBIX Real (*real*)

Definition at line 44 of file obix_utils.c.

7.20.4.36 const char* OBIX_OBJ_REF

oBIX Reference (*ref*)

Definition at line 38 of file obix_utils.c.

Referenced by getNodeByHrefRecursive(), and putDeviceReference().

7.20.4.37 const char* OBIX_OBJ_RELTIME

oBIX Relative Duration of Time (*reltime*)

Definition at line 48 of file obix_utils.c.

Referenced by getLeaseTime().

7.20.4.38 const char* OBIX_OBJ_STR

oBIX String (*str*)

Definition at line 45 of file obix_utils.c.

7.20.4.39 const char* OBIX_OBJ_URI

oBIX URI (*uri*)

Definition at line 49 of file obix_utils.c.

Referenced by processWatchIn().

7.20.4.40 const char* XML_FALSE

String representation of boolean *false* value.

Definition at line 78 of file obix_utils.c.

Referenced by config_getTagAttrBoolValue(), resetListener(), and target_sendUpdate().

7.20.4.41 const char* XML_TRUE

String representation of boolean *true* value.

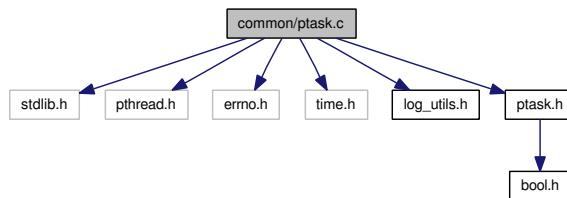
Definition at line 77 of file obix_utils.c.

Referenced by config_getTagAttrBoolValue(), sendFallEventUpdate(), target_sendUpdate(), and xmldb_updateDOM().

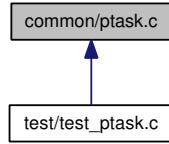
7.21 common/ptask.c File Reference

```
#include <stdlib.h>
#include <pthread.h>
#include <errno.h>
#include <time.h>
#include <log_utils.h>
#include "ptask.h"
```

Include dependency graph for ptask.c:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `_Periodic_Task`
- struct `_Task_Thread`

TypeDefs

- typedef struct `_Periodic_Task` `Periodic_Task`

Functions

- static int `generateId` (`Task_Thread` *thread)
- static `Periodic_Task` * `periodicTask_create` (`Task_Thread` *thread, `periodic_task` task, void *arg, long period, int executeTimes)
- static void `periodicTask_deleteRecursive` (`Periodic_Task` *ptask)
- static void `periodicTask_execute` (`Task_Thread` *thread, `Periodic_Task` *ptask)
- static void `periodicTask_free` (`Periodic_Task` *ptask)
- static void `periodicTask_generateNextExecTime` (`Periodic_Task` *ptask)
- static `Periodic_Task` * `periodicTask_get` (`Task_Thread` *thread, int id)
- static `Periodic_Task` * `periodicTask_getClosest` (`Task_Thread` *thread)
- static void `periodicTask_removeFromList` (`Task_Thread` *thread, `Periodic_Task` *ptask)
- static void `periodicTask_resetExecTime` (`Periodic_Task` *ptask)
- static void `periodicTask_setPeriod` (`Periodic_Task` *ptask, long period, int executeTimes)
- int `ptask_cancel` (`Task_Thread` *thread, int taskId, `BOOL` wait)

Removes task from the scheduled list.
- int `ptask_dispose` (`Task_Thread` *thread, `BOOL` wait)

Releases resources allocated for the provided `Task_Thread` instance.
- `Task_Thread` * `ptask_init` ()

Creates new instance of `Task_Thread`.
- `BOOL` `ptask_isScheduled` (`Task_Thread` *thread, int taskId)

Checks whether the task with provided id is scheduled for execution in the thread.
- int `ptask_reschedule` (`Task_Thread` *thread, int taskId, long period, int executeTimes, `BOOL` add)

Sets new execution period for the specified task.
- int `ptask_reset` (`Task_Thread` *thread, int taskId)

Resets time until the next execution of the specified task.

- int [ptask_schedule](#) (Task_Thread *thread, [periodic_task](#) task, void *arg, long period, int executeTimes)
Schedules new task for execution.
- static void [stopTask](#) (void *arg)
- static void * [threadCycle](#) (void *arg)

Utility methods for work with @a timespec structure @{

- static int [timespec_add](#) (struct timespec *time1, const struct timespec *time2)
- static int [timespec_cmp](#) (struct timespec *time1, struct timespec *time2)
- static void [timespec_copy](#) (struct timespec *dest, const struct timespec *source)
- static void [timespec_fromMillis](#) (struct timespec *time, long millis)

7.21.1 Detailed Description

[Todo](#)

Add description

Author:

Andrey Litvinov

Version:

1.0

Definition in file [ptask.c](#).

7.21.2 Typedef Documentation

7.21.2.1 [typedef struct _Periodic_Task Periodic_Task](#)

7.21.3 Function Documentation

7.21.3.1 [static int generateId \(Task_Thread * thread\) \[static\]](#)

Definition at line 129 of file [ptask.c](#).

References [_Task_Thread::id_gen](#).

Referenced by [periodicTask_create\(\)](#).

7.21.3.2 [static Periodic_Task* periodicTask_create \(Task_Thread * thread, periodic_task task, void * arg, long period, int executeTimes\) \[static\]](#)

Definition at line 154 of file [ptask.c](#).

References `_Periodic_Task::arg`, `FALSE`, `generateId()`, `_Periodic_Task::id`, `_Periodic_Task::isCancelled`, `_Periodic_Task::isExecuting`, `log_error`, `_Periodic_Task::next`, `periodicTask_resetExecTime()`, `periodicTask_setPeriod()`, `_Periodic_Task::prev`, and `_Periodic_Task::task`.

Referenced by `ptask_schedule()`.

7.21.3.3 static void periodicTask_deleteRecursive (`Periodic_Task * ptask`) [static]

Definition at line 569 of file `ptask.c`.

References `_Periodic_Task::next`, and `periodicTask_free()`.

Referenced by `stopTask()`, and `test_ptask()`.

7.21.3.4 static void periodicTask_execute (`Task_Thread * thread`, `Periodic_Task * ptask`) [static]

Definition at line 201 of file `ptask.c`.

References `_Periodic_Task::arg`, `_Periodic_Task::executeTimes`, `FALSE`, `_Periodic_Task::isCancelled`, `_Periodic_Task::isExecuting`, `periodicTask_free()`, `periodicTask_generateNextExecTime()`, `periodicTask_removeFromList()`, `_Periodic_Task::task`, `_Task_Thread::taskExecuted`, `_Task_Thread::taskListMutex`, and `TRUE`.

Referenced by `testGetClosestTask()`, and `threadCycle()`.

7.21.3.5 static void periodicTask_free (`Periodic_Task * ptask`) [static]

Definition at line 176 of file `ptask.c`.

Referenced by `periodicTask_deleteRecursive()`, `periodicTask_execute()`, and `ptask_cancel()`.

7.21.3.6 static void periodicTask_generateNextExecTime (`Periodic_Task * ptask`) [static]

Definition at line 134 of file `ptask.c`.

References `_Periodic_Task::nextScheduledTime`, `_Periodic_Task::period`, and `timespec_add()`.

Referenced by `periodicTask_execute()`, and `periodicTask_resetExecTime()`.

7.21.3.7 static `Periodic_Task*` periodicTask_get (`Task_Thread * thread`, `int id`) [static]

Definition at line 240 of file `ptask.c`.

References `_Periodic_Task::id`, `_Periodic_Task::next`, and `_Task_Thread::taskList`.

Referenced by `ptask_cancel()`, `ptask_isScheduled()`, `ptask_reschedule()`, `ptask_reset()`, and `testPtaskReschedule()`.

7.21.3.8 static Periodic_Task* periodicTask_getClosest (Task_Thread * *thread*) [static]

Definition at line 429 of file ptask.c.

References `_Periodic_Task::next`, `_Periodic_Task::nextScheduledTime`, `_Task_Thread::taskList`, and `timespec_cmp()`.

Referenced by `testGetClosestTask()`, and `threadCycle()`.

7.21.3.9 static void periodicTask_removeFromList (Task_Thread * *thread*, Periodic_Task * *ptask*) [static]

Definition at line 181 of file ptask.c.

References `_Periodic_Task::next`, `_Periodic_Task::prev`, and `_Task_Thread::taskList`.

Referenced by `periodicTask_execute()`, and `ptask_cancel()`.

7.21.3.10 static void periodicTask_resetExecTime (Periodic_Task * *ptask*) [static]

Definition at line 147 of file ptask.c.

References `_Periodic_Task::nextScheduledTime`, and `periodicTask_generateNextExecTime()`.

Referenced by `periodicTask_create()`, `ptask_reschedule()`, and `ptask_reset()`.

7.21.3.11 static void periodicTask_setPeriod (Periodic_Task * *ptask*, long *period*, int *executeTimes*) [static]

Definition at line 141 of file ptask.c.

References `_Periodic_Task::executeTimes`, `_Periodic_Task::period`, and `timespec_fromMillis()`.

Referenced by `periodicTask_create()`, and `ptask_reschedule()`.

7.21.3.12 int ptask_cancel (Task_Thread * *thread*, int *taskId*, BOOL *wait*)

Removes task from the scheduled list.

Parameters:

thread Thread in which task is scheduled.

taskId ID of the task to be removed.

wait When task is being executed it can be canceled only after execution is completed. This parameter defines whether the function should wait until the task is really canceled, or it can just mark the task as canceled, which guarantees that the task will be removed as soon as the current execution is completed. In case when this function is called while the task is not executed *wait* argument makes no difference.

Returns:

- *0* on success;
- *-1* if task with provided ID is not found.

Definition at line 396 of file ptask.c.

References `_Periodic_Task::isCancelled`, `_Periodic_Task::isExecuting`, `periodicTask_free()`, `periodicTask_get()`, `periodicTask_removeFromList()`, `_Task_Thread::taskExecuted`, `_Task_Thread::taskListMutex`, `_Task_Thread::taskListUpdated`, and `TRUE`.

Referenced by `main()`, `obixWatch_delete()`, `removeWatch()`, `testPeriodicTask()`, `testPtaskReschedule()`, `threadCycle()`, and `watchPollTask()`.

7.21.3.13 int ptask_dispose (Task_Thread * *thread*, BOOL *wait*)

Releases resources allocated for the provided `Task_Thread` instance.

All scheduled tasks are canceled.

Parameters:

- `thread` Pointer to the `Task_Thread` to be freed.
`wait` If `TRUE` than the method will block and wait until specified thread is really disposed. Otherwise, method will only schedule asynchronous disposing of the thread.

Returns:

0 on success, negative error code otherwise.

Definition at line 599 of file ptask.c.

References `ptask_schedule()`, `stopTask()`, and `_Task_Thread::thread`.

Referenced by `http_dispose()`, `main()`, `obixWatch_dispose()`, and `taskStopThread()`.

7.21.3.14 Task_Thread* ptask_init ()

Creates new instance of `Task_Thread`.

Returns:

Pointer to the new instance of `Task_Thread`, or `NULL` if some error occurred.

Definition at line 518 of file ptask.c.

References `_Task_Thread::id_gen`, `log_error`, `_Task_Thread::taskExecuted`, `_Task_Thread::taskList`, `_Task_Thread::taskListMutex`, `_Task_Thread::taskListUpdated`, `_Task_Thread::thread`, and `threadCycle()`.

Referenced by `http_init()`, `main()`, `obixWatch_init()`, and `test_ptask_byHands()`.

7.21.3.15 BOOL ptask_isScheduled (Task_Thread * *thread*, int *taskId*)

Checks whether the task with provided id is scheduled for execution in the thread.

Parameters:

- thread*** Thread where the task should be searched for.
taskId Task id which is searched for.

Returns:

TRUE if the task with specified *taskId* is scheduled, **FALSE** otherwise.

Definition at line 364 of file ptask.c.

References FALSE, periodicTask_get(), and TRUE.

7.21.3.16 int ptask_reschedule (Task_Thread * *thread*, int *taskId*, long *period*, int *executeTimes*, BOOL *add*)

Sets new execution period for the specified task.

Parameters:

- thread*** Thread in which the task is scheduled.
taskId Id of the scheduled task.
period New time interval in milliseconds (or time which will be added to the current task period).
executeTimes Defines how many times (min. 1) the task should be executed. If **EXECUTE_INDEFINITE** is provided than the task is executed until **ptask_cancel()** is called with corresponding task ID.
add Defines whether time provided in *period* argument will be used as new execution period, or will be added to the current one.

Note:

When *add* is set to **TRUE**, *period* will be also added to the next execution time, but when *add* is **FALSE** the next execution will be (current time + *period*).

Returns:

0 on success, negative error code otherwise.

Definition at line 309 of file ptask.c.

References **_Periodic_Task::executeTimes**, **_Periodic_Task::nextScheduledTime**, **_Periodic_Task::period**, **periodicTask_get()**, **periodicTask_resetExecTime()**, **periodicTask_setPeriod()**, **_Task_Thread::taskListMutex**, **_Task_Thread::taskListUpdated**, **timespec_add()**, **timespec_copy()**, and **timespec_fromMillis()**.

Referenced by **obixWatch_deleteHelper()**, **obixWatch_notifyPollTask()**, **obixWatch_resetLeaseTimer()**, **removeWatch()**, and **testPtaskReschedule()**.

7.21.3.17 int ptask_reset (Task_Thread * *thread*, int *taskId*)

Resets time until the next execution of the specified task.

The next execution time will be current time + *period* provided when the task was scheduled. If the *period* needs to be changed than use [ptask_reschedule\(\)](#) instead.

Parameters:

thread Thread where the task is scheduled.

taskId Id of the task whose execution time should be reset.

Returns:

0 on success, negative error code otherwise.

Definition at line 376 of file ptask.c.

References [periodicTask_get\(\)](#), [periodicTask_resetExecTime\(\)](#), [_Task_Thread::taskListMutex](#), and [_Task_Thread::taskListUpdated](#).

Referenced by [checkTargets\(\)](#), [obixWatch_resetLeaseTimer\(\)](#), [resetListener\(\)](#), and [targetRemoveTask\(\)](#).

7.21.3.18 int ptask_schedule (Task_Thread * *thread*, periodic_task *task*, void * *arg*, long *period*, int *executeTimes*)

Schedules new task for execution.

Parameters:

thread Thread in which the task will be executed.

task Task which should be scheduled.

arg Argument which will be passed to the task function each time when it is invoked.

period Time interval in milliseconds, which defines how often the task will be executed.

executeTimes Defines how many times (min. 1) the task should be executed. If [EXECUTE_INDEFINITE](#) is provided than the task is executed until [ptask_cancel\(\)](#) with corresponding task ID is called.

Returns:

- >0 - ID of the scheduled task.
- <0 - Error code.

Definition at line 250 of file ptask.c.

References [_Periodic_Task::id](#), [_Periodic_Task::next](#), [_Periodic_Task::period](#), [periodicTask_create\(\)](#), [_Periodic_Task::prev](#), [_Task_Thread::taskList](#), [_Task_Thread::taskListMutex](#), [_Task_Thread::taskListUpdated](#), and [timespec_cmp\(\)](#).

Referenced by [addListener\(\)](#), [main\(\)](#), [obixWatch_create\(\)](#), [obixWatch_holdPoll\(\)](#), [ptask_dispose\(\)](#), [target_get\(\)](#), [test_ptask_byHands\(\)](#), [testPeriodicTask\(\)](#), and [testPtaskReschedule\(\)](#).

7.21.3.19 static void stopTask (void * *arg*) [static]

Definition at line 581 of file ptask.c.

References `log_debug`, `periodicTask_deleteRecursive()`, `_Task_Thread::taskExecuted`, `_Task_Thread::taskList`, `_Task_Thread::taskListMutex`, and `_Task_Thread::taskListUpdated`.

Referenced by `ptask_dispose()`.

7.21.3.20 static void* threadCycle (void * *arg*) [static]

Definition at line 452 of file ptask.c.

References `FALSE`, `_Periodic_Task::id`, `log_debug`, `log_error`, `log_warning`, `_Periodic_Task::nextScheduledTime`, `periodicTask_execute()`, `periodicTask_getClosest()`, `ptask_cancel()`, `_Task_Thread::taskListMutex`, and `_Task_Thread::taskListUpdated`.

Referenced by `ptask_init()`.

7.21.3.21 static int timespec_add (struct timespec * *time1*, const struct timespec * *time2*) [static]

Definition at line 69 of file ptask.c.

Referenced by `periodicTask_generateNextExecTime()`, `ptask_reschedule()`, and `testPtaskReschedule()`.

7.21.3.22 static int timespec_cmp (struct timespec * *time1*, struct timespec * *time2*) [static]

Definition at line 99 of file ptask.c.

Referenced by `periodicTask_getClosest()`, `ptask_schedule()`, and `testPtaskReschedule()`.

7.21.3.23 static void timespec_copy (struct timespec * *dest*, const struct timespec * *source*) [static]

Definition at line 93 of file ptask.c.

Referenced by `ptask_reschedule()`, and `testPtaskReschedule()`.

7.21.3.24 static void timespec_fromMillis (struct timespec * *time*, long *millis*) [static]

Definition at line 63 of file ptask.c.

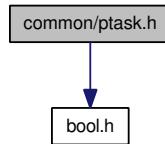
Referenced by `periodicTask_setPeriod()`, and `ptask_reschedule()`.

7.22 common/ptask.h File Reference

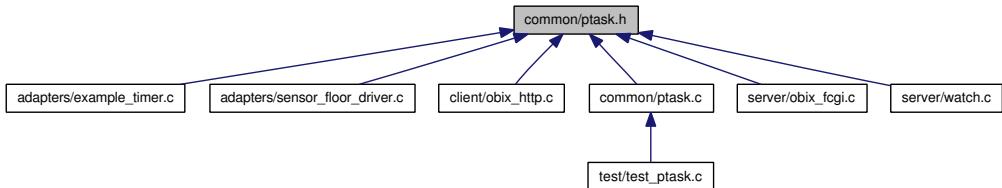
Periodic Task - tool for asynchronous task execution.

```
#include "bool.h"
```

Include dependency graph for ptask.h:



This graph shows which files directly or indirectly include this file:



Defines

- `#define EXECUTE_INDEFINITE -1`
Specifies that the task should be executed indefinite number of times (until `ptask_cancel()` is called).

Typedefs

- `typedef void(* periodic_task)(void *arg)`
Prototype of the function which can be scheduled.
- `typedef struct _Task_Thread Task_Thread`
Represents a separate thread which can be used to schedule tasks.

Functions

- `int ptask_cancel (Task_Thread *thread, int taskId, BOOL wait)`
Removes task from the scheduled list.
- `int ptask_dispose (Task_Thread *thread, BOOL wait)`
Releases resources allocated for the provided `Task_Thread` instance.
- `Task_Thread * ptask_init ()`
Creates new instance of `Task_Thread`.

- **BOOL ptask_isScheduled (Task_Thread *thread, int taskId)**
Checks whether the task with provided id is scheduled for execution in the thread.
- **int ptask_reschedule (Task_Thread *thread, int taskId, long period, int executeTimes, BOOL add)**
Sets new execution period for the specified task.
- **int ptask_reset (Task_Thread *thread, int taskId)**
Resets time until the next execution of the specified task.
- **int ptask_schedule (Task_Thread *thread, periodic_task task, void *arg, long period, int executeTimes)**
Schedules new task for execution.

7.22.1 Detailed Description

Periodic Task - tool for asynchronous task execution.

Periodic Task utility can be used to schedule some function(s) to be invoked periodically in a separate thread. A function can be scheduled to be invoked either defined number of times or indefinite (until it is canceled).

7.22.2 Usage

The following piece of code will schedule a function *foo()* to be executed every second in a separate thread:

```
// prints greetings
int foo(void* args)
{
    printf(Greetings from %s!, (char*) args);
}

...

// initialize a new thread
Task_Thread* thread = ptask_init();
if (thread == NULL)
{
    // initialization failed
    return -1;
}

long period = 1000;      // interval between executions in milliseconds
int timesToExecute = EXECUTE_INDEFINITE; // how many times to execute
char* arg = (char*) malloc(7); // It will be passed to the foo() as argument
strcpy(arg, "Andrey");

// schedule foo() method to be executed indefinitely once is a second
int taskId = ptask_schedule(thread, &foo, arg, period, timesToExecute);
```

Note:

foo() function should match **periodic_task** prototype.

The variable which is passed to the **ptask_schedule()** function as argument for *foo()* shouldn't be locally defined. Otherwise it can appear that it doesn't exist when the *foo()* is executed.

The returned task id can be then used to change execution period, or cancel the task.

One task thread can be used to schedule several tasks, but scheduled functions must be quick enough in order not to block other tasks to be executed in time.

At the end of application all initialized task thread should be freed using [ptask_dispose\(\)](#).

Author:

Andrey Litvinov

Version:

1.1

Definition in file [ptask.h](#).

7.22.3 Define Documentation

7.22.3.1 #define EXECUTE_INDEFINITE -1

Specifies that the task should be executed indefinite number of times (until [ptask_cancel\(\)](#) is called).

Definition at line 88 of file ptask.h.

Referenced by addListener(), main(), test_ptask_byHands(), testPeriodicTask(), and testPtTaskReschedule().

7.22.4 Typedef Documentation

7.22.4.1 `typedef void(* periodic_task)(void *arg)`

Prototype of the function which can be scheduled.

Parameters:

arg Argument which is passed to the function when it is invoked.

Definition at line 95 of file ptask.h.

7.22.4.2 `typedef struct _Task_Thread Task_Thread`

Represents a separate thread which can be used to schedule tasks.

Definition at line 100 of file ptask.h.

7.22.5 Function Documentation

7.22.5.1 `int ptask_cancel (Task_Thread *thread, int taskId, BOOL wait)`

Removes task from the scheduled list.

Parameters:

thread Thread in which task is scheduled.

taskId ID of the task to be removed.

wait When task is being executed it can be canceled only after execution is completed. This parameter defines whether the function should wait until the task is really canceled, or it can just mark the task as canceled, which guarantees that the task will be removed as soon as the current execution is completed. In case when this function is called while the task is not executed *wait* argument makes no difference.

Returns:

- *0* on success;
- *-1* if task with provided ID is not found.

Definition at line 396 of file ptask.c.

References *_Periodic_Task::isCancelled*, *_Periodic_Task::isExecuting*, *periodicTask_free()*, *periodicTask_get()*, *periodicTask_removeFromList()*, *_Task_Thread::taskExecuted*, *_Task_Thread::taskListMutex*, *_Task_Thread::taskListUpdated*, and *TRUE*.

Referenced by *main()*, *obixWatch_delete()*, *removeWatch()*, *testPeriodicTask()*, *testPtaskReschedule()*, *threadCycle()*, and *watchPollTask()*.

7.22.5.2 int ptask_dispose (Task_Thread * *thread*, BOOL *wait*)

Releases resources allocated for the provided [Task_Thread](#) instance.

All scheduled tasks are canceled.

Parameters:

thread Pointer to the [Task_Thread](#) to be freed.

wait If [TRUE](#) than the method will block and wait until specified thread is really disposed. Otherwise, method will only schedule asynchronous disposing of the thread.

Returns:

0 on success, negative error code otherwise.

Definition at line 599 of file ptask.c.

References *ptask_schedule()*, *stopTask()*, and *_Task_Thread::thread*.

Referenced by *http_dispose()*, *main()*, *obixWatch_dispose()*, and *taskStopThread()*.

7.22.5.3 Task_Thread* ptask_init ()

Creates new instance of [Task_Thread](#).

Returns:

Pointer to the new instance of [Task_Thread](#), or *NULL* if some error occurred.

Definition at line 518 of file ptask.c.

References `_Task_Thread::id_gen`, `log_error`, `_Task_Thread::taskExecuted`, `_Task_Thread::taskList`, `_Task_Thread::taskListMutex`, `_Task_Thread::taskListUpdated`, `_Task_Thread::thread`, and `threadCycle()`.

Referenced by `http_init()`, `main()`, `obixWatch_init()`, and `test_ptask_byHands()`.

7.22.5.4 BOOL ptask_isScheduled (Task_Thread * *thread*, int *taskId*)

Checks whether the task with provided id is scheduled for execution in the thread.

Parameters:

thread Thread where the task should be searched for.

taskId Task id which is searched for.

Returns:

`TRUE` if the task with specified *taskId* is scheduled, `FALSE` otherwise.

Definition at line 364 of file ptask.c.

References `FALSE`, `periodicTask_get()`, and `TRUE`.

7.22.5.5 int ptask_reschedule (Task_Thread * *thread*, int *taskId*, long *period*, int *executeTimes*, BOOL *add*)

Sets new execution period for the specified task.

Parameters:

thread Thread in which the task is scheduled.

taskId Id of the scheduled task.

period New time interval in milliseconds (or time which will be added to the current task period).

executeTimes Defines how many times (min. 1) the task should be executed. If `EXECUTE_INDEFINITE` is provided than the task is executed until `ptask_cancel()` is called with corresponding task ID.

add Defines whether time provided in *period* argument will be used as new execution period, or will be added to the current one.

Note:

When *add* is set to `TRUE`, *period* will be also added to the next execution time, but when *add* is `FALSE` the next execution will be (current time + *period*).

Returns:

0 on success, negative error code otherwise.

Definition at line 309 of file ptask.c.

References `_Periodic_Task::executeTimes`, `_Periodic_Task::nextScheduledTime`, `_Periodic_Task::period`, `periodicTask_get()`, `periodicTask_resetExecTime()`, `periodicTask_setPeriod()`, `_Task_Thread::taskListMutex`, `_Task_Thread::taskListUpdated`, `timespec_add()`, `timespec_copy()`, and `timespec_fromMillis()`.

Referenced by `obixWatch_deleteHelper()`, `obixWatch_notifyPollTask()`, `obixWatch_resetLeaseTimer()`, `removeWatch()`, and `testPtaskReschedule()`.

7.22.5.6 int ptask_reset (Task_Thread * *thread*, int *taskId*)

Resets time until the next execution of the specified task.

The next execution time will be current time + *period* provided when the task was scheduled. If the *period* needs to be changed than use [ptask_reschedule\(\)](#) instead.

Parameters:

thread Thread where the task is scheduled.

taskId Id of the task whose execution time should be reset.

Returns:

0 on success, negative error code otherwise.

Definition at line 376 of file ptask.c.

References `periodicTask_get()`, `periodicTask_resetExecTime()`, `_Task_Thread::taskListMutex`, and `_Task_Thread::taskListUpdated`.

Referenced by `checkTargets()`, `obixWatch_resetLeaseTimer()`, `resetListener()`, and `targetRemoveTask()`.

7.22.5.7 int ptask_schedule (Task_Thread * *thread*, periodic_task *task*, void * *arg*, long *period*, int *executeTimes*)

Schedules new task for execution.

Parameters:

thread Thread in which the task will be executed.

task Task which should be scheduled.

arg Argument which will be passed to the task function each time when it is invoked.

period Time interval in milliseconds, which defines how often the task will be executed.

executeTimes Defines how many times (min. 1) the task should be executed. If [EXECUTE_INDEFINITE](#) is provided than the task is executed until [ptask_cancel\(\)](#) with corresponding task ID is called.

Returns:

- >0 - ID of the scheduled task.
- <0 - Error code.

Definition at line 250 of file ptask.c.

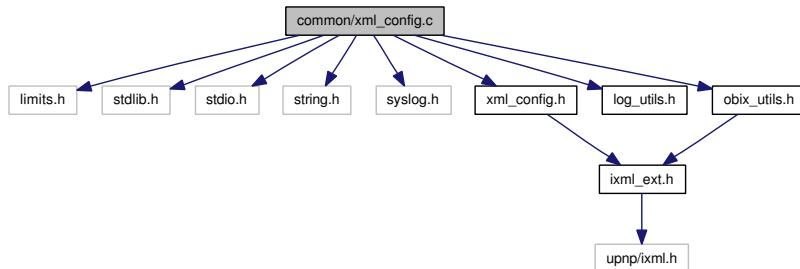
References `_Periodic_Task::id`, `_Periodic_Task::next`, `_Periodic_Task::period`, `periodicTask_create()`, `_Periodic_Task::prev`, `_Task_Thread::taskList`, `_Task_Thread::taskListMutex`, `_Task_Thread::taskListUpdated`, and `timespec_cmp()`.

Referenced by `addListener()`, `main()`, `obixWatch_create()`, `obixWatch_holdPoll()`, `ptask_dispose()`, `target_get()`, `test_ptask_byHands()`, `testPeriodicTask()`, and `testPtaskReschedule()`.

7.23 common/xml_config.c File Reference

```
#include <limits.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <syslog.h>
#include "xml_config.h"
#include "log_utils.h"
#include "obix_utils.h"
```

Include dependency graph for `xml_config.c`:



Functions

- `void config_finishInit (IXML_Element *conf, BOOL successful)`
Releases resources allocated for settings parsing.
- `IXML_Element * config_getChildTag (IXML_Element *conf, const char *tagName, BOOL obligatory)`
Returns child tag of the provided element with the specified name.
- `const char * config_getChildTagValue (IXML_Element *conf, const char *tagName, BOOL obligatory)`
Returns value of the child tag with specified name.
- `char * config_getResFullPath (const char *filename)`
Returns the address of the resource file by adding resource folder path to the filename.
- `int config_getTagAttrBoolValue (IXML_Element *tag, const char *attrName, BOOL obligatory)`

Returns boolean value of the specified tag attribute.

- const char * [config_getTagAttributeValue](#) (IXML_Element *tag, const char *attrName, **BOOL** obligatory)

Returns value of the specified tag attribute.

- int [config_getTagAttrIntValue](#) (IXML_Element *tag, const char *attrName, **BOOL** obligatory, int defaultValue)

Returns integer value of the specified tag attribute.

- long [config_getTagAttrLongValue](#) (IXML_Element *tag, const char *attrName, **BOOL** obligatory, long defaultValue)

Returns long integer value of the specified tag attribute.

- IXML_Element * [config_loadFile](#) (const char *filename)

Opens the configuration file and performs it's initial parsing.

- int [config_log](#) (IXML_Element *configTag)

Configures log system using parameters passed in XML format.

- void [config_setResourceDir](#) (char *path)

Sets the address of the resource folder where configuration file is stored.

Variables

- const char * **CT_CONFIG** = "config"

Main configuration tag's name.

- const char * **CT_LOG** = "log"

Parent log configuration tag, containing all settings.

- const char * **CT_LOG_LEVEL** = "level"

Log level tag.

- const char * **CT_LOG_USE_SYSLOG** = "use-syslog"

Defines whether syslog should be used for logging.

- const char * **CTA_LOG_FACILITY** = "facility"

Attribute which defines syslog facility.

- const char * **CTA_VALUE** = "val"

Most commonly used tag attribute: 'val'.

- const char * **CTAV_LOG_FACILITY_DAEMON** = "daemon"

Log facility 'daemon'.

- const char * **CTAV_LOG_FACILITY_LOCAL0** = "local0"

Log facility 'local0'.

- const char * **CTAV_LOGFacilityUser** = "user"
Log facility 'user'.
- const char * **CTAVLogLevelDebug** = "debug"
Log level debug.
- const char * **CTAVLogLevelError** = "error"
Log level error.
- const char * **CTAVLogLevelNo** = "no"
Log level no.
- const char * **CTAVLogLevelWarning** = "warning"
Log level warning.
- static char * **resourceFolder**

7.23.1 Detailed Description

Todo

add description

Author:

Andrey Litvinov

Version:

1.1

Definition in file [xml_config.c](#).

7.23.2 Function Documentation

7.23.2.1 void config_finishInit (IXML_Element * *conf*, BOOL *successful*)

Releases resources allocated for settings parsing.

Should be called once after all settings are loaded (or failed to load). Also writes message to log, telling that initialization is completed. Depending on *successful* parameter the log message tells that initialization completed or failed.

Parameters:

conf Settings which should be freed. It can be the link returned by [config_loadFile\(\)](#) or a link to any child tag.

successful Tells whether application initialized successfully or not.

Definition at line 275 of file [xml_config.c](#).

References [ixmlElement_freeOwnerDocument\(\)](#), [log_debug](#), and [log_error](#).

Referenced by [loadSettings\(\)](#), [obix_fcgi_init\(\)](#), [obix_loadConfigFile\(\)](#), and [testLog\(\)](#).

7.23.2.2 IXML_Element* config_getChildTag (IXML_Element * *conf*, const char * *tagName*, BOOL *obligatory*)

Returns child tag of the provided element with the specified name.

Parameters:

conf Configuration tag in which search should be performed.

tagName Name of the tag to be searched for.

obligatory If TRUE than an error message will be logged when the tag is not found.

Returns:

Found child tag, or NULL if nothing is found.

Definition at line 54 of file xml_config.c.

References ixmlNode_convertToElement(), log_error, and TRUE.

Referenced by config_getChildTagValue(), config_loadFile(), config_log(), connection_create(), http_initConnection(), loadDeviceData(), loadSettings(), obix_fcgi_loadConfig(), and obix_server_init().

7.23.2.3 const char* config_getChildTagValue (IXML_Element * *conf*, const char * *tagName*, BOOL *obligatory*)

Returns value of the child tag with specified name.

Tag value is the value of it's val attribute.

Parameters:

conf Configuration tag in which search should be performed.

tagName Name of the tag to be searched for.

obligatory If TRUE than an error message will be logged when the tag is not found, or it doesn't have any value.

Returns:

Value of the found child tag or NULL if the tag is not found or the found one doesn't have any value.

Definition at line 87 of file xml_config.c.

References config_getChildTag(), config_getTagAttributeValue(), and CTA_VALUE.

7.23.2.4 char* config_getResFullPath (const char * *filename*)

Returns the address of the resource file by adding resource folder path to the filename.

Note:

Do not forget to release memory allocated for returned string.

Parameters:

filename Name of the file.

Returns:

Full path to the resource file, or *NULL* on error.

Definition at line 288 of file xml_config.c.

References log_error, and resourceFolder.

Referenced by config_loadFile(), and xmldb_loadFile().

7.23.2.5 int config_getTagAttrBoolValue (IXML_Element *tag, const char *attrName, BOOL obligatory)

Returns boolean value of the specified tag attribute.

Parameters:

tag Configuration tag whose attribute should be parsed.

attrName Name of the attribute to read.

obligatory if TRUE than an error message will be written when the attribute is not found.

Returns:

- 0 - Attribute's value is "false";
- 1 - Attribute's value is "true";
- -1 - Parsing error.

Definition at line 188 of file xml_config.c.

References config_getTagAttributeValue(), FALSE, log_debug, log_error, log_warning, TRUE, XML_FALSE, and XML_TRUE.

7.23.2.6 const char* config_getTagAttributeValue (IXML_Element *tag, const char *attrName, BOOL obligatory)

Returns value of the specified tag attribute.

Parameters:

tag Configuration tag whose attribute should be returned.

attrName Name of the attribute to read.

obligatory if TRUE than an error message will be written when the attribute is not found.

Returns:

Attribute's value or *NULL* if the attribute is not found.

Definition at line 100 of file xml_config.c.

References log_error, and TRUE.

Referenced by config_getChildTagValue(), config_getTagAttrBoolValue(), config_getTagAttrLongValue(), config_log(), connection_create(), http_initConnection(), and obix_server_init().

7.23.2.7 int config_getTagAttrIntValue (IXML_Element *tag, const char *attrName, BOOL obligatory, int defaultValue)

Returns integer value of the specified tag attribute.

Can be used only for parsing positive values.

Parameters:

tag Configuration tag whose attribute should be parsed.

attrName Name of the attribute to read.

obligatory if TRUE than an error message will be written when the attribute is not found.

defaultValue if *obligatory* is FALSE than this value will be returned instead of the error code.

Returns:

- $>=0$ - Parsed integer attribute value;
- -1 - Error code. Error description will be logged.

Definition at line 111 of file xml_config.c.

References config_getTagAttrLongValue(), and log_error.

Referenced by connection_create(), loadDeviceData(), and obix_fcgi_loadConfig().

7.23.2.8 long config_getTagAttrLongValue (IXML_Element *tag, const char *attrName, BOOL obligatory, long defaultValue)

Returns long integer value of the specified tag attribute.

Can be used only for parsing positive values.

Parameters:

tag Configuration tag whose attribute should be parsed.

attrName Name of the attribute to read.

obligatory if TRUE than an error message will be written when the attribute is not found.

defaultValue if *obligatory* is FALSE than this value will be returned instead of the error code.

Returns:

- $>=0$ - Parsed long integer attribute value;
- -1 - Error code. Error description will be logged.

Definition at line 142 of file xml_config.c.

References config_getTagAttributeValue(), log_debug, and log_error.

Referenced by config_getTagAttrIntValue(), and http_initConnection().

7.23.2.9 IXML_Element* config_loadFile (const char *filename)

Opens the configuration file and performs it's initial parsing.

Note:

Returned XML DOM structure should not be freed manually. Instead, use [config_finishInit\(\)](#) after parsing all required settings.

Parameters:

filename Name of the configuration file.

Returns:

Reference to the <config> tag, which can be used to load application settings.

Definition at line 234 of file xml_config.c.

References config_getChildTag(), config_getResFullPath(), CT_CONFIG, ixmDocument_-getRootElement(), log_error, and TRUE.

Referenced by loadSettings(), obix_fcgi_loadConfig(), obix_loadConfigFile(), and testLog().

7.23.2.10 int config_log (IXML_Element *configTag)

Configures log system using parameters passed in XML format.

Passed XML should contain at least <log> tag with child <level>, which specifies log level. For example:

```
<log>
  <level val="debug" />
</log>
```

There are other optional configuration tags. Please refer to the <log> element in example_timer_-config.xml for the full description of possible tags.

Parameters:

configTag XML configuration document, which contains log configuration tags. Document can contain other elements which are ignored.

Returns:

0 on success; -1 on error.

Definition at line 342 of file xml_config.c.

References config_getChildTag(), config_getTagAttributeValue(), CT_LOG, CT_LOG_LEVEL, CT_LOG_USE_SYSLOG, CTA_LOGFacility, CTA_Value, CTAV_LOGFacility_DAEMON, CTAV_LOGFacility_LOCAL0, CTAV_LOGFacility_USER, CTAV_LOGLevel_DEBUG, CTAV_LOGLevel_ERROR, CTAV_LOGLevel_NO, CTAV_LOGLevel_WARNING, FALSE, log_debug, log_error, LOG_LEVEL_DEBUG, LOG_LEVEL_ERROR, LOG_LEVEL_NO, LOG_LEVEL_WARNING, log_setLevel(), log_usePrintf(), log_useSyslog(), and TRUE.

Referenced by loadSettings(), obix_fcgi_loadConfig(), obix_loadConfigFile(), and testLog().

7.23.2.11 void config_setResourceDir (char * *path*)

Sets the address of the resource folder where configuration file is stored.

The general idea is to keep all resource files in one place which is defined only once in the application. After that all resources (including configuration file) can be reached using [config_getResFullPath\(\)](#).

Parameters:

path Path to the application's resource folder.

Definition at line 309 of file xml_config.c.

References log_debug, log_error, and resourceFolder.

Referenced by obix_fcgi_loadConfig(), and test_server().

7.23.3 Variable Documentation

7.23.3.1 const char* CT_CONFIG = "config"

Main configuration tag's name.

Definition at line 37 of file xml_config.c.

Referenced by config_loadFile().

7.23.3.2 const char* CT_LOG = "log"

Parent log configuration tag, containing all settings.

Definition at line 40 of file xml_config.c.

Referenced by config_log().

7.23.3.3 const char* CT_LOG_LEVEL = "level"

Log level tag.

Defines which messages are logged.

Definition at line 41 of file xml_config.c.

Referenced by config_log().

7.23.3.4 const char* CT_LOG_USE_SYSLOG = "use-syslog"

Defines whether *syslog* should be used for logging.

If tag with such name is not present, than all messages are printed to *stdout*.

Definition at line 46 of file xml_config.c.

Referenced by config_log().

7.23.3.5 const char* CTA_LOG_FACILITY = "facility"

Attribute which defines syslog facility.

Definition at line 47 of file xml_config.c.

Referenced by config_log().

7.23.3.6 const char* CTA_VALUE = "val"

Most commonly used tag attribute: 'val'.

Definition at line 38 of file xml_config.c.

Referenced by config_getChildTagValue(), config_log(), connection_create(), http_initConnection(), obix_fcgi_loadConfig(), and obix_server_init().

7.23.3.7 const char* CTAV_LOG_FACILITY_DAEMON = "daemon"

Log facility 'daemon'.

Definition at line 49 of file xml_config.c.

Referenced by config_log().

7.23.3.8 const char* CTAV_LOG_FACILITY_LOCAL0 = "local0"

Log facility 'local0'.

Definition at line 50 of file xml_config.c.

Referenced by config_log().

7.23.3.9 const char* CTAV_LOG_FACILITY_USER = "user"

Log facility 'user'.

Definition at line 48 of file xml_config.c.

Referenced by config_log().

7.23.3.10 const char* CTAV_LOG_LEVEL_DEBUG = "debug"

Log level *debug*.

All messages are logged.

Definition at line 42 of file xml_config.c.

Referenced by config_log().

7.23.3.11 const char* CTAV_LOG_LEVEL_ERROR = "error"

Log level *error*.

Only *error* messages are logged.

Definition at line 44 of file xml_config.c.

Referenced by config_log().

7.23.3.12 const char* CTAV_LOG_LEVEL_NO = "no"

Log level *no*.

Nothing is logged at all.

Definition at line 45 of file xml_config.c.

Referenced by config_log().

7.23.3.13 const char* CTAV_LOG_LEVEL_WARNING = "warning"

Log level *warning*.

Only *warning* and *error* messages are logged.

Definition at line 43 of file xml_config.c.

Referenced by config_log().

7.23.3.14 char* resourceFolder [static]

Definition at line 52 of file xml_config.c.

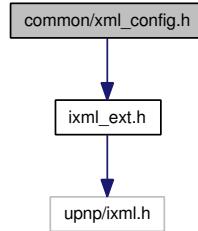
Referenced by config_getResFullPath(), and config_setResourceDir().

7.24 common/xml_config.h File Reference

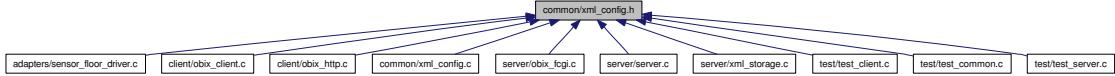
Declares configuration API.

```
#include <ixml_ext.h>
```

Include dependency graph for xml_config.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [config_finishInit](#) (IXML_Element *conf, **BOOL** successful)
Releases resources allocated for settings parsing.
- IXML_Element * [config_getChildTag](#) (IXML_Element *conf, const char *tagName, **BOOL** obligatory)
Returns child tag of the provided element with the specified name.
- const char * [config_getChildTagValue](#) (IXML_Element *conf, const char *tagName, **BOOL** obligatory)
Returns value of the child tag with specified name.
- char * [config_getResFullPath](#) (const char *filename)
Returns the address of the resource file by adding resource folder path to the filename.
- int [config_getTagAttrBoolValue](#) (IXML_Element *tag, const char *attrName, **BOOL** obligatory)
Returns boolean value of the specified tag attribute.
- const char * [config_getTagAttributeValue](#) (IXML_Element *tag, const char *attrName, **BOOL** obligatory)

Returns value of the specified tag attribute.

- int `config_getTagAttrIntValue` (IXML_Element *tag, const char *attrName, **BOOL** obligatory, int defaultValue)

Returns integer value of the specified tag attribute.

- long `config_getTagAttrLongValue` (IXML_Element *tag, const char *attrName, **BOOL** obligatory, long defaultValue)

Returns long integer value of the specified tag attribute.

- IXML_Element * `config_loadFile` (const char *filename)

Opens the configuration file and performs it's initial parsing.

- int `config_log` (IXML_Element *configTag)

Configures log system using parameters passed in XML format.

- void `config_setResourceDir` (char *path)

Sets the address of the resource folder where configuration file is stored.

Variables

Common configuration file keywords

The following acronyms are used in constants' names:

- CT means Configuration Tag;
- CTA - Configuration Tag's Attribute;
- CTAV - Configuration Tag's Attribute's Value.
- const char * `CT_CONFIG`
Main configuration tag's name.
- const char * `CTA_VALUE`
Most commonly used tag attribute: 'val'.

Log configuration keywords

These variables contain names of all log configuration tags, their attributes and possible attributes' values.

- const char * `CT_LOG`
Parent log configuration tag, containing all settings.
- const char * `CT_LOG_LEVEL`
Log level tag.
- const char * `CT_LOG_USE_SYSLOG`
Defines whether syslog should be used for logging.
- const char * `CTA_LOG_FACILITY`
Attribute which defines syslog facility.

- const char * **CTAV_LOGFacility_daemon**
Log facility 'daemon'.
- const char * **CTAV_LOGFacility_local0**
Log facility 'local0'.
- const char * **CTAV_LOGFacility_user**
Log facility 'user'.
- const char * **CTAV_LogLevel_debug**
Log level debug.
- const char * **CTAV_LogLevel_error**
Log level error.
- const char * **CTAV_LogLevel_no**
Log level no.
- const char * **CTAV_LogLevel_warning**
Log level warning.

7.24.1 Detailed Description

Declares configuration API.

Configuration API allows loading settings from XML file which contains <config> element. This header defines required functions which simplify parsing of XML settings.

The API is tightly integrated with the logging system ([log_utils.h](#)). All functions extensively use [log_utils.h](#) utilities for logging warning and error messages occurred during parsing. There is also a possibility to load log system configuration from an XML file.

Author:

Andrey Litvinov

Version:

1.1

Definition in file [xml_config.h](#).

7.24.2 Function Documentation

7.24.2.1 void config_finishInit (IXML_Element * *conf*, BOOL *successful*)

Releases resources allocated for settings parsing.

Should be called once after all settings are loaded (or failed to load). Also writes message to log, telling that initialization is completed. Depending on *successful* parameter the log message tells that initialization completed or failed.

Parameters:

conf Settings which should be freed. It can be the link returned by [config_loadFile\(\)](#) or a link to any child tag.

successful Tells whether application initialized successfully or not.

Definition at line 275 of file xml_config.c.

References `ixmlElement_freeOwnerDocument()`, `log_debug`, and `log_error`.

Referenced by `loadSettings()`, `obix_fcgi_init()`, `obix_loadConfigFile()`, and `testLog()`.

7.24.2.2 IXML_Element* config_getChildTag (IXML_Element * *conf*, const char * *tagName*, BOOL *obligatory*)

Returns child tag of the provided element with the specified name.

Parameters:

conf Configuration tag in which search should be performed.

tagName Name of the tag to be searched for.

obligatory If `TRUE` than an error message will be logged when the tag is not found.

Returns:

Found child tag, or `NULL` if nothing is found.

Definition at line 54 of file xml_config.c.

References `ixmlNode_convertToElement()`, `log_error`, and `TRUE`.

Referenced by `config_getChildTagValue()`, `config_loadFile()`, `config_log()`, `connection_create()`, `http_initConnection()`, `loadDeviceData()`, `loadSettings()`, `obix_fcgi_loadConfig()`, and `obix_server_init()`.

7.24.2.3 const char* config_getChildTagValue (IXML_Element * *conf*, const char * *tagName*, BOOL *obligatory*)

Returns value of the child tag with specified name.

Tag value is the value of it's `val` attribute.

Parameters:

conf Configuration tag in which search should be performed.

tagName Name of the tag to be searched for.

obligatory If `TRUE` than an error message will be logged when the tag is not found, or it doesn't have any value.

Returns:

Value of the found child tag or `NULL` if the tag is not found or the found one doesn't have any value.

Definition at line 87 of file xml_config.c.

References `config_getChildTag()`, `config_getTagAttributeValue()`, and `CTA_VALUE`.

7.24.2.4 char* config_getResFullPath (const char *filename)

Returns the address of the resource file by adding resource folder path to the filename.

Note:

Do not forget to release memory allocated for returned string.

Parameters:

filename Name of the file.

Returns:

Full path to the resource file, or *NULL* on error.

Definition at line 288 of file xml_config.c.

References log_error, and resourceFolder.

Referenced by config_loadFile(), and xmldb_loadFile().

7.24.2.5 int config_getTagAttrBoolValue (IXML_Element *tag, const char *attrName, BOOL obligatory)

Returns boolean value of the specified tag attribute.

Parameters:

tag Configuration tag whose attribute should be parsed.

attrName Name of the attribute to read.

obligatory if TRUE than an error message will be written when the attribute is not found.

Returns:

- 0 - Attribute's value is "false";
- 1 - Attribute's value is "true";
- -1 - Parsing error.

Definition at line 188 of file xml_config.c.

References config_getTagAttributeValue(), FALSE, log_debug, log_error, log_warning, TRUE, XML_FALSE, and XML_TRUE.

7.24.2.6 const char* config_getTagAttributeValue (IXML_Element *tag, const char *attrName, BOOL obligatory)

Returns value of the specified tag attribute.

Parameters:

tag Configuration tag whose attribute should be returned.

attrName Name of the attribute to read.

obligatory if TRUE than an error message will be written when the attribute is not found.

Returns:

Attribute's value or NULL if the attribute is not found.

Definition at line 100 of file xml_config.c.

References log_error, and TRUE.

Referenced by config_getChildTagValue(), config_getTagAttrBoolValue(), config_getTagAttrLongValue(), config_log(), connection_create(), http_initConnection(), and obix_server_init().

7.24.2.7 int config_getTagAttrIntValue (IXML_Element *tag, const char *attrName, BOOL obligatory, int defaultValue)

Returns integer value of the specified tag attribute.

Can be used only for parsing positive values.

Parameters:

tag Configuration tag whose attribute should be parsed.

attrName Name of the attribute to read.

obligatory if TRUE than an error message will be written when the attribute is not found.

defaultValue if *obligatory* is FALSE than this value will be returned instead of the error code.

Returns:

- ≥ 0 - Parsed integer attribute value;
- -1 - Error code. Error description will be logged.

Definition at line 111 of file xml_config.c.

References config_getTagAttrLongValue(), and log_error.

Referenced by connection_create(), loadDeviceData(), and obix_fcgi_loadConfig().

7.24.2.8 long config_getTagAttrLongValue (IXML_Element *tag, const char *attrName, BOOL obligatory, long defaultValue)

Returns long integer value of the specified tag attribute.

Can be used only for parsing positive values.

Parameters:

tag Configuration tag whose attribute should be parsed.

attrName Name of the attribute to read.

obligatory if TRUE than an error message will be written when the attribute is not found.

defaultValue if *obligatory* is FALSE than this value will be returned instead of the error code.

Returns:

- $>=0$ - Parsed long integer attribute value;
- -1 - Error code. Error description will be logged.

Definition at line 142 of file xml_config.c.

References config_getTagAttributeValue(), log_debug, and log_error.

Referenced by config_getTagAttrIntValue(), and http_initConnection().

7.24.2.9 IXML_Element* config_loadFile (const char *filename)

Opens the configuration file and performs it's initial parsing.

Note:

Returned XML DOM structure should not be freed manually. Instead, use config_finishInit() after parsing all required settings.

Parameters:

filename Name of the configuration file.

Returns:

Reference to the <config> tag, which can be used to load application settings.

Definition at line 234 of file xml_config.c.

References config_getChildTag(), config_getResFullPath(), CT_CONFIG, ixmldocument_-getRootElement(), log_error, and TRUE.

Referenced by loadSettings(), obix_fcgi_loadConfig(), obix_loadConfigFile(), and testLog().

7.24.2.10 int config_log (IXML_Element * configTag)

Configures log system using parameters passed in XML format.

Passed XML should contain at least <log> tag with child <level>, which specifies log level. For example:

```
<log>
  <level val="debug" />
</log>
```

There are other optional configuration tags. Please refer to the <log> element in example_timer_-config.xml for the full description of possible tags.

Parameters:

configTag XML configuration document, which contains log configuration tags. Document can contain other elements which are ignored.

Returns:

0 on success; -1 on error.

Definition at line 342 of file xml_config.c.

References config_getChildTag(), config_getTagAttributeValue(), CT_LOG, CT_LOG_LEVEL, CT_LOG_USE_SYSLOG, CTA_LOG_FACILITY, CTA_VALUE, CTAV_LOG_FACILITY_DAEMON, CTAV_LOG_FACILITY_LOCAL0, CTAV_LOG_FACILITY_USER, CTAV_LOG_LEVEL_DEBUG, CTAV_LOG_LEVEL_ERROR, CTAV_LOG_LEVEL_NO, CTAV_LOG_LEVEL_WARNING, FALSE, log_debug, log_error, LOG_LEVEL_DEBUG, LOG_LEVEL_ERROR, LOG_LEVEL_NO, LOG_LEVEL_WARNING, log_setLevel(), log_usePrintf(), log_useSyslog(), and TRUE.

Referenced by loadSettings(), obix_fcgi_loadConfig(), obix_loadConfigFile(), and testLog().

7.24.2.11 void config_setResourceDir (char * *path*)

Sets the address of the resource folder where configuration file is stored.

The general idea is to keep all resource files in one place which is defined only once in the application. After that all resources (including configuration file) can be reached using [config_getResFullPath\(\)](#).

Parameters:

path Path to the application's resource folder.

Definition at line 309 of file xml_config.c.

References log_debug, log_error, and resourceFolder.

Referenced by obix_fcgi_loadConfig(), and test_server().

7.24.3 Variable Documentation**7.24.3.1 const char* CT_CONFIG**

Main configuration tag's name.

Definition at line 37 of file xml_config.c.

Referenced by config_loadFile().

7.24.3.2 const char* CT_LOG

Parent log configuration tag, containing all settings.

Definition at line 40 of file xml_config.c.

Referenced by config_log().

7.24.3.3 const char* CT_LOG_LEVEL

Log level tag.

Defines which messages are logged.

Definition at line 41 of file xml_config.c.

Referenced by config_log().

7.24.3.4 const char* CT_LOG_USE_SYSLOG

Defines whether *syslog* should be used for logging.

If tag with such name is not present, than all messages are printed to *stdout*.

Definition at line 46 of file xml_config.c.

Referenced by config_log().

7.24.3.5 const char* CTA_LOG_FACILITY

Attribute which defines syslog facility.

Definition at line 47 of file xml_config.c.

Referenced by config_log().

7.24.3.6 const char* CTA_VALUE

Most commonly used tag attribute: 'val'.

Definition at line 38 of file xml_config.c.

Referenced by config_getChildTagValue(), config_log(), connection_create(), http_initConnection(), obix_fcgi_loadConfig(), and obix_server_init().

7.24.3.7 const char* CTAV_LOG_FACILITY_DAEMON

Log facility 'daemon'.

Definition at line 49 of file xml_config.c.

Referenced by config_log().

7.24.3.8 const char* CTAV_LOG_FACILITY_LOCAL0

Log facility 'local0'.

Definition at line 50 of file xml_config.c.

Referenced by config_log().

7.24.3.9 const char* CTAV_LOG_FACILITY_USER

Log facility 'user'.

Definition at line 48 of file xml_config.c.

Referenced by config_log().

7.24.3.10 const char* CTAV_LOG_LEVEL_DEBUG

Log level *debug*.

All messages are logged.

Definition at line 42 of file xml_config.c.

Referenced by config_log().

7.24.3.11 const char* CTAV_LOG_LEVEL_ERROR

Log level *error*.

Only *error* messages are logged.

Definition at line 44 of file xml_config.c.

Referenced by config_log().

7.24.3.12 const char* CTAV_LOG_LEVEL_NO

Log level *no*.

Nothing is logged at all.

Definition at line 45 of file xml_config.c.

Referenced by config_log().

7.24.3.13 const char* CTAV_LOG_LEVEL_WARNING

Log level *warning*.

Only *warning* and *error* messages are logged.

Definition at line 43 of file xml_config.c.

Referenced by config_log().

7.25 doxygen/cot_main.h File Reference

Doxygen documentation source file.

7.25.1 Detailed Description

Doxygen documentation source file.

Contains source for [CoBIX Tools Full Reference Manual](#).

Definition in file [cot_main.h](#).

7.26 doxygen/libcot_desc.h File Reference

Doxygen documentation source file.

7.26.1 Detailed Description

Doxygen documentation source file.

Contains source for [oBIX Client Library Overview](#).

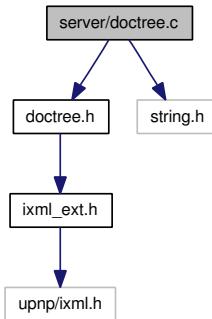
Definition in file [libcot_desc.h](#).

7.27 server/doctree.c File Reference

```
#include "doctree.h"
```

```
#include <string.h>
```

Include dependency graph for doctree.c:



Data Structures

- struct [_TreeNode](#)

Typedefs

- `typedef struct _TreeNode TreeNode`

Functions

- `static TreeNode * doctree_findNode (const char *start_ptr, int length, TreeNode *node)`
- `IXML_Element * doctree_get (const char *uri)`
- `int doctree_put (const char *uri, IXML_Element *data)`

Variables

- `static TreeNode * _tree = NULL`

7.27.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [doctree.c](#).

7.27.2 Typedef Documentation

7.27.2.1 `typedef struct _TreeNode TreeNode`

7.27.3 Function Documentation

7.27.3.1 `static TreeNode* doctree_findNode (const char * start_ptr, int length, TreeNode * node)` [static]

Definition at line 43 of file doctree.c.

References `_TreeNode::child`, `_TreeNode::name`, and `_TreeNode::neighbor`.

Referenced by `doctree_get()`.

7.27.3.2 `IXML_Element* doctree_get (const char * uri)`

Definition at line 82 of file doctree.c.

References `doctree_findNode()`.

7.27.3.3 int doctree_put (const char *uri, IXML_Element *data)

Definition at line 96 of file doctree.c.

7.27.4 Variable Documentation

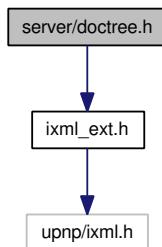
7.27.4.1 TreeNode* _tree = NULL [static]

Definition at line 41 of file doctree.c.

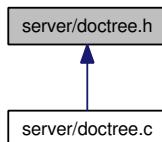
7.28 server/doctree.h File Reference

```
#include <ixml_ext.h>
```

Include dependency graph for doctree.h:



This graph shows which files directly or indirectly include this file:



Functions

- IXML_Element * [doctree_get](#) (const char *uri)
- int [doctree_put](#) (const char *uri, IXML_Element *data)

7.28.1 Detailed Description

[Todo](#)

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [doctree.h](#).**7.28.2 Function Documentation****7.28.2.1 IXML_Element* doctree_get (const char * uri)**

Definition at line 82 of file doctree.c.

References doctree_findNode().

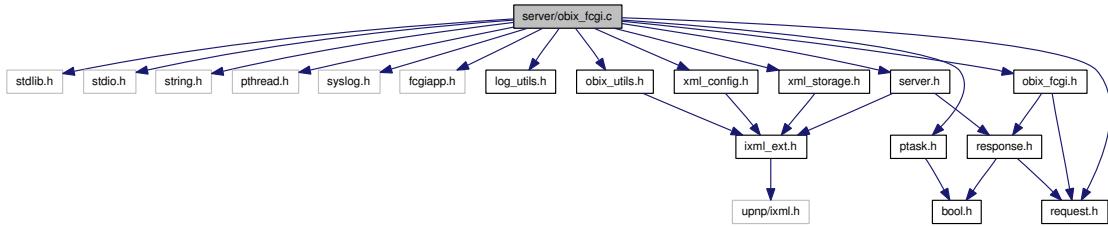
7.28.2.2 int doctree_put (const char * uri, IXML_Element * data)

Definition at line 96 of file doctree.c.

7.29 server/obix_fcgi.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <syslog.h>
#include <fcgiapp.h>
#include <log_utils.h>
#include <obix_utils.h>
#include <xml_config.h>
#include <ptask.h>
#include "xml_storage.h"
#include "server.h"
#include "request.h"
#include "obix_fcgi.h"
```

Include dependency graph for obix_fcgi.c:



Data Structures

- struct `_Request`

Defines

- #define `LISTENSOCK_FILENO` 0
- #define `LISTENSOCK_FLAGS` 0
- #define `MAX_PARALLEL_REQUEST_DEFAULT` 10

Functions

- int `main` (int argc, char **argv)
Entry point of FCGI script.
- void `obix_fcgi_dumpEnvironment` (Response *response)
- void `obix_fcgi_handleRequest` (Request *request)
- int `obix_fcgi_init` (char *resourceDir)
- IXML_Element * `obix_fcgi_loadConfig` (char *resourceDir)
- char * `obix_fcgi_readRequestInput` (Request *request)
- void `obix_fcgi_sendResponse` (Response *response)
- void `obix_fcgi_sendStaticErrorMessage` (Request *request)
- void `obix_fcgi_shutdown` (FCGX_Request *request)
- static int `obixRequest_create` ()
- static void `obixRequest_free` (Request *request)
- static Request * `obixRequest_get` ()
- static Request * `obixRequest_getHead` ()
- static void `obixRequest_release` (Request *request)
- static Request * `obixRequest_wait` ()
- static char * `parseArguments` (int argc, char **argv)

Variables

- static int `_requestIds` = 0
- static Request * `_requestList`
- pthread_mutex_t `_requestListMutex` = PTHREAD_MUTEX_INITIALIZER
- pthread_cond_t `_requestListNew` = PTHREAD_COND_INITIALIZER
- static int `_requestMaxCount` = MAX_PARALLEL_REQUEST_DEFAULT + 1

- static int _requestsInUse = 0
- static const char * **CONFIG_FILE** = "server_config.xml"
- static const char * **CT_HOLD_REQUEST_MAX** = "hold-request-max"
- static const char * **ERROR_STATIC**

Header of error answer.

- static const char * **HTTP_CONTENT_LOCATION** = "Content-Location: %s\r\n"
- static const char * **HTTP_STATUS_OK**

Standard header of any server answer.

- static const char * **XML_HEADER** = "\r\n<?xml version=\"1.0\" encoding=\"UTF-8\"?>\r\n"

7.29.1 Define Documentation

7.29.1.1 #define LISTENSOCK_FILENO 0

Definition at line 38 of file obix_fcgi.c.

Referenced by obixRequest_create().

7.29.1.2 #define LISTENSOCK_FLAGS 0

Definition at line 39 of file obix_fcgi.c.

Referenced by obixRequest_create().

7.29.1.3 #define MAX_PARALLEL_REQUEST_DEFAULT 10

Definition at line 41 of file obix_fcgi.c.

Referenced by obix_fcgi_loadConfig().

7.29.2 Function Documentation

7.29.2.1 int main (int *argc*, char ** *argv*)

Entry point of FCGI script.

Definition at line 143 of file obix_fcgi.c.

References **_Request::id**, **log_debug**, **log_warning**, **obix_fcgi_handleRequest()**, **obix_fcgi_init()**, **obix_fcgi_shutdown()**, **obixRequest_get()**, **parseArguments()**, and **_Request::r**.

7.29.2.2 void obix_fcgi_dumpEnvironment (Response * *response*)

Definition at line 462 of file obix_fcgi.c.

References Response::body, FALSE, log_debug, log_error, Response::next, obixResponse_add(), obixResponse_send(), obixResponse_setText(), _Request::r, Response::request, TRUE, and xmldb_getDump().

Referenced by obix_fcgi_handleRequest().

7.29.2.3 void obix_fcgi_handleRequest (Request * *request*)

Definition at line 262 of file obix_fcgi.c.

References _Request::canWait, log_error, log_warning, OBIX_CONTRACT_ERR_UNSUPPORTED, obix_fcgi_dumpEnvironment(), obix_fcgi_readRequestInput(), obix_fcgi_sendResponse(), obix_fcgi_sendStaticErrorMessage(), obix_server_generateObixErrorMessage(), obix_server_handleGET(), obix_server_handlePOST(), obix_server_handlePUT(), obixResponse_create(), _Request::r, xmldb_compareServerAddr(), and xmldb_getServerAddressLength().

Referenced by main().

7.29.2.4 int obix_fcgi_init (char * *resourceDir*)

Definition at line 223 of file obix_fcgi.c.

References config_finishInit(), FALSE, log_error, obix_fcgi_loadConfig(), obix_fcgi_sendResponse(), obix_server_init(), and obixResponse_setListener().

Referenced by main().

7.29.2.5 IXML_Element* obix_fcgi_loadConfig (char * *resourceDir*)

Definition at line 189 of file obix_fcgi.c.

References _requestMaxCount, CONFIG_FILE, config_getChildTag(), config_getTagAttrIntValue(), config_loadFile(), config_log(), config_setResourceDir(), CT_HOLD_REQUEST_MAX, CTA_VALUE, FALSE, ixmlelement_freeOwnerDocument(), and MAX_PARALLEL_REQUEST_DEFAULT.

Referenced by obix_fcgi_init().

7.29.2.6 char* obix_fcgi_readRequestInput (Request * *request*)

Definition at line 413 of file obix_fcgi.c.

References log_debug, log_error, and _Request::r.

Referenced by obix_fcgi_handleRequest().

7.29.2.7 void obix_fcgi_sendResponse (Response * *response*)

Definition at line 366 of file obix_fcgi.c.

References Response::body, HTTP_CONTENT_LOCATION, HTTP_STATUS_OK, log_error, Response::next, obix_fcgi_sendStaticErrorMessage(), obixRequest_release(), obixResponse_free(), obixResponse_setError(), _Request::r, Response::request, Response::uri, and XML_HEADER.

Referenced by obix_fcgi_handleRequest(), and obix_fcgi_init().

7.29.2.8 void obix_fcgi_sendStaticErrorMessage (Request * *request*)

Definition at line 356 of file obix_fcgi.c.

References ERROR_STATIC, HTTP_STATUS_OK, obixRequest_release(), _Request::r, and XML_HEADER.

Referenced by obix_fcgi_handleRequest(), and obix_fcgi_sendResponse().

7.29.2.9 void obix_fcgi_shutdown (FCGX_Request * *request*)

Definition at line 249 of file obix_fcgi.c.

References _requestListMutex, obix_server_shutdown(), and obixRequest_free().

Referenced by main().

7.29.2.10 static int obixRequest_create () [static]

Definition at line 578 of file obix_fcgi.c.

References _requestIds, _Request::id, LISTENSOCK_FILENO, LISTENSOCK_FLAGS, log_error, _Request::next, and _Request::r.

Referenced by obixRequest_get().

7.29.2.11 static void obixRequest_free (Request * *request*) [static]

Definition at line 647 of file obix_fcgi.c.

References _Request::next, _Request::r, and TRUE.

Referenced by obix_fcgi_shutdown().

7.29.2.12 static Request * obixRequest_get () [static]

Definition at line 607 of file obix_fcgi.c.

References _requestListMutex, _requestMaxCount, _requestsInUse, log_error, obixRequest_create(), obixRequest_getHead(), and obixRequest_wait().

Referenced by main().

7.29.2.13 static Request* obixRequest_getHead () [static]

Definition at line 552 of file obix_fcgi.c.

References _requestMaxCount, _requestsInUse, _Request::canWait, FALSE, _Request::next, and TRUE.

Referenced by obixRequest_get(), and obixRequest_wait().

7.29.2.14 static void obixRequest_release (Request * *request*) [static]

Definition at line 542 of file obix_fcgi.c.

References _requestListMutex, _requestListNew, _requestsInUse, and _Request::next.

Referenced by obix_fcgi_sendResponse(), and obix_fcgi_sendStaticErrorMessage().

7.29.2.15 static Request* obixRequest_wait () [static]

Definition at line 571 of file obix_fcgi.c.

References _requestListMutex, _requestListNew, and obixRequest_getHead().

Referenced by obixRequest_get().

7.29.2.16 static char* parseArguments (int *argc*, char ** *argv*) [static]

Definition at line 85 of file obix_fcgi.c.

References log_debug, log_useSyslog(), and log_warning.

Referenced by main().

7.29.3 Variable Documentation

7.29.3.1 int _requestIds = 0 [static]

Definition at line 76 of file obix_fcgi.c.

Referenced by obixRequest_create().

7.29.3.2 Request* _requestList [static]

Definition at line 74 of file obix_fcgi.c.

7.29.3.3 pthread_mutex_t _requestListMutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 78 of file obix_fcgi.c.

Referenced by obix_fcgi_shutdown(), obixRequest_get(), obixRequest_release(), and obixRequest_wait().

7.29.3.4 pthread_cond_t _requestListNew = PTHREAD_COND_INITIALIZER

Definition at line 79 of file obix_fcgi.c.

Referenced by obixRequest_release(), and obixRequest_wait().

7.29.3.5 int _requestMaxCount = MAX_PARALLEL_REQUEST_DEFAULT + 1 [static]

Definition at line 77 of file obix_fcgi.c.

Referenced by obix_fcgi_loadConfig(), obixRequest_get(), and obixRequest_getHead().

7.29.3.6 int _requestsInUse = 0 [static]

Definition at line 75 of file obix_fcgi.c.

Referenced by obixRequest_get(), obixRequest_getHead(), and obixRequest_release().

7.29.3.7 const char* CONFIG_FILE = "server_config.xml" [static]

Definition at line 43 of file obix_fcgi.c.

Referenced by obix_fcgi_loadConfig().

7.29.3.8 const char* CT_HOLD_REQUEST_MAX = "hold-request-max" [static]

Definition at line 45 of file obix_fcgi.c.

Referenced by obix_fcgi_loadConfig().

7.29.3.9 const char* ERROR_STATIC [static]**Initial value:**

```
"<err displayName=\"Internal Server Error\" "
```

```
"display=\"Unable to process the request.\"
"This is a static error message which is "
"returned when things go really bad./>\""
```

Header of error answer.

Note that according oBIX specification, oBIX error messages should still have Status OK header

Definition at line 62 of file obix_fcgi.c.

Referenced by obix_fcgi_sendStaticErrorMessage().

7.29.3.10 const char* HTTP_CONTENT_LOCATION = "Content-Location: %s\r\n" [static]

Definition at line 51 of file obix_fcgi.c.

Referenced by obix_fcgi_sendResponse().

7.29.3.11 const char* HTTP_STATUS_OK [static]

Initial value:

```
"Status: 200 OK\r\n"
"Content-Type: text/xml\r\n"
```

Standard header of any server answer.

Definition at line 48 of file obix_fcgi.c.

Referenced by obix_fcgi_sendResponse(), and obix_fcgi_sendStaticErrorMessage().

7.29.3.12 const char* XML_HEADER = "\r\n<?xml version=\"1.0\"" encoding=\"UTF-8\"?>\r\n" [static]

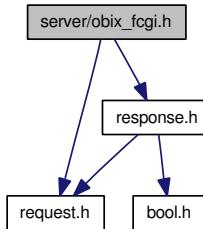
Definition at line 52 of file obix_fcgi.c.

Referenced by obix_fcgi_sendResponse(), and obix_fcgi_sendStaticErrorMessage().

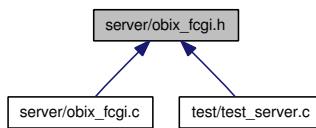
7.30 server/obix_fcgi.h File Reference

```
#include "request.h"
#include "response.h"
```

Include dependency graph for obix_fcgi.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `obix_fcgi_dumpEnvironment (Response *response)`
- void `obix_fcgi_handleRequest (Request *request)`
- int `obix_fcgi_init ()`
- char * `obix_fcgi_readRequestInput (Request *request)`
- void `obix_fcgi_sendResponse (Response *response)`
- void `obix_fcgi_sendStaticErrorMessage (Request *request)`
- void `obix_fcgi_shutdown ()`

7.30.1 Detailed Description

Todo

add description

Author:

Andrey Litvinov

Version:

1.0

Definition in file [obix_fcgi.h](#).

7.30.2 Function Documentation

7.30.2.1 void `obix_fcgi_dumpEnvironment (Response * response)`

Definition at line 462 of file [obix_fcgi.c](#).

References Response::body, FALSE, log_debug, log_error, Response::next, obixResponse_add(), obixResponse_send(), obixResponse_setText(), _Request::r, Response::request, TRUE, and xmldb_getDump().

Referenced by obix_fcgi_handleRequest().

7.30.2.2 void obix_fcgi_handleRequest (Request * *request*)

Definition at line 262 of file obix_fcgi.c.

References _Request::canWait, log_error, log_warning, OBIX_CONTRACT_ERR_UNSUPPORTED, obix_fcgi_dumpEnvironment(), obix_fcgi_readRequestInput(), obix_fcgi_sendResponse(), obix_fcgi_sendStaticErrorMessage(), obix_server_generateObixErrorMessage(), obix_server_handleGET(), obix_server_handlePOST(), obix_server_handlePUT(), obixResponse_create(), _Request::r, xmldb_compareServerAddr(), and xmldb_getServerAddressLength().

Referenced by main().

7.30.2.3 int obix_fcgi_init ()

7.30.2.4 char* obix_fcgi_readRequestInput (Request * *request*)

Definition at line 413 of file obix_fcgi.c.

References log_debug, log_error, and _Request::r.

Referenced by obix_fcgi_handleRequest().

7.30.2.5 void obix_fcgi_sendResponse (Response * *response*)

Definition at line 366 of file obix_fcgi.c.

References Response::body, HTTP_CONTENT_LOCATION, HTTP_STATUS_OK, log_error, Response::next, obix_fcgi_sendStaticErrorMessage(), obixRequest_release(), obixResponse_free(), obixResponse_setError(), _Request::r, Response::request, Response::uri, and XML_HEADER.

Referenced by obix_fcgi_handleRequest(), and obix_fcgi_init().

7.30.2.6 void obix_fcgi_sendStaticErrorMessage (Request * *request*)

Definition at line 356 of file obix_fcgi.c.

References ERROR_STATIC, HTTP_STATUS_OK, obixRequest_release(), _Request::r, and XML_HEADER.

Referenced by obix_fcgi_handleRequest(), and obix_fcgi_sendResponse().

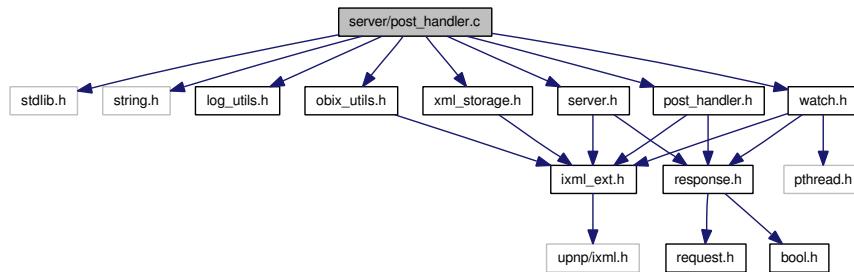
7.30.2.7 void obix_fcgi_shutdown ()

7.31 server/post_handler.c File Reference

Contains handlers for various oBIX invoke commands.

```
#include <stdlib.h>
#include <string.h>
#include <log_utils.h>
#include <obix_utils.h>
#include "xml_storage.h"
#include "watch.h"
#include "server.h"
#include "post_handler.h"
```

Include dependency graph for post_handler.c:



Defines

- #define BATCH_OUT_POSTFIX "\r\n</list>\r\n"
- #define BATCH_OUT_PREFIX "<list is=\\"obix:BatchOut\\" of=\\"obix:obj\\">\r\n"
- #define DEVICE_LIST_URI "/obix/devices/"
- #define WATCH_OUT_POSTFIX "\r\n</list>\r\n</obj>\r\n"
- #define WATCH_OUT_PREFIX "<obj is=\\"obix:WatchOut\\">\r\n<list name=\\"values\\" of=\\"obix:obj\\">\r\n"

Functions

- static Response * addResponsePart (Response *respHead, Response *respTail, const char *uri, const char *operationName)

Adds new response part to the tail of the response and sends the error message if creation failed.
- static void completeWatchPollResponse (const char *operationName, Response *respHead, Response *respTail, const char *uri)
- static int getUriSet (IXML_NodeList *nodeList, const char ***uriSet)

Returns array of unique URI's retrieved from the list of <uri> objects.

- void `handlerBatch` (`Response` *response, const char *uri, `IXML_Element` *input)
- void `handlerError` (`Response` *response, const char *uri, `IXML_Element` *input)
- void `handlerSignUp` (`Response` *response, const char *uri, `IXML_Element` *input)
- void `handlerWatchAdd` (`Response` *response, const char *uri, `IXML_Element` *input)
- void `handlerWatchDelete` (`Response` *response, const char *uri, `IXML_Element` *input)
- void `handlerWatchLongPoll` (`oBIX_Watch` *watch, `Response` *response, const char *uri)

This method is used to perform delayed poll response.

- void `handlerWatchPollChanges` (`Response` *response, const char *uri, `IXML_Element` *input)
- static void `handlerWatchPollHelper` (`Response` *response, const char *uri, `BOOL` changedOnly)

Common function to handle both pollChanges and pollRefresh calls.

- void `handlerWatchPollRefresh` (`Response` *response, const char *uri, `IXML_Element` *input)
- void `handlerWatchRemove` (`Response` *response, const char *uri, `IXML_Element` *input)
- void `handlerWatchServiceMake` (`Response` *response, const char *uri, `IXML_Element` *input)
- `obix_server_postHandler` `obix_server_getPostHandler` (int id)
- static `Response` * `pollWatchItemIterator` (const char *operationName, `BOOL` changedOnly, `oBIX_Watch` *watch, `Response` *response, const char *uri)
- static int `processWatchIn` (`IXML_Element` *input, const char ***uriSet)
- static int `putDeviceReference` (`IXML_Element` *newDevice)
- static void `sendErrorMessage` (`Response` *response, const char *uri, const char *operationName, const char *message)

Helper method which is used to send error message when some handler fails.

Variables

- static const `obix_server_postHandler` `POST_HANDLER` []
- static const int `POST_HANDLERS_COUNT` = 9

7.31.1 Detailed Description

Contains handlers for various oBIX invoke commands.

The following oBIX commands are implemented:

- `watchService.make` Creates new Watch object.
- `Watch.add` Adds object to the watch list.
- `Watch.remove` Removes object from the watch list.
- `Watch.pollChanges` Returns objects which are changed since last poll.
- `Watch.pollRefresh` Returns all objects from the watch list.
- `Watch.delete` Deletes the Watch object.
- `signUp` Adds new device data to the server.

Author:

Andrey Litvinov

Version:

1.0

Definition in file [post_handler.c](#).**7.31.2 Define Documentation****7.31.2.1 #define BATCH_OUT_POSTFIX "\r\n</list>\r\n"**

Definition at line 53 of file post_handler.c.

Referenced by handlerBatch().

7.31.2.2 #define BATCH_OUT_PREFIX "<list is=\"obix:BatchOut\" of=\"obix:obj\">\r\n"

Definition at line 52 of file post_handler.c.

Referenced by handlerBatch().

7.31.2.3 #define DEVICE_LIST_URI "/obix/devices/"

Definition at line 55 of file post_handler.c.

7.31.2.4 #define WATCH_OUT_POSTFIX "\r\n </list>\r\n</obj>\r\n"

Definition at line 50 of file post_handler.c.

Referenced by completeWatchPollResponse(), and handlerWatchAdd().

7.31.2.5 #define WATCH_OUT_PREFIX "<obj is=\"obix:WatchOut\">\r\n <list name=\"values\" of=\"obix:obj\">\r\n"

Definition at line 49 of file post_handler.c.

Referenced by handlerWatchAdd(), and handlerWatchPollHelper().

7.31.3 Function Documentation**7.31.3.1 static Response* addResponsePart (Response * respHead, Response * respTail, const char * uri, const char * operationName) [static]**

Adds new response part to the tail of the response and sends the error message if creation failed.

Parameters:

respHead Head of the response (used when the error message is sent).

respTail Tail of the response to which a new part will be added.

uri Uri of the operation which invoked this function. It will be used in the error message.

handlerName Name of the operation handler which will appear in the error message.

Returns:

New response object which is a next part of the provided response; *NULL* on error.

Definition at line 302 of file post_handler.c.

References log_error, obixResponse_add(), and sendErrorMessage().

Referenced by completeWatchPollResponse(), handlerBatch(), handlerWatchAdd(), and pollWatchItemIterator().

7.31.3.2 static void completeWatchPollResponse (const char * *operationName*, Response * *respHead*, Response * *respTail*, const char * *uri*) [static]

Definition at line 508 of file post_handler.c.

References addResponsePart(), obixResponse_send(), obixResponse_setText(), TRUE, and WATCH_OUT_POSTFIX.

Referenced by handlerWatchLongPoll(), and handlerWatchPollHelper().

7.31.3.3 static int getUriSet (IXML_NodeList * *nodeList*, const char * *uriSet*) [static]**

Returns array of unique URI's retrieved from the list of <uri> objects.

Some elements of the array can be *NULL* which mean that they were filtered out.

Parameters:

nodeList List of <uri> objects.

uriSet The address where array of URI's will be written to.

Returns:

The size of the returned URI set, or negative error code.

Definition at line 217 of file post_handler.c.

References ixmlNode_convertToElement(), and OBIX_ATTR_VAL.

Referenced by processWatchIn().

7.31.3.4 void handlerBatch (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 908 of file post_handler.c.

References addResponsePart(), BATCH_OUT_POSTFIX, BATCH_OUT_PREFIX, ixmlelement_getNode(), ixmleNode_convertToElement(), Response::next, OBIX_ATTR_VAL, obix_obj_implementsContract(), obix_server_invoke(), obix_server_read(), obix_server_write(), obixResponse_send(), obixResponse_setText(), sendErrorMessage(), and TRUE.

7.31.3.5 void handlerError (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 113 of file post_handler.c.

References log_debug, OBIX_CONTRACT_ERR_UNSUPPORTED, obix_server_generateObixErrorMessage(), and obixResponse_send().

7.31.3.6 void handlerSignUp (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 839 of file post_handler.c.

References log_debug, OBIX_ATTR_HREF, obix_server_generateResponse(), obixResponse_isError(), obixResponse_send(), putDeviceReference(), sendErrorMessage(), TRUE, xmldb_delete(), xmldb_getServerAddressLength(), and xmldb_putDOM().

7.31.3.7 void handlerWatchAdd (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 319 of file post_handler.c.

References addResponsePart(), oBIX_Watch_Item::doc, FALSE, log_debug, OBIX_CONTRACT_ERR_BAD_URI, obix_server_generateObixErrorMessage(), obix_server_generateResponse(), OBIX_WATCHLEASE_NO_CHANGE, obixResponse_send(), obixResponse_setText(), obixWatch_createWatchItem(), obixWatch_getByUri(), obixWatch_resetLeaseTimer(), processWatchIn(), sendErrorMessage(), TRUE, WATCH_OUT_POSTFIX, and WATCH_OUT_PREFIX.

7.31.3.8 void handlerWatchDelete (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 713 of file post_handler.c.

References log_debug, OBIX_OBJ_NULL_TEMPLATE, obixResponse_send(), obixResponse_setText(), obixWatch_delete(), obixWatch_getByUri(), sendErrorMessage(), and TRUE.

7.31.3.9 void handlerWatchLongPoll (oBIX_Watch * *watch*, Response * *response*, const char * *uri*)

This method is used to perform delayed poll response.

Definition at line 571 of file post_handler.c.

References completeWatchPollResponse(), log_debug, pollWatchItemIterator(), and TRUE.

Referenced by handlerWatchPollHelper().

7.31.3.10 void handlerWatchPollChanges (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 697 of file post_handler.c.

References handlerWatchPollHelper(), and TRUE.

7.31.3.11 static void handlerWatchPollHelper (Response * *response*, const char * *uri*, BOOL *changedOnly*) [static]

Common function to handle both pollChanges and pollRefresh calls.

Definition at line 594 of file post_handler.c.

References completeWatchPollResponse(), FALSE, handlerWatchLongPoll(), Response::next, OBIX_WATCHLEASE_NO_CHANGE, obixResponse_free(), obixResponse_setText(), obixWatch_getByUri(), obixWatch_holdPoll(), obixWatch_isLongPoll(), obixWatch_resetLeaseTimer(), pollWatchItemIterator(), sendErrorMessage(), TRUE, and WATCH_OUT_PREFIX.

Referenced by handlerWatchPollChanges(), and handlerWatchPollRefresh().

7.31.3.12 void handlerWatchPollRefresh (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 705 of file post_handler.c.

References FALSE, and handlerWatchPollHelper().

7.31.3.13 void handlerWatchRemove (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 442 of file post_handler.c.

References log_debug, OBIX_OBJ_NULL_TEMPLATE, OBIX_WATCHLEASE_NO_CHANGE, obixResponse_send(), obixResponse_setText(), obixWatch_deleteWatchItem(), obixWatch_getByUri(), obixWatch_resetLeaseTimer(), processWatchIn(), sendErrorMessage(), and TRUE.

7.31.3.14 void handlerWatchServiceMake (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 158 of file post_handler.c.

References FALSE, ixmlelement_freeOwnerDocument(), log_debug, log_warning, OBIX_ATTR_HREF, obix_server_generateResponse(), obixResponse_send(), obixWatch_create(), sendErrorMessage(), and TRUE.

7.31.3.15 obix_server_postHandler obix_server_getPostHandler (int *id*)

Definition at line 101 of file post_handler.c.

References POST_HANDLER, and POST_HANDLERS_COUNT.

Referenced by obix_server_invoke().

7.31.3.16 static Response* pollWatchItemIterator (const char * *operationName*, BOOL *changedOnly*, oBIX_Watch * *watch*, Response * *response*, const char * *uri*) [static]

Definition at line 525 of file post_handler.c.

References addResponsePart(), oBIX_Watch_Item::doc, FALSE, oBIX_Watch::items, oBIX_Watch_Item::next, obix_server_generateResponse(), obixWatchItem_isUpdated(), obixWatchItem_setUpdated(), TRUE, and oBIX_Watch_Item::uri.

Referenced by handlerWatchLongPoll(), and handlerWatchPollHelper().

7.31.3.17 static int processWatchIn (IXML_Element * *input*, const char *** *uriSet*) [static]

Definition at line 264 of file post_handler.c.

References getUriSet(), and OBIX_OBJ_URI.

Referenced by handlerWatchAdd(), and handlerWatchRemove().

7.31.3.18 static int putDeviceReference (IXML_Element * *newDevice*) [static]

Definition at line 760 of file post_handler.c.

References FALSE, ixmlelement_copyAttributeWithLog(), ixmlelement_getNode(), log_error, OBIX_ATTR_DISPLAY, OBIX_ATTR_DISPLAY_NAME, OBIX_ATTR_HREF, OBIX_ATTR_NAME, OBIX_OBJ_REF, TRUE, and xmldb_getDOM().

Referenced by handlerSignUp().

7.31.3.19 static void sendErrorMessage (Response * *response*, const char * *uri*, const char * *operationName*, const char * *message*) [static]

Helper method which is used to send error message when some handler fails.

Parameters:

response Response object.

operationName Name of the failed operation.

message Message which should be sent.

Definition at line 133 of file post_handler.c.

References log_warning, Response::next, obix_server_generateObixErrorMessage(), obixResponse_free(), and obixResponse_send().

Referenced by addResponsePart(), handlerBatch(), handlerSignUp(), handlerWatchAdd(), handlerWatchDelete(), handlerWatchPollHelper(), handlerWatchRemove(), and handlerWatchServiceMake().

7.31.4 Variable Documentation

7.31.4.1 const obix_server_postHandler POST_HANDLER[] [static]

Initial value:

```
{  
    &handlerError,  
    &handlerWatchServiceMake,  
    &handlerWatchAdd,  
    &handlerWatchRemove,  
    &handlerWatchPollChanges,  
    &handlerWatchPollRefresh,  
    &handlerWatchDelete,  
    &handlerSignUp,  
    &handlerBatch  
}
```

Definition at line 86 of file post_handler.c.

Referenced by obix_server_getPostHandler().

7.31.4.2 const int POST_HANDLERS_COUNT = 9 [static]

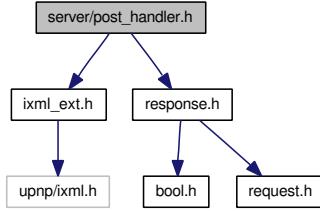
Definition at line 99 of file post_handler.c.

Referenced by obix_server_getPostHandler().

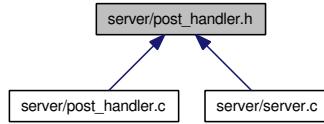
7.32 server/post_handler.h File Reference

```
#include <ixml_ext.h>  
#include "response.h"
```

Include dependency graph for post_handler.h:



This graph shows which files directly or indirectly include this file:



TypeDefs

- `typedef void(* obix_server_postHandler)(Response *response, const char *uri, IXML_Element *input)`

Functions

- `obix_server_postHandler obix_server_getPostHandler (int id)`

7.32.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [post_handler.h](#).

7.32.2 Typedef Documentation

7.32.2.1 `typedef void(* obix_server_postHandler)(Response *response, const char *uri, IXML_Element *input)`

Todo

describe me

Definition at line 36 of file post_handler.h.

7.32.3 Function Documentation

7.32.3.1 obix_server_postHandler obix_server_getPostHandler (int *id*)

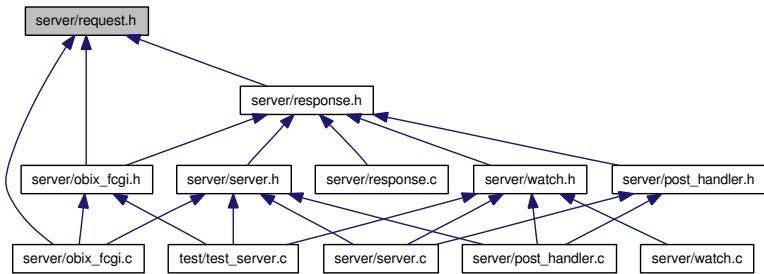
Definition at line 101 of file post_handler.c.

References POST_HANDLER, and POST_HANDLERS_COUNT.

Referenced by obix_server_invoke().

7.33 server/request.h File Reference

This graph shows which files directly or indirectly include this file:



TypeDefs

- [typedef struct _Request Request](#)

7.33.1 Typedef Documentation

7.33.1.1 [typedef struct _Request Request](#)

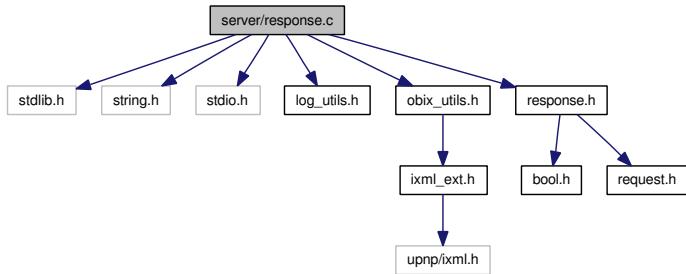
Definition at line 32 of file request.h.

7.34 server/response.c File Reference

```

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <log_utils.h>
#include <obix_utils.h>
#include "response.h"
  
```

Include dependency graph for response.c:



Functions

- `Response * obixResponse_add (Response *response)`
- `Response * obixResponse_create (Request *request, BOOL canWait)`
- `void obixResponse_free (Response *response)`
- `BOOL obixResponse_isError (Response *response)`
- `BOOL obixResponse_isHead (Response *response)`
- `int obixResponse_send (Response *response)`
- `int obixResponse_setError (Response *response, char *description)`
- `void obixResponse_setErrorFlag (Response *response, BOOL error)`
- `void obixResponse_setListener (obix_response_listener listener)`
- `void obixResponse_setRightUri (Response *response, const char *requestUri, int slashFlag)`
- `int obixResponse_setText (Response *response, const char *text, BOOL copy)`

Variables

- `static obix_response_listener _responseListener = NULL`
- `const char * OBIX_OBJ_ERR_TEMPLATE`

7.34.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [response.c](#).

7.34.2 Function Documentation

7.34.2.1 Response* obixResponse_add (Response * *response*)

Definition at line 66 of file response.c.

References FALSE, Response::next, and obixResponse_create().

Referenced by addResponsePart(), and obix_fcgi_dumpEnvironment().

7.34.2.2 Response* obixResponse_create (Request * *request*, BOOL *canWait*)

Definition at line 48 of file response.c.

References Response::body, Response::canWait, Response::error, FALSE, Response::next, Response::request, and Response::uri.

Referenced by obix_fcgi_handleRequest(), obixResponse_add(), testGenerateResponse(), testPutHandler(), testResponse_setRightUri(), testSignUpHelper(), testWatch(), testWatchPollChanges(), and testWatchRemove().

7.34.2.3 void obixResponse_free (Response * *response*)

Definition at line 76 of file response.c.

References Response::body, Response::next, obixResponse_free(), and Response::uri.

Referenced by handlerWatchPollHelper(), obix_fcgi_sendResponse(), obixResponse_free(), sendErrorMessage(), testGenerateResponse(), testPutHandler(), testResponse_setRightUri(), testSignUpHelper(), testWatch(), testWatchPollChanges(), and testWatchRemove().

7.34.2.4 BOOL obixResponse_isError (Response * *response*)

Definition at line 133 of file response.c.

References Response::error, and TRUE.

Referenced by handlerSignUp().

7.34.2.5 BOOL obixResponse_isHead (Response * *response*)

Definition at line 199 of file response.c.

References FALSE, Response::request, and TRUE.

Referenced by obix_server_generateResponse(), and obixResponse_send().

7.34.2.6 int obixResponse_send (Response * *response*)

Definition at line 204 of file response.c.

References obixResponse_isHead().

Referenced by completeWatchPollResponse(), handlerBatch(), handlerError(), handlerSignUp(), handlerWatchAdd(), handlerWatchDelete(), handlerWatchRemove(), handlerWatchServiceMake(), obix_fcgi_dumpEnvironment(), obix_server_handleGET(), obix_server_handlePUT(), obix_server_invoke(), and sendErrorMessage().

7.34.2.7 int obixResponse_setError (Response * *response*, char * *description*)

Definition at line 97 of file response.c.

References Response::body, Response::error, log_error, OBIX_OBJ_ERR_TEMPLATE, and TRUE.

Referenced by obix_fcgi_sendResponse(), and obix_server_generateResponse().

7.34.2.8 void obixResponse_setErrorFlag (Response * *response*, BOOL *error*)

Definition at line 128 of file response.c.

References Response::error.

Referenced by obix_server_generateObixErrorMessage().

7.34.2.9 void obixResponse_setListener (obix_response_listener *listener*)

Definition at line 43 of file response.c.

References _responseListener.

Referenced by obix_fcgi_init(), testSignUpHelper(), testWatch(), testWatchPollChanges(), and testWatchRemove().

7.34.2.10 void obixResponse_setRightUri (Response * *response*, const char * *requestUri*, int *slashFlag*)

Definition at line 178 of file response.c.

References log_error, and Response::uri.

Referenced by obix_server_invoke(), and testResponse_setRightUri().

7.34.2.11 int obixResponse_setText (Response * *response*, const char * *text*, BOOL *copy*)

Definition at line 145 of file response.c.

References Response::body, and log_error.

Referenced by completeWatchPollResponse(), handlerBatch(), handlerWatchAdd(), handlerWatchDelete(), handlerWatchPollHelper(), handlerWatchRemove(), obix_fcgi_dumpEnvironment(), and obix_server_generateResponse().

7.34.3 Variable Documentation

7.34.3.1 obix_response_listener _responseListener = NULL [static]

Definition at line 41 of file response.c.

Referenced by obixResponse_setListener().

7.34.3.2 const char* OBIX_OBJ_ERR_TEMPLATE

Initial value:

```
"<err displayName=\"Internal Server Error\""
      " display=\"%s\"/ >"
```

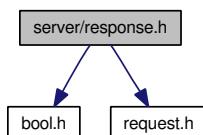
Definition at line 37 of file response.c.

Referenced by obixResponse_setError().

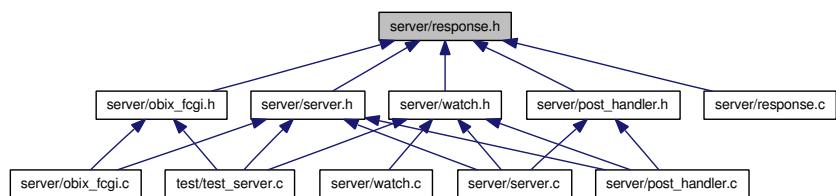
7.35 server/response.h File Reference

```
#include "bool.h"
#include "request.h"
```

Include dependency graph for response.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [Response](#)

TypeDefs

- [typedef void\(* obix_response_listener \)\(Response *response\)](#)

Functions

- [Response * obixResponse_add \(Response *response\)](#)
- [Response * obixResponse_create \(Request *request, BOOL canWait\)](#)
- [void obixResponse_free \(Response *response\)](#)
- [BOOL obixResponse_isError \(Response *response\)](#)
- [BOOL obixResponse_isHead \(Response *response\)](#)
- [int obixResponse_send \(Response *response\)](#)
- [int obixResponse_setError \(Response *response, char *description\)](#)
- [void obixResponse_setErrorFlag \(Response *response, BOOL error\)](#)
- [void obixResponse_setListener \(obix_response_listener listener\)](#)
- [void obixResponse_setRightUri \(Response *response, const char *requestUri, int slashFlag\)](#)
- [int obixResponse_setText \(Response *response, const char *text, BOOL copy\)](#)

7.35.1 Detailed Description

[Todo](#)

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [response.h](#).

7.35.2 TypeDef Documentation

7.35.2.1 [typedef void\(* obix_response_listener\)\(Response *response\)](#)

Definition at line 46 of file [response.h](#).

7.35.3 Function Documentation

7.35.3.1 [Response* obixResponse_add \(Response * response\)](#)

Definition at line 66 of file response.c.

References FALSE, Response::next, and obixResponse_create().

Referenced by addResponsePart(), and obix_fcgi_dumpEnvironment().

7.35.3.2 Response* obixResponse_create (Request * *request*, BOOL *canWait*)

Definition at line 48 of file response.c.

References Response::body, Response::canWait, Response::error, FALSE, Response::next, Response::request, and Response::uri.

Referenced by obix_fcgi_handleRequest(), obixResponse_add(), testGenerateResponse(), testPutHandler(), testResponse_setRightUri(), testSignUpHelper(), testWatch(), testWatchPollChanges(), and testWatchRemove().

7.35.3.3 void obixResponse_free (Response * *response*)

Definition at line 76 of file response.c.

References Response::body, Response::next, obixResponse_free(), and Response::uri.

Referenced by handlerWatchPollHelper(), obix_fcgi_sendResponse(), obixResponse_free(), sendErrorMessage(), testGenerateResponse(), testPutHandler(), testResponse_setRightUri(), testSignUpHelper(), testWatch(), testWatchPollChanges(), and testWatchRemove().

7.35.3.4 BOOL obixResponse_isError (Response * *response*)

Definition at line 133 of file response.c.

References Response::error, and TRUE.

Referenced by handlerSignUp().

7.35.3.5 BOOL obixResponse_isHead (Response * *response*)

Definition at line 199 of file response.c.

References FALSE, Response::request, and TRUE.

Referenced by obix_server_generateResponse(), and obixResponse_send().

7.35.3.6 int obixResponse_send (Response * *response*)

Definition at line 204 of file response.c.

References obixResponse_isHead().

Referenced by completeWatchPollResponse(), handlerBatch(), handlerError(), handlerSignUp(), handlerWatchAdd(), handlerWatchDelete(), handlerWatchRemove(), handlerWatchServiceMake(), obix_fcgi_dumpEnvironment(), obix_server_handleGET(), obix_server_handlePUT(), obix_server_invoke(), and sendErrorMessage().

7.35.3.7 int obixResponse_setError (Response * *response*, char * *description*)

Definition at line 97 of file response.c.

References Response::body, Response::error, log_error, OBIX_OBJ_ERR_TEMPLATE, and TRUE.

Referenced by obix_fcgi_sendResponse(), and obix_server_generateResponse().

7.35.3.8 void obixResponse_setErrorFlag (Response * *response*, BOOL *error*)

Definition at line 128 of file response.c.

References Response::error.

Referenced by obix_server_generateObixErrorMessage().

7.35.3.9 void obixResponse_setListener (obix_response_listener *listener*)

Definition at line 43 of file response.c.

References _responseListener.

Referenced by obix_fcgi_init(), testSignUpHelper(), testWatch(), testWatchPollChanges(), and testWatchRemove().

7.35.3.10 void obixResponse_setRightUri (Response * *response*, const char * *requestUri*, int *slashFlag*)

Definition at line 178 of file response.c.

References log_error, and Response::uri.

Referenced by obix_server_invoke(), and testResponse_setRightUri().

7.35.3.11 int obixResponse_setText (Response * *response*, const char * *text*, BOOL *copy*)

Definition at line 145 of file response.c.

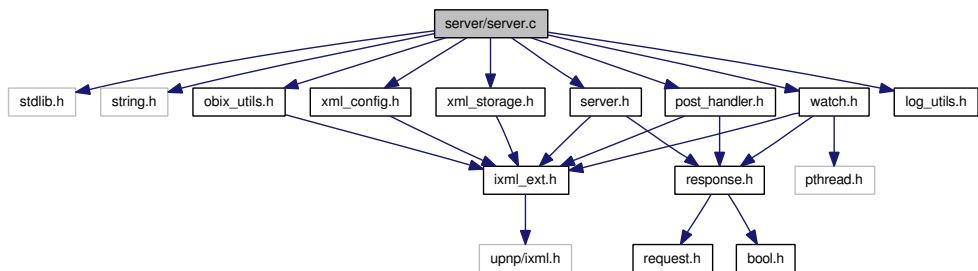
References Response::body, and log_error.

Referenced by completeWatchPollResponse(), handlerBatch(), handlerWatchAdd(), handlerWatchDelete(), handlerWatchPollHelper(), handlerWatchRemove(), obix_fcgi_dumpEnvironment(), and obix_server_generateResponse().

7.36 server/server.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <obix_utils.h>
#include <xml_config.h>
#include <log_utils.h>
#include "xml_storage.h"
#include "post_handler.h"
#include "watch.h"
#include "server.h"
```

Include dependency graph for server.c:



Defines

- #define LISTENSOCK_FILENO 0
- #define LISTENSOCK_FLAGS 0

Functions

- static char * normalizeObixDocument (IXML_Element *oBIXdoc, char *fullUri, **BOOL** addXmllns, **BOOL** saveChanges)

Adds standard attributes to the parent node of the document and returns string representation.
- static char * normalizeUri (const char *requestUri, IXML_Element *doc, int slashFlag)

Returns full URI corresponding to the request URI.
- void obix_server_generateObixErrorMessage (**Response** *response, const char *uri, const char *type, const char *name, const char *desc)
- void obix_server_generateResponse (**Response** *response, IXML_Element *doc, const char *requestUri, **BOOL** changeUri, **BOOL** useObjectUri, int slashFlag, **BOOL** saveChanges)
- void obix_server_handleGET (**Response** *response, const char *uri)
- void obix_server_handlePOST (**Response** *response, const char *uri, const char *input)
- void obix_server_handlePUT (**Response** *response, const char *uri, const char *input)
- int obix_server_init (IXML_Element *settings)
- void obix_server_invoke (**Response** *response, const char *uri, IXML_Element *input)
- void obix_server_read (**Response** *response, const char *uri)

- void `obix_server_shutdown ()`
- void `obix_server_write (Response *response, const char *uri, IXML_Element *input)`
- static void `updateMetaWatch (IXML_Node *node)`

Updates meta watch attributes recursively for the object and its parents.

Variables

- static const char * `CT_SERVER_ADDRESS` = "server-address"
- static const char * `OBIX_META_ATTR_OP` = "op"

7.36.1 Detailed Description

Todo

add description there

Definition in file [server.c](#).

7.36.2 Define Documentation

7.36.2.1 #define LISTENSOCK_FILENO 0

Definition at line 36 of file [server.c](#).

7.36.2.2 #define LISTENSOCK_FLAGS 0

Definition at line 37 of file [server.c](#).

7.36.3 Function Documentation

7.36.3.1 static char* normalizeObixDocument (IXML_Element * *oBIXdoc*, char * *fullUri*, BOOL *addXmlns*, BOOL *saveChanges*) [static]

Adds standard attributes to the parent node of the document and returns string representation.

- Modifies 'href' attribute to contain full URI including server address
- Adds following attributes:
 - xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 - xsi:schemaLocation="http://obix.org/ns/schema/1.0"
 - xmlns="http://obix.org/ns/schema/1.0"
- Removes *handler* attribute from <op> nodes.

Parameters:

oBIXdoc Pointer to the root node of the oBIX document.
fullUri URI containing address starting from the root of the server.
addXmlns If true than xmlns attributes are added to the parent tag.
saveChanges If TRUE saves changes in the original DOM structure, otherwise modifies a copy.

Returns:

string representation of the document. **Don't forget** to free memory after usage.

Definition at line 458 of file server.c.

References Response::error, ixmlelement_cloneWithLog(), ixmlelement_freeOwnerDocument(), ixmlelement_getNode(), ixmlelement_setAttributeWithLog(), log_error, OBIX_ATTR_HREF, and removeMetaInfo().

Referenced by obix_server_generateResponse().

7.36.3.2 static char* normalizeUri (const char * *requestUri*, IXML_Element * *doc*, int *slashFlag*) [static]

Returns full URI corresponding to the request URI.

Adds server address to the requested URI and also adds/removes ending slash ('/') if needed.

Parameters:

requestUri requested URI. It should be absolute (show address from the server root)
doc requested oBIX document. Object should contain *href* attribute. If NULL is provided than full URI is created only based on request URI.
slashFlag flag indicating difference in ending slash between requested URI and object's URI.

Returns:

Full URI. If the original oBIX document already contains full URI then *NULL* will be returned.

Note:

Don't forget to free memory after usage.

Definition at line 397 of file server.c.

References log_error, OBIX_ATTR_HREF, xmldb_compareServerAddr(), and xmldb_getFullUri().

Referenced by obix_server_generateResponse().

7.36.3.3 void obix_server_generateObixErrorMessage (Response * *response*, const char * *uri*, const char * *type*, const char * *name*, const char * *desc*)

Definition at line 77 of file server.c.

References `Response::error`, `FALSE`, `ixmlElement_freeOwnerDocument()`, `ixmlElement_setAttributeWithLog()`, `log_error`, `OBIX_ATTR_DISPLAY`, `OBIX_ATTR_DISPLAY_NAME`, `OBIX_ATTR_IS`, `obix_server_generateResponse()`, `OBIX_SYS_ERROR_STUB`, `obixResponse_setErrorFlag()`, `TRUE`, and `xmldb_getObixSysObject()`.

Referenced by `handlerError()`, `handlerWatchAdd()`, `obix_fcgi_handleRequest()`, `obix_server_invoke()`, `obix_server_read()`, `obix_server_write()`, and `sendErrorMessage()`.

7.36.3.4 void obix_server_generateResponse (Response * *response*, IXML_Element * *doc*, const char * *requestUri*, BOOL *changeUri*, BOOL *useObjectUri*, int *slashFlag*, BOOL *saveChanges*)

Definition at line 506 of file server.c.

References `FALSE`, `log_error`, `normalizeObixDocument()`, `normalizeUri()`, `OBIX_ATTR_HREF`, `obixResponse_isHead()`, `obixResponse_setError()`, `obixResponse_setText()`, and `Response::uri`.

Referenced by `handlerSignUp()`, `handlerWatchAdd()`, `handlerWatchServiceMake()`, `obix_server_generateObixErrorMessage()`, `obix_server_read()`, `obix_server_write()`, `pollWatchItemIterator()`, and `testGenerateResponse()`.

7.36.3.5 void obix_server_handleGET (Response * *response*, const char * *uri*)

Definition at line 162 of file server.c.

References `obix_server_read()`, and `obixResponse_send()`.

Referenced by `obix_fcgi_handleRequest()`.

7.36.3.6 void obix_server_handlePOST (Response * *response*, const char * *uri*, const char * *input*)

Definition at line 358 of file server.c.

References `ixmlElement_freeOwnerDocument()`, `ixmlElement_parseBuffer()`, and `obix_server_invoke()`.

Referenced by `obix_fcgi_handleRequest()`, `testSignUpHelper()`, `testWatch()`, `testWatchPollChanges()`, and `testWatchRemove()`.

7.36.3.7 void obix_server_handlePUT (Response * *response*, const char * *uri*, const char * *input*)

Definition at line 278 of file server.c.

References `ixmlElement_freeOwnerDocument()`, `ixmlElement_parseBuffer()`, `obix_server_write()`, and `obixResponse_send()`.

Referenced by `obix_fcgi_handleRequest()`, `testPutHandler()`, and `testWatch()`.

7.36.3.8 int obix_server_init (IXML_Element * *settings*)

Definition at line 43 of file server.c.

References config_getChildTag(), config_getTagAttributeValue(), CT_SERVER_ADDRESS, CTA_VALUE, Response::error, log_error, obixWatch_init(), TRUE, and xmldb_init().

Referenced by obix_fcgi_init().

7.36.3.9 void obix_server_invoke (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 298 of file server.c.

References getMetaInfo(), log_debug, OBIX_CONTRACT_ERR_BAD_URI, OBIX_META_ATTR_OP, OBIX_OBJ_OP, obix_server_generateObixErrorMessage(), obix_server_getPostHandler(), obixResponse_send(), obixResponse_setRightUri(), and xmldb_getDOM().

Referenced by handlerBatch(), and obix_server_handlePOST().

7.36.3.10 void obix_server_read (Response * *response*, const char * *uri*)

Definition at line 129 of file server.c.

References FALSE, log_warning, OBIX_CONTRACT_ERR_BAD_URI, obix_server_generateObixErrorMessage(), obix_server_generateResponse(), OBIX_WATCHLEASE_NOCHANGE, obixWatch_getByUri(), obixWatch_resetLeaseTimer(), TRUE, and xmldb_getDOM().

Referenced by handlerBatch(), and obix_server_handleGET().

7.36.3.11 void obix_server_shutdown ()

Definition at line 373 of file server.c.

References log_debug, obixWatch_dispose(), and xmldb_dispose().

Referenced by obix_fcgi_shutdown().

7.36.3.12 void obix_server_write (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 192 of file server.c.

References Response::error, FALSE, ixmlelement_getNode(), log_warning, OBIX_CONTRACT_ERR_BAD_URI, OBIX_CONTRACT_ERR_PERMISSION, obix_server_generateObixErrorMessage(), obix_server_generateResponse(), obixWatch_processTimeUpdates(), TRUE, updateMetaWatch(), and xmldb_updateDOM().

Referenced by handlerBatch(), and obix_server_handlePUT().

7.36.3.13 static void updateMetaWatch (IXML_Node * *node*) [static]

Updates meta watch attributes recursively for the object and its parents.

All these attributes are set to the updated state. So that all subscribed Watches could see that the value was updated.

Todo

think about the whole conception of meta attributes (how to unify its usage)

Definition at line 175 of file server.c.

References getMetaInfo(), ixmleNode_convertToElement(), and obixWatch_updateMeta().

Referenced by obix_server_write().

7.36.4 Variable Documentation

7.36.4.1 const char* CT_SERVER_ADDRESS = "server-address" [static]

Definition at line 39 of file server.c.

7.36.4.2 const char* OBIX_META_ATTR_OP = "op" [static]

Definition at line 41 of file server.c.

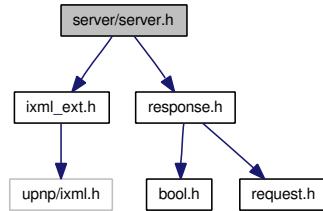
Referenced by obix_server_invoke().

7.37 server/server.h File Reference

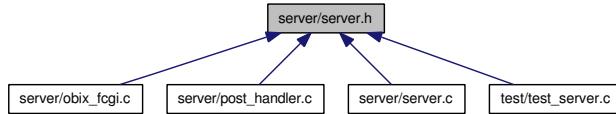
TODO add description there.

```
#include <ixml_ext.h>
#include "response.h"
```

Include dependency graph for server.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [obix_server_generateObixErrorMessage](#) ([Response](#) *response, const char *uri, const char *type, const char *name, const char *desc)
- void [obix_server_generateResponse](#) ([Response](#) *response, [IXML_Element](#) *doc, const char *requestUri, [BOOL](#) changeUri, [BOOL](#) useObjectUri, int slashFlag, [BOOL](#) saveChanges)
- void [obix_server_handleGET](#) ([Response](#) *response, const char *uri)
- void [obix_server_handlePOST](#) ([Response](#) *response, const char *uri, const char *input)
- void [obix_server_handlePUT](#) ([Response](#) *response, const char *uri, const char *input)
- int [obix_server_init](#) ([IXML_Element](#) *settings)
- void [obix_server_invoke](#) ([Response](#) *response, const char *uri, [IXML_Element](#) *input)
- void [obix_server_read](#) ([Response](#) *response, const char *uri)
- void [obix_server_shutdown](#) ()
- void [obix_server_write](#) ([Response](#) *response, const char *uri, [IXML_Element](#) *input)

7.37.1 Detailed Description

TODO add description there.

Definition in file [server.h](#).

7.37.2 Function Documentation

7.37.2.1 void obix_server_generateObixErrorMessage ([Response](#) * *response*, const char * *uri*, const char * *type*, const char * *name*, const char * *desc*)

Definition at line 77 of file [server.c](#).

References [Response::error](#), [FALSE](#), [ixmlElement_freeOwnerDocument\(\)](#), [ixmlElement_setAttributeWithLog\(\)](#), [log_error](#), [OBIX_ATTR_DISPLAY](#), [OBIX_ATTR_DISPLAY_NAME](#), [OBIX_ATTR_IS](#), [obix_server_generateResponse\(\)](#), [OBIX_SYS_ERROR_STUB](#), [obixResponse_setErrorFlag\(\)](#), [TRUE](#), and [xmlDb_getObixSysObject\(\)](#).

Referenced by [handlerError\(\)](#), [handlerWatchAdd\(\)](#), [obix_fcgi_handleRequest\(\)](#), [obix_server_invoke\(\)](#), [obix_server_read\(\)](#), [obix_server_write\(\)](#), and [sendErrorMessage\(\)](#).

7.37.2.2 void obix_server_generateResponse ([Response](#) * *response*, [IXML_Element](#) * *doc*, const char * *requestUri*, [BOOL](#) *changeUri*, [BOOL](#) *useObjectUri*, int *slashFlag*, [BOOL](#) *saveChanges*)

Definition at line 506 of file [server.c](#).

References FALSE, log_error, normalizeObixDocument(), normalizeUri(), OBIX_ATTR_HREF, obixResponse_isHead(), obixResponse_setError(), obixResponse_setText(), and Response::uri.

Referenced by handlerSignUp(), handlerWatchAdd(), handlerWatchServiceMake(), obix_server_generateObixErrorMessage(), obix_server_read(), obix_server_write(), pollWatchItemIterator(), and testGenerateResponse().

7.37.2.3 void obix_server_handleGET (Response * *response*, const char * *uri*)

Definition at line 162 of file server.c.

References obix_server_read(), and obixResponse_send().

Referenced by obix_fcgi_handleRequest().

7.37.2.4 void obix_server_handlePOST (Response * *response*, const char * *uri*, const char * *input*)

Definition at line 358 of file server.c.

References ixmlelement_freeOwnerDocument(), ixmlelement_parseBuffer(), and obix_server_invoke().

Referenced by obix_fcgi_handleRequest(), testSignUpHelper(), testWatch(), testWatchPollChanges(), and testWatchRemove().

7.37.2.5 void obix_server_handlePUT (Response * *response*, const char * *uri*, const char * *input*)

Definition at line 278 of file server.c.

References ixmlelement_freeOwnerDocument(), ixmlelement_parseBuffer(), obix_server_write(), and obixResponse_send().

Referenced by obix_fcgi_handleRequest(), testPutHandler(), and testWatch().

7.37.2.6 int obix_server_init (IXML_Element * *settings*)

Definition at line 43 of file server.c.

References config_getChildTag(), config_getTagAttributeValue(), CT_SERVER_ADDRESS, CTA_VALUE, Response::error, log_error, obixWatch_init(), TRUE, and xmldb_init().

Referenced by obix_fcgi_init().

7.37.2.7 void obix_server_invoke (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 298 of file server.c.

References getMetaInfo(), log_debug, OBIX_CONTRACT_ERR_BAD_URI, OBIX_META_ATTR_OP, OBIX_OBJ_OP, obix_server_generateObixErrorMessage(), obix_server_getPostHandler(),

obixResponse_send(), obixResponse_setRightUri(), and xmldb_getDOM().

Referenced by handlerBatch(), and obix_server_handlePOST().

7.37.2.8 void obix_server_read (Response * *response*, const char * *uri*)

Definition at line 129 of file server.c.

References FALSE, log_warning, OBIX_CONTRACT_ERR_BAD_URI, obix_server_generateObixErrorMessage(), obix_server_generateResponse(), OBIX_WATCHLEASE_NOCHANGE, obixWatch_getByUri(), obixWatch_resetLeaseTimer(), TRUE, and xmldb_getDOM().

Referenced by handlerBatch(), and obix_server_handleGET().

7.37.2.9 void obix_server_shutdown ()

Definition at line 373 of file server.c.

References log_debug, obixWatch_dispose(), and xmldb_dispose().

Referenced by obix_fcgi_shutdown().

7.37.2.10 void obix_server_write (Response * *response*, const char * *uri*, IXML_Element * *input*)

Definition at line 192 of file server.c.

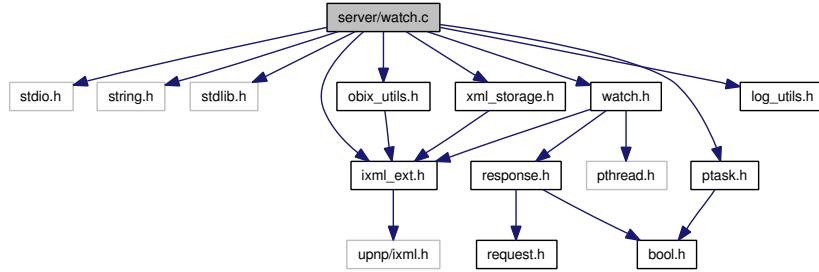
References Response::error, FALSE, ixmlelement_getNode(), log_warning, OBIX_CONTRACT_ERR_BAD_URI, OBIX_CONTRACT_ERR_PERMISSION, obix_server_generateObixErrorMessage(), obix_server_generateResponse(), obixWatch_processTimeUpdates(), TRUE, updateMetaWatch(), and xmldb_updateDOM().

Referenced by handlerBatch(), and obix_server_handlePUT().

7.38 server/watch.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ixml_ext.h>
#include <ptask.h>
#include <log_utils.h>
#include <obix_utils.h>
#include "xml_storage.h"
#include "watch.h"
```

Include dependency graph for watch.c:



Data Structures

- struct [PollTaskParams](#)

Functions

- static int [findWatchItem](#) ([oBIX_Watch](#) *watch, const char *watchItemUri, [oBIX_Watch_Item](#) **item, [oBIX_Watch_Item](#) **parent)
- static char * [generateWatchUri](#) (int watchId)
- static long [getLeaseTime](#) (IXML_Element *watchDOM)
- int [obixWatch_appendWatchItem](#) ([oBIX_Watch](#) *watch, [oBIX_Watch_Item](#) *item)
- int [obixWatch_create](#) (IXML_Element **watchDOM)

Creates new oBIX Watch instance.

- int [obixWatch_createWatchItem](#) ([oBIX_Watch](#) *watch, const char *uri, [oBIX_Watch_Item](#) **watchItem)
- int [obixWatch_delete](#) ([oBIX_Watch](#) *watch)

Deletes watch instance.

- static int [obixWatch_deleteHelper](#) ([oBIX_Watch](#) *watch)
- int [obixWatch_deleteWatchItem](#) ([oBIX_Watch](#) *watch, const char *watchItemUri)
- int [obixWatch_dispose](#) ()
- static int [obixWatch_free](#) ([oBIX_Watch](#) *watch)
- [oBIX_Watch](#) * [obixWatch_get](#) (int watchId)
- [oBIX_Watch](#) * [obixWatch_getByUri](#) (const char *uri)
- char * [obixWatch_getUri](#) ([oBIX_Watch](#) *watch)
- [oBIX_Watch_Item](#) * [obixWatch_getWatchItem](#) ([oBIX_Watch](#) *watch, const char *watchItemUri)
- int [obixWatch_holdPoll](#) ([obixWatch_pollHandler](#) pollHandler, [oBIX_Watch](#) *watch, [Response](#) *response, const char *uri, [BOOL](#) maxWait)
- int [obixWatch_init](#) ()
- [BOOL](#) [obixWatch_isLongPoll](#) ([oBIX_Watch](#) *watch)
- [BOOL](#) [obixWatch_isWatchUri](#) (const char *uri)
- void [obixWatch_longPollTask](#) (void *arg)
- static void [obixWatch_notifyPollTask](#) (IXML_Element *meta)
- int [obixWatch_processTimeUpdates](#) (const char *uri, IXML_Element *element)
- int [obixWatch_resetLeaseTimer](#) ([oBIX_Watch](#) *watch, long newPeriod)
- void [obixWatch_updateMeta](#) (IXML_Element *meta)
- static [oBIX_Watch_Item](#) * [obixWatchItem_free](#) ([oBIX_Watch_Item](#) *item)

Frees memory, allocated for provided watch item.

- static void `obixWatchItem_freeRecursive (oBIX_Watch_Item *item)`
- `BOOL obixWatchItem_isUpdated (oBIX_Watch_Item *item)`
- int `obixWatchItem_setUpdated (oBIX_Watch_Item *item, BOOL isUpdated)`
- static void `taskDeleteWatch (void *arg)`

Variables

- static `Task_Thread * _threadLease`
Thread for removing unused watches.
- static `Task_Thread * _threadLongPoll`
Thread for parking long poll requests.
- static `oBIX_Watch ** _watches`
Array for storing Watch objects created by users.
- static int `_watchesCount`
- static const int `MAX_WATCH_COUNT = 50`
- static const char * `OBIX_META_ATTR_WATCH_TEMPLATE = "wi-%d"`
- const char * `OBIX_META_WATCH_UPDATED_NO = "n"`
- const char * `OBIX_META_WATCH_UPDATED_YES = "y"`
- const long `OBIX_WATCH_LEASE_NO_CHANGE = -1`
- static const int `WATCH_URI_PREFIX_LENGTH = 24`
- static const char * `WATCH_URI_TEMPLATE = "/obix/watchService/watch%d/"`

7.38.1 Detailed Description

Todo

add description

Author:

Andrey Litvinov

Version:

1.0

Definition in file [watch.c](#).

7.38.2 Function Documentation

7.38.2.1 static int findWatchItem (oBIX_Watch * *watch*, const char * *watchItemUri*, oBIX_Watch_Item ** *item*, oBIX_Watch_Item ** *parent*) [static]

Definition at line 595 of file [watch.c](#).

References `oBIX_Watch::items`, `oBIX_Watch_Item::next`, and `oBIX_Watch_Item::uri`.

Referenced by `obixWatch_deleteWatchItem()`, and `obixWatch_getWatchItem()`.

7.38.2.2 static char* generateWatchUri (int *watchId*) [static]

Definition at line 267 of file watch.c.

References log_error, WATCH_URI_PREFIX_LENGTH, and WATCH_URI_TEMPLATE.

Referenced by obixWatch_create(), and obixWatch_getUri().

7.38.2.3 static long getLeaseTime (IXML_Element * *watchDOM*) [static]

Definition at line 287 of file watch.c.

References ixmlelement_getNode(), log_error, OBIX_ATTR_VAL, OBIX_OBJ_RELTIME, and obix_reltimes_parseToLong().

Referenced by obixWatch_create().

7.38.2.4 int obixWatch_appendWatchItem (oBIX_Watch * *watch*, oBIX_Watch_Item * *item*)

Definition at line 568 of file watch.c.

References oBIX_Watch::items, log_error, and oBIX_Watch_Item::next.

Referenced by obixWatch_createWatchItem().

7.38.2.5 int obixWatch_create (IXML_Element ** *watchDOM*)

Creates new oBIX Watch instance.

Todo

Move error codes to the global error enum.

Parameters:

watch Address where pointer to the new oBIX Watch object will be written to.

Returns:

- **>0** - Index of the new watch object;
- **-1** - Unable to allocate enough memory for a new object;
- **-2** - Maximum number of watches is reached;
- **-3** - Storage error - unable to save watch;
- **-4** - Internal error.

Definition at line 321 of file watch.c.

References `_watchesCount`, `FALSE`, `generateWatchUri()`, `getLeaseTime()`, `oBIX_Watch::id`, `oBIX_Watch::isPollWaitingMax`, `oBIX_Watch::items`, `ixmlElement_freeOwnerDocument()`, `ixmlElement_getNode()`, `ixmlElement_setAttributeWithLog()`, `oBIX_Watch::leaseTimerId`, `log_error`, `log_warning`, `MAX_WATCH_COUNT`, `OBIX_ATTR_HREF`, `OBIX_SYS_WATCH_STUB`, `oBIX_Watch::pollTaskCompleted`, `oBIX_Watch::pollTaskId`, `oBIX_Watch::pollTaskMutex`, `oBIX_Watch::pollWaitMax`, `oBIX_Watch::pollWaitMin`, `ptask_schedule()`, `taskDeleteWatch()`, `xmldb_getObixSysObject()`, and `xmldb_put()`.

Referenced by `handlerWatchServiceMake()`.

7.38.2.6 int obixWatch_createWatchItem (oBIX_Watch * *watch*, const char * *uri*, oBIX_Watch_Item ** *watchItem*)

Todo

describe me; rename to `createWatchItem`

Parameters:

watch

uri

watchItem

Returns:

- *0* - Watch item is added successfully;
- *-1* - URI not found;
- *-2* - URI is an `<op>` object;
- *-3* - Internal error.

Definition at line 487 of file `watch.c`.

References `oBIX_Watch_Item::doc`, `oBIX_Watch::id`, `log_error`, `oBIX_Watch_Item::next`, `OBIX_META_ATTR_WATCH_TEMPLATE`, `OBIX_META_WATCH_UPDATED_NO`, `OBIX_OBJ_OP`, `obixWatch_appendWatchItem()`, `obixWatch_getWatchItem()`, `oBIX_Watch_Item::updated`, `oBIX_Watch_Item::uri`, `xmldb_deleteMeta()`, `xmldb_getDOM()`, and `xmldb_putMeta()`.

Referenced by `handlerWatchAdd()`.

7.38.2.7 int obixWatch_delete (oBIX_Watch * *watch*)

Deletes watch instance.

Parameters:

watch Watch instance to be deleted.

Returns:

0 on success, negative error code otherwise.

Definition at line 241 of file watch.c.

References oBIX_Watch::leaseTimerId, log_error, obixWatch_deleteHelper(), ptask_cancel(), and TRUE.
Referenced by handlerWatchDelete(), and obixWatch_dispose().

7.38.2.8 static int obixWatch_deleteHelper (oBIX_Watch * *watch*) [static]

Definition at line 193 of file watch.c.

References FALSE, log_error, obixWatch_free(), obixWatch_getUri(), oBIX_Watch::pollTaskCompleted, oBIX_Watch::pollTaskId, oBIX_Watch::pollTaskMutex, ptask_reschedule(), and xmldb_delete().
Referenced by obixWatch_delete(), and taskDeleteWatch().

7.38.2.9 int obixWatch_deleteWatchItem (oBIX_Watch * *watch*, const char * *watchItemUri*)

Definition at line 634 of file watch.c.

References findWatchItem(), oBIX_Watch::items, log_error, oBIX_Watch_Item::next, and obixWatchItem_free().

Referenced by handlerWatchRemove().

7.38.2.10 int obixWatch_dispose ()

Definition at line 108 of file watch.c.

References _watchesCount, MAX_WATCH_COUNT, obixWatch_delete(), ptask_dispose(), and TRUE.
Referenced by obix_server_shutdown().

7.38.2.11 static int obixWatch_free (oBIX_Watch * *watch*) [static]

Definition at line 171 of file watch.c.

References _watchesCount, oBIX_Watch::id, oBIX_Watch::items, obixWatchItem_freeRecursive(), oBIX_Watch::pollTaskCompleted, and oBIX_Watch::pollTaskMutex.

Referenced by obixWatch_deleteHelper().

7.38.2.12 oBIX_Watch* obixWatch_get (int *watchId*)

Definition at line 447 of file watch.c.

References log_warning, and MAX_WATCH_COUNT.

Referenced by obixWatch_getByUri(), and obixWatch_notifyPollTask().

7.38.2.13 oBIX_Watch* obixWatch_getByUri (const char * *uri*)

Definition at line 466 of file watch.c.

References log_warning, obixWatch_get(), obixWatch_isWatchUri(), and WATCH_URI_PREFIX_LENGTH.

Referenced by handlerWatchAdd(), handlerWatchDelete(), handlerWatchPollHelper(), handlerWatchRemove(), obix_server_read(), and obixWatch_processTimeUpdates().

7.38.2.14 char* obixWatch_getUri (oBIX_Watch * *watch*)

Definition at line 461 of file watch.c.

References generateWatchUri(), and oBIX_Watch::id.

Referenced by obixWatch_deleteHelper().

7.38.2.15 oBIX_Watch_Item* obixWatch_getWatchItem (oBIX_Watch * *watch*, const char * *watchItemUri*)

Definition at line 665 of file watch.c.

References findWatchItem(), and log_error.

Referenced by obixWatch_createWatchItem().

7.38.2.16 int obixWatch_holdPoll (obixWatch_pollHandler *pollHandler*, oBIX_Watch * *watch*, Response * *response*, const char * *uri*, BOOL *maxWait*)

Definition at line 900 of file watch.c.

References Response::canWait, oBIX_Watch::isPollWaitingMax, log_debug, log_error, obixWatch_longPollTask(), PollTaskParams::pollHandler, oBIX_Watch::pollTaskId, oBIX_Watch::pollTaskMutex, oBIX_Watch::pollWaitMax, oBIX_Watch::pollWaitMin, ptask_schedule(), PollTaskParams::response, PollTaskParams::uri, and PollTaskParams::watch.

Referenced by handlerWatchPollHelper().

7.38.2.17 int obixWatch_init ()

Definition at line 73 of file watch.c.

References _watchesCount, log_debug, log_error, log_warning, MAX_WATCH_COUNT, and ptask_init().

Referenced by obix_server_init(), and test_server().

7.38.2.18 BOOL obixWatch_isLongPoll (oBIX_Watch * *watch*)

Definition at line 881 of file watch.c.

References FALSE, oBIX_Watch::pollWaitMax, and TRUE.

Referenced by handlerWatchPollHelper().

7.38.2.19 BOOL obixWatch_isWatchUri (const char * *uri*)

Definition at line 771 of file watch.c.

References FALSE, TRUE, WATCH_URI_PREFIX_LENGTH, and WATCH_URI_TEMPLATE.

Referenced by obixWatch_getByUri().

7.38.2.20 void obixWatch_longPollTask (void * *arg*)

Definition at line 887 of file watch.c.

References PollTaskParams::pollHandler, oBIX_Watch::pollTaskCompleted, oBIX_Watch::pollTaskId, oBIX_Watch::pollTaskMutex, PollTaskParams::response, PollTaskParams::uri, and PollTaskParams::watch.

Referenced by obixWatch_holdPoll().

7.38.2.21 static void obixWatch_notifyPollTask (IXML_Element * *meta*) [static]

Definition at line 704 of file watch.c.

References FALSE, oBIX_Watch::isPollWaitingMax, log_error, obixWatch_get(), oBIX_Watch::pollTaskId, oBIX_Watch::pollTaskMutex, oBIX_Watch::pollWaitMax, oBIX_Watch::pollWaitMin, ptask_reschedule(), and TRUE.

Referenced by obixWatch_updateMeta().

7.38.2.22 int obixWatch_processTimeUpdates (const char * *uri*, IXML_Element * *element*)

Definition at line 814 of file watch.c.

References oBIX_Watch::id, log_warning, OBIX_ATTR_VAL, obix_reltimes_parseToLong(), obixWatch_getByUri(), obixWatch_resetLeaseTimer(), oBIX_Watch::pollWaitMax, and oBIX_Watch::pollWaitMin.

Referenced by obix_server_write().

7.38.2.23 int obixWatch_resetLeaseTimer (oBIX_Watch * *watch*, long *newPeriod*)

Definition at line 783 of file watch.c.

References FALSE, oBIX_Watch::id, oBIX_Watch::leaseTimerId, log_error, ptask_reschedule(), and ptask_reset().

Referenced by handlerWatchAdd(), handlerWatchPollHelper(), handlerWatchRemove(), obix_server_-read(), and obixWatch_processTimeUpdates().

7.38.2.24 void obixWatch_updateMeta (IXML_Element * *meta*)

Definition at line 733 of file watch.c.

References ixmlelement_getNode(), ixmlelement_convertToElement(), log_error, OBIX_ATTR_VAL, OBIX_META_WATCH_UPDATED_NO, OBIX_META_WATCH_UPDATED_YES, and obixWatch_notifyPollTask().

Referenced by updateMetaWatch().

7.38.2.25 static oBIX_Watch_Item* obixWatchItem_free (oBIX_Watch_Item * *item*) [static]

Frees memory, allocated for provided watch item.

Also removes meta tag associated with that watch item.

Parameters:

item Watch item which should be freed.

Returns:

Next watch item in the list, or NULL if the provided watch item was at the end of the list.

Definition at line 148 of file watch.c.

References log_error, oBIX_Watch_Item::next, oBIX_Watch_Item::updated, oBIX_Watch_Item::uri, and xmldb_deleteMeta().

Referenced by obixWatch_deleteWatchItem(), and obixWatchItem_freeRecursive().

7.38.2.26 static void obixWatchItem_freeRecursive (oBIX_Watch_Item * *item*) [static]

Definition at line 161 of file watch.c.

References oBIX_Watch_Item::next, and obixWatchItem_free().

Referenced by obixWatch_free().

7.38.2.27 BOOL obixWatchItem_isUpdated (oBIX_Watch_Item * *item*)

Definition at line 678 of file watch.c.

References FALSE, OBIX_META_WATCH_UPDATED_YES, TRUE, and oBIX_Watch_Item::updated.

Referenced by pollWatchItemIterator().

7.38.2.28 int obixWatchItem_setUpdated (oBIX_Watch_Item * *item*, BOOL *isUpdated*)

Definition at line 695 of file watch.c.

References OBIX_META_WATCH_UPDATED_NO, OBIX_META_WATCH_UPDATED_YES, oBIX_Watch_Item::updated, and xmldb_updateMeta().

Referenced by pollWatchItemIterator().

7.38.2.29 static void taskDeleteWatch (void * *arg*) [static]

Definition at line 255 of file watch.c.

References oBIX_Watch::id, log_debug, log_error, and obixWatch_deleteHelper().

Referenced by obixWatch_create().

7.38.3 Variable Documentation

7.38.3.1 Task_Thread* _threadLease [static]

Thread for removing unused watches.

Definition at line 69 of file watch.c.

7.38.3.2 Task_Thread* _threadLongPoll [static]

Thread for parking long poll requests.

Definition at line 71 of file watch.c.

7.38.3.3 oBIX_Watch** _watches [static]

Array for storing Watch objects created by users.

Definition at line 66 of file watch.c.

7.38.3.4 int _watchesCount [static]

Definition at line 67 of file watch.c.

Referenced by obixWatch_create(), obixWatch_dispose(), obixWatch_free(), and obixWatch_init().

7.38.3.5 **const int MAX_WATCH_COUNT = 50 [static]**

Definition at line 55 of file watch.c.

Referenced by obixWatch_create(), obixWatch_dispose(), obixWatch_get(), and obixWatch_init().

7.38.3.6 **const char* OBIX_META_ATTR_WATCH_TEMPLATE = "wi-%d" [static]**

Definition at line 57 of file watch.c.

Referenced by obixWatch_createWatchItem().

7.38.3.7 **const char* OBIX_META_WATCH_UPDATED_NO = "n"**

Definition at line 50 of file watch.c.

Referenced by obixWatch_createWatchItem(), obixWatch_updateMeta(), and obixWatchItem_setUpdated().

7.38.3.8 **const char* OBIX_META_WATCH_UPDATED_YES = "y"**

Definition at line 49 of file watch.c.

Referenced by obixWatch_updateMeta(), obixWatchItem_isUpdated(), and obixWatchItem_setUpdated().

7.38.3.9 **const long OBIX_WATCHLEASE_NO_CHANGE = -1**

Definition at line 52 of file watch.c.

Referenced by handlerWatchAdd(), handlerWatchPollHelper(), handlerWatchRemove(), and obix_server_read().

7.38.3.10 **const int WATCH_URI_PREFIX_LENGTH = 24 [static]**

Definition at line 61 of file watch.c.

Referenced by generateWatchUri(), obixWatch_getByUri(), and obixWatch_isWatchUri().

7.38.3.11 const char* WATCH_URI_TEMPLATE = "/obix/watchService/watch%d/" [static]

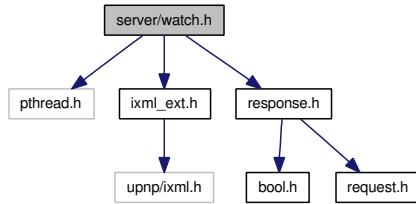
Definition at line 59 of file watch.c.

Referenced by generateWatchUri(), and obixWatch_isWatchUri().

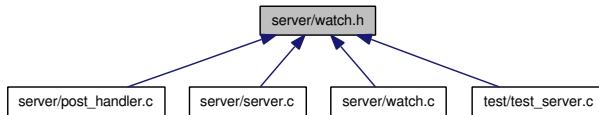
7.39 server/watch.h File Reference

```
#include <pthread.h>
#include <ixml_ext.h>
#include "response.h"
```

Include dependency graph for watch.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [oBIX_Watch](#)
Represents an oBIX Watch object.
- struct [oBIX_Watch_Item](#)
Represents a separate watch item.

TypeDefs

- [typedef void\(* obixWatch_pollHandler \)\(oBIX_Watch *watch, Response *response, const char *uri\)](#)

Functions

[oBIX Watch utilities](#) @{

- [int obixWatch_appendWatchItem \(oBIX_Watch *watch, oBIX_Watch_Item *item\)](#)
- [int obixWatch_create \(IXML_Element **watchDOM\)](#)

Creates new oBIX Watch instance.

- int `obixWatch_createWatchItem` (`oBIX_Watch` *watch, const char *uri, `oBIX_Watch_Item` **watchItem)

- int `obixWatch_delete` (`oBIX_Watch` *watch)

Deletes watch instance.

- int `obixWatch_deleteWatchItem` (`oBIX_Watch` *watch, const char *watchItemUri)
- int `obixWatch_dispose` ()
- `oBIX_Watch` * `obixWatch_get` (int watchId)
- `oBIX_Watch` * `obixWatch_getByUri` (const char *uri)
- char * `obixWatch_getUri` (`oBIX_Watch` *watch)
- `oBIX_Watch_Item` * `obixWatch_getWatchItem` (`oBIX_Watch` *watch, const char *watchItemUri)
- int `obixWatch_holdPoll` (`obixWatch_pollHandler` pollHandler, `oBIX_Watch` *watch, `Response` *response, const char *uri, `BOOL` maxWait)
- int `obixWatch_init` ()
- `BOOL` `obixWatch_isLongPoll` (`oBIX_Watch` *watch)
- `BOOL` `obixWatch_isWatchUri` (const char *uri)
- int `obixWatch_processTimeUpdates` (const char *uri, `IXML_Element` *element)
- int `obixWatch_resetLeaseTimer` (`oBIX_Watch` *watch, long newPeriod)
- void `obixWatch_updateMeta` (`IXML_Element` *meta)
- `BOOL` `obixWatchItem_isUpdated` (`oBIX_Watch_Item` *item)
- int `obixWatchItem_setUpdated` (`oBIX_Watch_Item` *item, `BOOL` isUpdated)

Variables

- const char * `OBIX_META_WATCH_UPDATED_NO`
- const char * `OBIX_META_WATCH_UPDATED_YES`
- const long `OBIX_WATCHLEASE_NO_CHANGE`

7.39.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file `watch.h`.

7.39.2 Typedef Documentation

7.39.2.1 `typedef void(* obixWatch_pollHandler)(oBIX_Watch *watch, Response *response, const char *uri)`

Definition at line 99 of file `watch.h`.

7.39.3 Function Documentation

7.39.3.1 int obixWatch_appendWatchItem (oBIX_Watch * *watch*, oBIX_Watch_Item * *item*)

Definition at line 568 of file watch.c.

References oBIX_Watch::items, log_error, and oBIX_Watch_Item::next.

Referenced by obixWatch_createWatchItem().

7.39.3.2 int obixWatch_create (IXML_Element ** *watchDOM*)

Creates new oBIX Watch instance.

Todo

Move error codes to the global error enum.

Parameters:

watch Address where pointer to the new oBIX Watch object will be written to.

Returns:

- >0 - Index of the new watch object;
- -1 - Unable to allocate enough memory for a new object;
- -2 - Maximum number of watches is reached;
- -3 - Storage error - unable to save watch;
- -4 - Internal error.

Definition at line 321 of file watch.c.

References _watchesCount, FALSE, generateWatchUri(), getLeaseTime(), oBIX_Watch::id, oBIX_Watch::isPollWaitingMax, oBIX_Watch::items, ixmlelement_freeOwnerDocument(), ixmlelement_getNode(), ixmlelement_setAttributeWithLog(), oBIX_Watch::leaseTimerId, log_error, log_warning, MAX_WATCH_COUNT, OBIX_ATTR_HREF, OBIX_SYS_WATCH_STUB, oBIX_Watch::pollTaskCompleted, oBIX_Watch::pollTaskId, oBIX_Watch::pollTaskMutex, oBIX_Watch::pollWaitMax, oBIX_Watch::pollWaitMin, ptask_schedule(), taskDeleteWatch(), xmldb_getObixSysObject(), and xmldb_put().

Referenced by handlerWatchServiceMake().

7.39.3.3 int obixWatch_createWatchItem (oBIX_Watch * *watch*, const char * *uri*, oBIX_Watch_Item ** *watchItem*)

Todo

describe me; rename to createWatchItem

Parameters:

watch
uri
watchItem

Returns:

- *0* - Watch item is added successfully;
- *-1* - URI not found;
- *-2* - URI is an <op> object;
- *-3* - Internal error.

Definition at line 487 of file watch.c.

References oBIX_Watch_Item::doc, oBIX_Watch::id, log_error, oBIX_Watch_Item::next, OBIX_META_ATTR_WATCH_TEMPLATE, OBIX_META_WATCH_UPDATED_NO, OBIX_OBJ_OP, obixWatch_appendWatchItem(), obixWatch_getWatchItem(), oBIX_Watch_Item::updated, oBIX_Watch_Item::uri, xmldb_deleteMeta(), xmldb_getDOM(), and xmldb_putMeta().

Referenced by handlerWatchAdd().

7.39.3.4 int obixWatch_delete (oBIX_Watch * *watch*)

Deletes watch instance.

Parameters:

watch Watch instance to be deleted.

Returns:

0 on success, negative error code otherwise.

Definition at line 241 of file watch.c.

References oBIX_Watch::leaseTimerId, log_error, obixWatch_deleteHelper(), ptask_cancel(), and TRUE.

Referenced by handlerWatchDelete(), and obixWatch_dispose().

7.39.3.5 int obixWatch_deleteWatchItem (oBIX_Watch * *watch*, const char * *watchItemUri*)

Definition at line 634 of file watch.c.

References findWatchItem(), oBIX_Watch::items, log_error, oBIX_Watch_Item::next, and obixWatchItem_free().

Referenced by handlerWatchRemove().

7.39.3.6 int obixWatch_dispose ()

Definition at line 108 of file watch.c.

References _watchesCount, MAX_WATCH_COUNT, obixWatch_delete(), ptask_dispose(), and TRUE.

Referenced by obix_server_shutdown().

7.39.3.7 oBIX_Watch* obixWatch_get (int *watchId*)

Definition at line 447 of file watch.c.

References log_warning, and MAX_WATCH_COUNT.

Referenced by obixWatch_getByUri(), and obixWatch_notifyPollTask().

7.39.3.8 oBIX_Watch* obixWatch_getByUri (const char * *uri*)

Definition at line 466 of file watch.c.

References log_warning, obixWatch_get(), obixWatch_isWatchUri(), and WATCH_URI_PREFIX_LENGTH.

Referenced by handlerWatchAdd(), handlerWatchDelete(), handlerWatchPollHelper(), handlerWatchRemove(), obix_server_read(), and obixWatch_processTimeUpdates().

7.39.3.9 char* obixWatch_getUri (oBIX_Watch * *watch*)

Definition at line 461 of file watch.c.

References generateWatchUri(), and oBIX_Watch::id.

Referenced by obixWatch_deleteHelper().

7.39.3.10 oBIX_Watch_Item* obixWatch_getWatchItem (oBIX_Watch * *watch*, const char * *watchItemUri*)

Definition at line 665 of file watch.c.

References findWatchItem(), and log_error.

Referenced by obixWatch_createWatchItem().

7.39.3.11 int obixWatch_holdPoll (obixWatch_pollHandler *pollHandler*, oBIX_Watch * *watch*, Response * *response*, const char * *uri*, BOOL *maxWait*)

Definition at line 900 of file watch.c.

References Response::canWait, oBIX_Watch::isPollWaitingMax, log_debug, log_error, obixWatch_longPollTask(), PollTaskParams::pollHandler, oBIX_Watch::pollTaskId, oBIX_Watch::pollTaskMutex,

`oBIX_Watch::pollWaitMax`, `oBIX_Watch::pollWaitMin`, `ptask_schedule()`, `PollTaskParams::response`, `PollTaskParams::uri`, and `PollTaskParams::watch`.

Referenced by `handlerWatchPollHelper()`.

7.39.3.12 int obixWatch_init ()

Definition at line 73 of file watch.c.

References `_watchesCount`, `log_debug`, `log_error`, `log_warning`, `MAX_WATCH_COUNT`, and `ptask_init()`.

Referenced by `obix_server_init()`, and `test_server()`.

7.39.3.13 BOOL obixWatch_isLongPoll (oBIX_Watch * *watch*)

Definition at line 881 of file watch.c.

References `FALSE`, `oBIX_Watch::pollWaitMax`, and `TRUE`.

Referenced by `handlerWatchPollHelper()`.

7.39.3.14 BOOL obixWatch_isWatchUri (const char * *uri*)

Definition at line 771 of file watch.c.

References `FALSE`, `TRUE`, `WATCH_URI_PREFIX_LENGTH`, and `WATCH_URI_TEMPLATE`.

Referenced by `obixWatch_getByUri()`.

7.39.3.15 int obixWatch_processTimeUpdates (const char * *uri*, IXML_Element * *element*)

Definition at line 814 of file watch.c.

References `oBIX_Watch::id`, `log_warning`, `OBIX_ATTR_VAL`, `obix_reltimes_parseToLong()`, `obixWatch_getByUri()`, `obixWatch_resetLeaseTimer()`, `oBIX_Watch::pollWaitMax`, and `oBIX_Watch::pollWaitMin`.

Referenced by `obix_server_write()`.

7.39.3.16 int obixWatch_resetLeaseTimer (oBIX_Watch * *watch*, long *newPeriod*)

Definition at line 783 of file watch.c.

References `FALSE`, `oBIX_Watch::id`, `oBIX_Watch::leaseTimerId`, `log_error`, `ptask_reschedule()`, and `ptask_reset()`.

Referenced by `handlerWatchAdd()`, `handlerWatchPollHelper()`, `handlerWatchRemove()`, `obix_server_read()`, and `obixWatch_processTimeUpdates()`.

7.39.3.17 void obixWatch_updateMeta (IXML_Element * *meta*)

Definition at line 733 of file watch.c.

References `ixmlElement_getNode()`, `ixmlNode_convertToElement()`, `log_error`, `OBIX_ATTR_VAL`, `OBIX_META_WATCH_UPDATED_NO`, `OBIX_META_WATCH_UPDATED_YES`, and `obixWatch_notifyPollTask()`.

Referenced by `updateMetaWatch()`.

7.39.3.18 BOOL obixWatchItem_isUpdated (oBIX_Watch_Item * *item*)

Definition at line 678 of file watch.c.

References `FALSE`, `OBIX_META_WATCH_UPDATED_YES`, `TRUE`, and `oBIX_Watch_Item::updated`.

Referenced by `pollWatchItemIterator()`.

7.39.3.19 int obixWatchItem_setUpdated (oBIX_Watch_Item * *item*, BOOL *isUpdated*)

Definition at line 695 of file watch.c.

References `OBIX_META_WATCH_UPDATED_NO`, `OBIX_META_WATCH_UPDATED_YES`, `oBIX_Watch_Item::updated`, and `xmldb_updateMeta()`.

Referenced by `pollWatchItemIterator()`.

7.39.4 Variable Documentation**7.39.4.1 const char* OBIX_META_WATCH_UPDATED_NO**

Definition at line 50 of file watch.c.

Referenced by `obixWatch_createWatchItem()`, `obixWatch_updateMeta()`, and `obixWatchItem_setUpdated()`.

7.39.4.2 const char* OBIX_META_WATCH_UPDATED_YES

Definition at line 49 of file watch.c.

Referenced by `obixWatch_updateMeta()`, `obixWatchItem_isUpdated()`, and `obixWatchItem_setUpdated()`.

7.39.4.3 const long OBIX_WATCHLEASE_NO_CHANGE

Definition at line 52 of file watch.c.

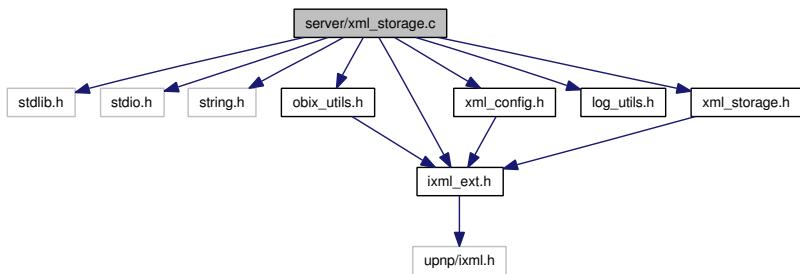
Referenced by handlerWatchAdd(), handlerWatchPollHelper(), handlerWatchRemove(), and obix_server_-read().

7.40 server/xml_storage.c File Reference

Simple implementation of XML storage.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <obix_utils.h>
#include <ixml_ext.h>
#include <xml_config.h>
#include <log_utils.h>
#include "xml_storage.h"
```

Include dependency graph for xml_storage.c:



Functions

- static const char * [checkNode](#) (IXML_Node *node, **BOOL** checkPrefix)
Checks the node to comply with storage standards.
- static int [compare_uri](#) (const char *currentUri, const char *requiredUri, int checked, int *slashFlag)
Compares URIs.
- static int [getLastSlashPosition](#) (const char *str, int startPosition)
- IXML_Element * [getMetaInfo](#) (IXML_Element *doc)
Returns meta tag of the object.
- static IXML_Node * [getNodeByHref](#) (IXML_Document *doc, const char *href, int *slashFlag)
- static IXML_Node * [getNodeByHrefRecursive](#) (IXML_Node *node, const char *href, int checked, int *slashFlag)
Helper function for getNodeByHref(IXML_Document,const char*).*
- static void [insertServerAddress](#) (IXML_Node *node, **BOOL** checkPrefix)
- static void [printXMLContents](#) (IXML_Node *node, const char *title)
- void [removeMetaInfo](#) (IXML_Element *doc)

Removes all OBIX_META tags from the document.

- int `xmldb_compareServerAddr` (const char *uri)
- int `xmldb_delete` (const char *href)

Removes XML node from the storage.

- int `xmldb_deleteMeta` (IXML_Node *meta)
- void `xmldb_dispose` ()

Stops work of the storage and releases all resources.

- char * `xmldb_get` (const char *href, int *slashFlag)

Retrieves XML node with specified URI from the storage.

- IXML_Element * `xmldb_getDOM` (const char *href, int *slashFlag)

Retrieves DOM structure of XML node with specified URI from the storage.

- char * `xmldb_getDump` ()

- char * `xmldb_getFullUri` (const char *absUri, int slashFlag)

1 - add slash -1 - remove slash 0 - do nothing

- IXML_Element * `xmldb_getObixSysObject` (const char *objType)

- const char * `xmldb_getServerAddress` ()

- int `xmldb_getServerAddressLength` ()

- int `xmldb_init` (const char *serverAddr)

Initializes storage.

- int `xmldb_loadFile` (const char *filename)

Loads xml file to the storage.

- void `xmldb_printDump` ()

Prints to standard log current contents of the storage.

- int `xmldb_put` (const char *data)

Returns the remaining slash flag of the last URI comparison operation.

- int `xmldb_putDOM` (IXML_Element *data)

- static int `xmldb_putDOMHelper` (IXML_Element *data, **BOOL** checkPrefix)

- static int `xmldb_putHelper` (const char *data, **BOOL** checkPrefix)

Adds XML node to the storage.

- IXML_Node * `xmldb_putMeta` (IXML_Element *element, const char *name, const char *value)

- int `xmldb_updateDOM` (IXML_Element *input, const char *href, IXML_Element **updatedNode, int *slashFlag)

Updates XML node to the storage.

- int `xmldb_updateMeta` (IXML_Node *meta, const char *newValue)

Variables

- `char * _serverAddress = NULL`
Address of the current server.
- `int _serverAddressLength`
- `static IXML_Document * _storage = NULL`
The place where all data is stored.
- `const char * DEVICE_URL_PREFIX = "/obix"`
- `const int DEVICE_URL_PREFIX_LENGTH = 5`
- `const char * OBIX_META = "meta"`
- `static const char * OBIX_STORAGE_FILES []`
- `static const int OBIX_STORAGE_FILES_COUNT = 7`
- `const char * OBIX_SYS_ERROR_STUB = "/sys/error-stub/"`
- `const char * OBIX_SYS_WATCH_OUT_STUB = "/sys/watch-out-stub/"`
- `const char * OBIX_SYS_WATCH_STUB = "/sys/watch-stub/"`

7.40.1 Detailed Description

Simple implementation of XML storage.

All data is stored in one DOM structure in memory.

Definition in file [xml_storage.c](#).

7.40.2 Function Documentation

7.40.2.1 static const char* checkNode (IXML_Node * *node*, BOOL *checkPrefix*) [static]

Checks the node to comply with storage standards.

Todo

Check that all objects in the input document have required attributes and correct URI's.

Parameters:

node Node to check

checkPrefix If TRUE than all nodes with absolute URIs will be also checked to have /obix prefix.

Returns:

href attribute of the parent node, or *NULL* if check fails.

Definition at line 437 of file [xml_storage.c](#).

References `insertServerAddress()`, `ixmlNode_convertToElement()`, `log_warning`, and `OBIX_ATTR_HREF`.

Referenced by `xmldb_putDOMHelper()`.

7.40.2.2 static int compare_uri (const char * *currentUri*, const char * *requiredUri*, int *checked*, int * *slashFlag*) [static]

Compares URIs.

Helper function. Assumes that requiredUri already contains correct server address and shows only relative address from the server root. checked - number of symbols in requiredUri which were already compared and match with parent node's address.

Returns:

- **0** If currentUri matches with requiredUri assuming that first 'checked' symbols already match.
- **>0** If currentUri matches with part of requiredUri. In that case returning value is a number of symbols already checked in requiredUri.
- **<0** If currentUri doesn't match with requiredUri.

Definition at line 188 of file xml_storage.c.

References _serverAddressLength, FALSE, getLastSlashPosition(), TRUE, and xmldb_compareServerAddr().

Referenced by getNodeByHrefRecursive().

7.40.2.3 static int getLastSlashPosition (const char * *str*, int *startPosition*) [static]

Definition at line 124 of file xml_storage.c.

Referenced by compare_uri().

7.40.2.4 IXML_Element* getMetaInfo (IXML_Element * *doc*)

Returns meta tag of the object.

Parameters:

doc Node whose meta data should be retrieved.

Returns:

Meta tag if exists, *NULL* otherwise.

Definition at line 855 of file xml_storage.c.

References ixmlelement_getNode(), ixmlelement_convertToElement(), and OBIX_META.

Referenced by obix_server_invoke(), updateMetaWatch(), and xmldb_putMeta().

7.40.2.5 static IXML_Node* getNodeByHref (IXML_Document * *doc*, const char * *href*, int * *slashFlag*) [static]

Definition at line 347 of file xml_storage.c.

References _serverAddressLength, getNodeByHrefRecursive(), ixmDocument_getNode(), and xmldb_compareServerAddr().

Referenced by xmldb_delete(), xmldb_get(), xmldb_getDOM(), and xmldb_putDOMHelper().

7.40.2.6 static IXML_Node* getNodeByHrefRecursive (IXML_Node * *node*, const char * *href*, int *checked*, int * *slashFlag*) [static]

Helper function for getNodeByHref(IXML_Document*,const char*).

Parameters:

checked specifies number of symbols from href which are already checked (and match) in parent node.

Definition at line 296 of file xml_storage.c.

References compare_uri(), ixmNode_convertToElement(), OBIX_ATTR_HREF, and OBIX_OBJ_REF.

Referenced by getNodeByHref().

7.40.2.7 static void insertServerAddress (IXML_Node * *node*, BOOL *checkPrefix*) [static]

Definition at line 380 of file xml_storage.c.

References _serverAddress, _serverAddressLength, DEVICE_URI_PREFIX, DEVICE_URI_PREFIX_LENGTH, ixmNode_convertToElement(), log_warning, and OBIX_ATTR_HREF.

Referenced by checkNode().

7.40.2.8 static void printXMLContents (IXML_Node * *node*, const char * *title*) [static]

Definition at line 62 of file xml_storage.c.

References log_debug.

Referenced by xmldb_printDump().

7.40.2.9 void removeMetaInfo (IXML_Element * *doc*)

Removes all **OBIX_META** tags from the document.

For <op> objects these tags contain id of the operation handler. For any other objects they can contain ids of watches subscribed for those object changes.

Parameters:

doc Node which should be cleaned.

Definition at line 892 of file xml_storage.c.

References log_debug, log_warning, and OBIX_META.

Referenced by normalizeObixDocument().

7.40.2.10 int xmldb_compareServerAddr (const char * *uri*)

Definition at line 761 of file xml_storage.c.

References _serverAddress, and _serverAddressLength.

Referenced by compare_uri(), getNodeByHref(), normalizeUri(), and obix_fcgi_handleRequest().

7.40.2.11 int xmldb_delete (const char * *href*)

Removes XML node from the storage.

Parameters:

href address of the node to be deleted.

Definition at line 625 of file xml_storage.c.

References _storage, getNodeByHref(), and log_warning.

Referenced by handlerSignUp(), obixWatch_deleteHelper(), and testDelete().

7.40.2.12 int xmldb_deleteMeta (IXML_Node * *meta*)

Definition at line 833 of file xml_storage.c.

References ixmlelement_getOwnerElement(), ixmlelement_getNode(), ixmlnode_convertToAttr(), and log_error.

Referenced by obixWatch_createWatchItem(), and obixWatchItem_free().

7.40.2.13 void xmldb_dispose ()

Stops work of the storage and releases all resources.

Definition at line 113 of file xml_storage.c.

References _serverAddress, and _storage.

Referenced by obix_server_shutdown().

7.40.2.14 char* xmldb_get (const char * *href*, int * *slashFlag*)

Retrieves XML node with specified URI from the storage.

Parameters:

href address of the XML node to be retrieved.

Returns:

XML data in plain text format or NULL on error.

Definition at line 372 of file xml_storage.c.

References _storage, and getNodeByHref().

Referenced by testDelete(), testSearch(), and testWriteToDatabase().

7.40.2.15 IXML_Element* xmldb_getDOM (const char * *href*, int * *slashFlag*)

Retrieves DOM structure of XML node with specified URI from the storage.

Parameters:

href address of the XML node to be retrieved.

Returns:

XML data as a DOM structure NULL on error.

Definition at line 367 of file xml_storage.c.

References _storage, getNodeByHref(), and ixmlelement_convertToElement().

Referenced by obix_server_invoke(), obix_server_read(), obixWatch_createWatchItem(), putDeviceReference(), testGenerateResponse(), xmldb_getObixSysObject(), and xmldb_updateDOM().

7.40.2.16 char* xmldb_getDump ()

Definition at line 705 of file xml_storage.c.

References _storage, and ixmldocument_getNode().

Referenced by obix_fcgid_dumpEnvironment().

7.40.2.17 char* xmldb_getFullUri (const char * *absUri*, int *slashFlag*)

1 - add slash -1 - remove slash 0 - do nothing

Parameters:

absUri

slashFlag

Returns:

Definition at line 723 of file xml_storage.c.

References _serverAddress, _serverAddressLength, log_error, and log_warning.

Referenced by normalizeUri().

7.40.2.18 IXML_Element* xmldb_getObixSysObject (const char * *objType*)

Definition at line 710 of file xml_storage.c.

References ixmlelement_cloneWithLog(), and xmldb_getDOM().

Referenced by obix_server_generateObixErrorMessage(), and obixWatch_create().

7.40.2.19 const char* xmldb_getServerAddress ()

Definition at line 767 of file xml_storage.c.

References _serverAddress.

7.40.2.20 int xmldb_getServerAddressLength ()

Definition at line 772 of file xml_storage.c.

References _serverAddressLength.

Referenced by handlerSignUp(), and obix_fcgi_handleRequest().

7.40.2.21 int xmldb_init (const char * *serverAddr*)

Initializes storage.

Should be executed only once on startup.

Parameters:

serverAddr of the server. Storage should know it in cases when requested URI contains full address.
If NULL is provided that address will be retrieved from the Lobby object.

Returns:

error code or 0 on success.

Definition at line 69 of file xml_storage.c.

References _serverAddress, _serverAddressLength, _storage, log_debug, log_error, OBIX_STORAGE_FILES, OBIX_STORAGE_FILES_COUNT, and xmldb_loadFile().

Referenced by obix_server_init(), and test_server().

7.40.2.22 int xmldb_loadFile (const char **filename*)

Loads xml file to the storage.

Parameters:

filename name of the xml file to load.

Returns:

error code or 0 on success.

Definition at line 645 of file xml_storage.c.

References config_getResFullPath(), FALSE, log_error, and xmldb_putHelper().

Referenced by test_server(), and xmldb_init().

7.40.2.23 void xmldb_printDump ()

Prints to standard log current contents of the storage.

Definition at line 700 of file xml_storage.c.

References _storage, ixmDocument_getNode(), and printXMLContents().

Referenced by test_server(), and testGenerateResponse().

7.40.2.24 int xmldb_put (const char **data*)

Returns the remaining slash flag of the last URI comparison operation.

URI comparison in database is done ignoring ending slash ('/'). It is done every time when some object is retrieved from the storage. Slash flag shows whether requested URI and URI found in storage contain ending slash.

Slash flag is set by following functions:

- [xmldb_get](#);
- [xmldb_getDOM](#);
- [xmldb_update](#);
- [xmldb_put](#). //
- [xmldb_compareUri](#).

Note:

The value of the flag can be changed also by other xmldb functions.

Returns:

- **0** if both URI had the same ending symbol;
- **1** if the object in the storage had ending slash but requested URI hadn't;
- **-1** if requested URI had ending slash, but the object hadn't. Compares two URI ignoring ending slash ('/'). This function also sets the slash flag.

See also:

`xmldb_getLastUriCompSlashFlag`

Parameters:

uri1 first URI to compare (object's URI for the slash flag).
uri2 second URI to compare (request URI for the slash flag).

Returns:

`TRUE` is URI match, `FALSE` otherwise. Adds new XML node to the storage. The node should contain *href* attribute. If node with the same URI already exists in the storage then new node is not added.

Parameters:

data XML node represented in plain text format.

Returns:

error code or *0* on success.

Definition at line 555 of file `xml_storage.c`.

References `TRUE`, and `xmldb_putHelper()`.

Referenced by `obixWatch_create()`, and `testWriteToDatabase()`.

7.40.2.25 int xmldb_putDOM (IXML_Element * *data*)

Definition at line 525 of file `xml_storage.c`.

References `TRUE`, and `xmldb_putDOMHelper()`.

Referenced by `handlerSignUp()`.

7.40.2.26 static int xmldb_putDOMHelper (IXML_Element * *data*, BOOL *checkPrefix*) [static]

Definition at line 470 of file `xml_storage.c`.

References `_storage`, `checkNode()`, `getNodeByHref()`, `ixmlDocument_getNode()`, `ixmlElement_getNode()`, `log_warning`, and `TRUE`.

Referenced by `xmldb_putDOM()`, and `xmldb_putHelper()`.

7.40.2.27 static int xmldb_putHelper (const char * *data*, BOOL *checkPrefix*) [static]

Adds XML node to the storage.

The data is stored in the root of the document.

Definition at line 534 of file xml_storage.c.

References `ixmlNode_convertToElement()`, `ixmlNode_parseBuffer()`, `log_error`, and `xmldb_putDOMHelper()`.

Referenced by `xmldb_loadFile()`, and `xmldb_put()`.

7.40.2.28 IXML_Node* xmldb_putMeta (IXML_Element * *element*, const char * *name*, const char * *value*)

Definition at line 777 of file xml_storage.c.

References `_storage`, `getMetaInfo()`, `ixmlAttr_getNode()`, `ixmlElement_getNode()`, `log_error`, `OBIX_ATTR_VAL`, and `OBIX_META`.

Referenced by `obixWatch_createWatchItem()`.

7.40.2.29 int xmldb_updateDOM (IXML_Element * *input*, const char * *href*, IXML_Element ** *updatedNode*, int * *slashFlag*)

Updates XML node to the storage.

Only *val* attribute is updated and only for nodes which have *writable* attribute equal to *true*.

Parameters:

data XML node represented in plain text format. The node should contain *val* attribute. Other data is ignored.

href URI of the object to be updated.

updatedNode If not *NULL* is provided and update is successful than the address of the updated node will be written there.

Returns:

- **0** if value is successfully overwritten;
- **1** if request is processed but new value is the same;
- **-1** if request data is corrupted/wrong format;
- **-2** if object with specified href is not found;
- **-3** if object is not writable;
- **-4** if request failed because of internal server error.

Definition at line 560 of file xml_storage.c.

References `ixmlElement_setAttributeWithLog()`, `log_warning`, `OBIX_ATTR_VAL`, `OBIX_ATTR_WRITABLE`, `XML_TRUE`, and `xmldb_getDOM()`.

Referenced by `obix_server_write()`, and `testWriteToDatabase()`.

7.40.2.30 int xmldb_updateMeta (IXML_Node * meta, const char * newValue)

Definition at line 850 of file xml_storage.c.

Referenced by obixWatchItem_setUpdated().

7.40.3 Variable Documentation**7.40.3.1 char* _serverAddress = NULL**

Address of the current server.

Definition at line 59 of file xml_storage.c.

Referenced by insertServerAddress(), xmldb_compareServerAddr(), xmldb_dispose(), xmldb_getFullUri(), xmldb_getServerAddress(), and xmldb_init().

7.40.3.2 int _serverAddressLength

Definition at line 60 of file xml_storage.c.

Referenced by compare_uri(), getNodeByHref(), insertServerAddress(), xmldb_compareServerAddr(), xmldb_getFullUri(), xmldb_getServerAddressLength(), and xmldb_init().

7.40.3.3 IXML_Document* _storage = NULL [static]

The place where all data is stored.

Definition at line 56 of file xml_storage.c.

Referenced by xmldb_delete(), xmldb_dispose(), xmldb_get(), xmldb_getDOM(), xmldb_getDump(), xmldb_init(), xmldb_printDump(), xmldb_putDOMHelper(), and xmldb_putMeta().

7.40.3.4 const char* DEVICE_URI_PREFIX = "/obix"

Definition at line 35 of file xml_storage.c.

Referenced by insertServerAddress().

7.40.3.5 const int DEVICE_URI_PREFIX_LENGTH = 5

Definition at line 36 of file xml_storage.c.

Referenced by insertServerAddress().

7.40.3.6 const char* OBIX_META = "meta"

Definition at line 42 of file xml_storage.c.

Referenced by getMetaInfo(), removeMetaInfo(), testGenerateResponse(), and xmldb_putMeta().

7.40.3.7 const char* OBIX_STORAGE_FILES[] [static]**Initial value:**

```
{ "server_lobby.xml",
  "server_about.xml",
  "server_watch.xml",
  "server_sys_objects.xml",
  "server_devices.xml",
  "server_def.xml",
  "server_test_device.xml"
}
```

Definition at line 44 of file xml_storage.c.

Referenced by xmldb_init().

7.40.3.8 const int OBIX_STORAGE_FILES_COUNT = 7 [static]

Definition at line 53 of file xml_storage.c.

Referenced by xmldb_init().

7.40.3.9 const char* OBIX_SYS_ERROR_STUB = "/sys/error-stub/"

Definition at line 39 of file xml_storage.c.

Referenced by obix_server_generateObixErrorMessage().

7.40.3.10 const char* OBIX_SYS_WATCH_OUT_STUB = "/sys/watch-out-stub/"

Definition at line 40 of file xml_storage.c.

7.40.3.11 const char* OBIX_SYS_WATCH_STUB = "/sys/watch-stub/"

Definition at line 38 of file xml_storage.c.

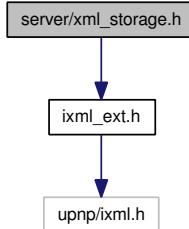
Referenced by obixWatch_create().

7.41 server/xml_storage.h File Reference

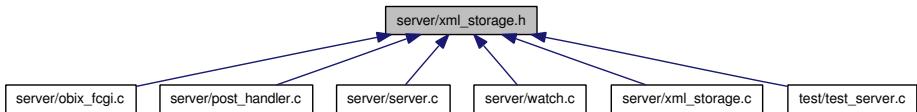
Defines the interface to the XML storage.

```
#include <ixml_ext.h>
```

Include dependency graph for xml_storage.h:



This graph shows which files directly or indirectly include this file:



Functions

- `IXML_Element * getMetaInfo (IXML_Element *doc)`
Returns meta tag of the object.
- `void removeMetaInfo (IXML_Element *doc)`
Removes all OBIX_META tags from the document.
- `int xmldb_compareServerAddr (const char *uri)`
- `int xmldb_delete (const char *href)`
Removes XML node from the storage.
- `int xmldb_deleteMeta (IXML_Node *attr)`
- `void xmldb_dispose ()`
Stops work of the storage and releases all resources.
- `char * xmldb_get (const char *href, int *slashFlag)`
Retrieves XML node with specified URI from the storage.
- `IXML_Element * xmldb_getDOM (const char *href, int *slashFlag)`
Retrieves DOM structure of XML node with specified URI from the storage.
- `char * xmldb_getDump ()`
- `char * xmldb_getFullUri (const char *absUri, int slashFlag)`
1 - add slash -1 - remove slash 0 - do nothing

- IXML_Element * [xmlDb_getObixSysObject](#) (const char *objType)
- const char * [xmlDb_getServerAddress](#) ()
- int [xmlDb_getServerAddressLength](#) ()
- int [xmlDb_init](#) (const char *serverAddr)
Initializes storage.
- int [xmlDb_loadFile](#) (const char *filename)
Loads xml file to the storage.
- void [xmlDb_printDump](#) ()
Prints to standard log current contents of the storage.
- int [xmlDb_put](#) (const char *data)
Returns the remaining slash flag of the last URI comparison operation.
- int [xmlDb_putDOM](#) (IXML_Element *data)
- IXML_Node * [xmlDb_putMeta](#) (IXML_Element *element, const char *name, const char *value)
- int [xmlDb_updateDOM](#) (IXML_Element *input, const char *href, IXML_Element **updatedNode, int *slashFlag)
Updates XML node to the storage.
- int [xmlDb_updateMeta](#) (IXML_Node *meta, const char *newValue)

Variables

- const char * [OBIX_META](#)
- const char * [OBIX_SYS_ERROR_STUB](#)
- const char * [OBIX_SYS_WATCH_OUT_STUB](#)
- const char * [OBIX_SYS_WATCH_STUB](#)

7.41.1 Detailed Description

Defines the interface to the XML storage.

TODO: Add error codes enumeration

Definition in file [xml_storage.h](#).

7.41.2 Function Documentation

7.41.2.1 IXML_Element* [getMetaInfo](#) (IXML_Element * *doc*)

Returns meta tag of the object.

Parameters:

doc Node whose meta data should be retrieved.

Returns:

Meta tag if exists, *NULL* otherwise.

Definition at line 855 of file xml_storage.c.

References ixmlelement_getNode(), ixmlelement_convertToElement(), and OBIX_META.

Referenced by obix_server_invoke(), updateMetaWatch(), and xmldb_putMeta().

7.41.2.2 void removeMetaInfo (IXML_Element * *doc*)

Removes all OBIX_META tags from the document.

For <op> objects these tags contain id of the operation handler. For any other objects they can contain ids of watches subscribed for those object changes.

Parameters:

doc Node which should be cleaned.

Definition at line 892 of file xml_storage.c.

References log_debug, log_warning, and OBIX_META.

Referenced by normalizeObixDocument().

7.41.2.3 int xmldb_compareServerAddr (const char * *uri*)

Definition at line 761 of file xml_storage.c.

References _serverAddress, and _serverAddressLength.

Referenced by compare_uri(), getNodeByHref(), normalizeUri(), and obix_fcgi_handleRequest().

7.41.2.4 int xmldb_delete (const char * *href*)

Removes XML node from the storage.

Parameters:

href address of the node to be deleted.

Definition at line 625 of file xml_storage.c.

References _storage, getNodeByHref(), and log_warning.

Referenced by handlerSignUp(), obixWatch_deleteHelper(), and testDelete().

7.41.2.5 int xmldb_deleteMeta (IXML_Node * *attr*)

Definition at line 833 of file xml_storage.c.

References ixmlelement_getOwnerElement(), ixmlelement_getNode(), ixmlelement_convertToAttr(), and log_error.

Referenced by obixWatch_createWatchItem(), and obixWatchItem_free().

7.41.2.6 void xmldb_dispose ()

Stops work of the storage and releases all resources.

Definition at line 113 of file xml_storage.c.

References _serverAddress, and _storage.

Referenced by obix_server_shutdown().

7.41.2.7 char* xmldb_get (const char * href, int * slashFlag)

Retrieves XML node with specified URI from the storage.

Parameters:

href address of the XML node to be retrieved.

Returns:

XML data in plain text format or NULL on error.

Definition at line 372 of file xml_storage.c.

References _storage, and getNodeByHref().

Referenced by testDelete(), testSearch(), and testWriteToDatabase().

7.41.2.8 IXML_Element* xmldb_getDOM (const char * href, int * slashFlag)

Retrieves DOM structure of XML node with specified URI from the storage.

Parameters:

href address of the XML node to be retrieved.

Returns:

XML data as a DOM structure NULL on error.

Definition at line 367 of file xml_storage.c.

References _storage, getNodeByHref(), and ixmlelement_convertToElement().

Referenced by obix_server_invoke(), obix_server_read(), obixWatch_createWatchItem(), putDeviceReference(), testGenerateResponse(), xmldb_getObixSysObject(), and xmldb_updateDOM().

7.41.2.9 char* xmldb_getDump ()

Definition at line 705 of file xml_storage.c.

References _storage, and ixmldocument_getNode().

Referenced by obix_fcgidumpEnvironment().

7.41.2.10 char* xmldb_getFullUri (const char * *absUri*, int *slashFlag*)

1 - add slash -1 - remove slash 0 - do nothing

Parameters:

absUri

slashFlag

Returns:

Definition at line 723 of file xml_storage.c.

References _serverAddress, _serverAddressLength, log_error, and log_warning.

Referenced by normalizeUri().

7.41.2.11 IXML_Element* xmldb_getObixSysObject (const char * *objType*)

Definition at line 710 of file xml_storage.c.

References ixmlelement_cloneWithLog(), and xmldb_getDOM().

Referenced by obix_server_generateObixErrorMessage(), and obixWatch_create().

7.41.2.12 const char* xmldb_getServerAddress ()

Definition at line 767 of file xml_storage.c.

References _serverAddress.

7.41.2.13 int xmldb_getServerAddressLength ()

Definition at line 772 of file xml_storage.c.

References _serverAddressLength.

Referenced by handlerSignUp(), and obix_fcgi_handleRequest().

7.41.2.14 int xmldb_init (const char * *serverAddr*)

Initializes storage.

Should be executed only once on startup.

Parameters:

serverAddr of the server. Storage should know it in cases when requested URI contains full address.
If NULL is provided that address will be retrieved from the Lobby object.

Returns:

error code or 0 on success.

Definition at line 69 of file xml_storage.c.

References _serverAddress, _serverAddressLength, _storage, log_debug, log_error, OBIX_STORAGE_FILES, OBIX_STORAGE_FILES_COUNT, and xmldb_loadFile().

Referenced by obix_server_init(), and test_server().

7.41.2.15 int xmldb_loadFile (const char **filename*)

Loads xml file to the storage.

Parameters:

filename name of the xml file to load.

Returns:

error code or 0 on success.

Definition at line 645 of file xml_storage.c.

References config_getResFullPath(), FALSE, log_error, and xmldb_putHelper().

Referenced by test_server(), and xmldb_init().

7.41.2.16 void xmldb_printDump ()

Prints to standard log current contents of the storage.

Definition at line 700 of file xml_storage.c.

References _storage, ixmDocument_getNode(), and printXMLContents().

Referenced by test_server(), and testGenerateResponse().

7.41.2.17 int xmldb_put (const char **data*)

Returns the remaining slash flag of the last URI comparison operation.

URI comparison in database is done ignoring ending slash ('/'). It is done every time when some object is retrieved from the storage. Slash flag shows whether requested URI and URI found in storage contain ending slash.

Slash flag is set by following functions:

- [xmldb_get](#);
- [xmldb_getDOM](#);
- [xmldb_update](#);
- [xmldb_put](#). //
- [xmldb_compareUri](#).

Note:

The value of the flag can be changed also by other xmldb functions.

Returns:

- **0** if both URI had the same ending symbol;
- **1** if the object in the storage had ending slash but requested URI hadn't;
- **-1** if requested URI had ending slash, but the object hadn't. Compares two URI ignoring ending slash ('/'). This function also sets the slash flag.

See also:

[xmldb_getLastUriCompSlashFlag](#)

Parameters:

- uri1* first URI to compare (object's URI for the slash flag).
uri2 second URI to compare (request URI for the slash flag).

Returns:

TRUE is URI match, **FALSE** otherwise. Adds new XML node to the storage. The node should contain *href* attribute. If node with the same URI already exists in the storage then new node is not added.

Parameters:

- data* XML node represented in plain text format.

Returns:

error code or *0* on success.

Definition at line 555 of file `xml_storage.c`.

References **TRUE**, and [xmldb_putHelper\(\)](#).

Referenced by [obixWatch_create\(\)](#), and [testWriteToDatabase\(\)](#).

7.41.2.18 int xmldb_putDOM (IXML_Element * *data*)

Definition at line 525 of file `xml_storage.c`.

References **TRUE**, and [xmldb_putDOMHelper\(\)](#).

Referenced by [handlerSignUp\(\)](#).

7.41.2.19 IXML_Node* xmldb_putMeta (IXML_Element * *element*, const char * *name*, const char * *value*)

Definition at line 777 of file xml_storage.c.

References _storage, getMetaInfo(), ixmlelement_getNode(), log_error, OBIX_ATTR_VAL, and OBIX_META.

Referenced by obixWatch_createWatchItem().

7.41.2.20 int xmldb_updateDOM (IXML_Element * *input*, const char * *href*, IXML_Element ** *updatedNode*, int * *slashFlag*)

Updates XML node to the storage.

Only *val* attribute is updated and only for nodes which have *writable* attribute equal to *true*.

Parameters:

data XML node represented in plain text format. The node should contain *val* attribute. Other data is ignored.

href URI of the object to be updated.

updatedNode If not *NULL* is provided and update is successful than the address of the updated node will be written there.

Returns:

- **0** if value is successfully overwritten;
- **1** if request is processed but new value is the same;
- **-1** if request data is corrupted/wrong format;
- **-2** if object with specified href is not found;
- **-3** if object is not writable;
- **-4** if request failed because of internal server error.

Definition at line 560 of file xml_storage.c.

References ixmlelement_setAttributeWithLog(), log_warning, OBIX_ATTR_VAL, OBIX_ATTR_WRITABLE, XML_TRUE, and xmldb_getDOM().

Referenced by obix_server_write(), and testWriteToDatabase().

7.41.2.21 int xmldb_updateMeta (IXML_Node * *meta*, const char * *newValue*)

Definition at line 850 of file xml_storage.c.

Referenced by obixWatchItem_setUpdated().

7.41.3 Variable Documentation

7.41.3.1 const char* OBIX_META

Definition at line 42 of file xml_storage.c.

Referenced by getMetaInfo(), removeMetaInfo(), testGenerateResponse(), and xmldb_putMeta().

7.41.3.2 const char* OBIX_SYS_ERROR_STUB

Definition at line 39 of file xml_storage.c.

Referenced by obix_server_generateObixErrorMessage().

7.41.3.3 const char* OBIX_SYS_WATCH_OUT_STUB

Definition at line 40 of file xml_storage.c.

7.41.3.4 const char* OBIX_SYS_WATCH_STUB

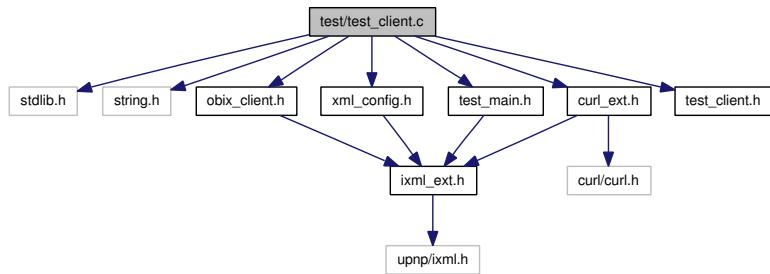
Definition at line 38 of file xml_storage.c.

Referenced by obixWatch_create().

7.42 test/test_client.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <obix_client.h>
#include <curl_ext.h>
#include <xml_config.h>
#include "test_main.h"
#include "test_client.h"
```

Include dependency graph for test_client.c:



Defines

- #define REQUEST_HTTP_GET 0
- #define REQUEST_HTTP_POST 2
- #define REQUEST_HTTP_PUT 1

Functions

- int test_client ()
- void test_client_byHands ()
- int testBatch ()
- int testConnectionAndDevices ()
- static int testCurlExt ()
- static int testCurlExtRequest (const char *testName, int requestType, CURL_EXT *handle, const char *uri, BOOL exists)
- static int testObixLoadConfigFile ()

7.42.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [test_client.c](#).

7.42.2 Define Documentation

7.42.2.1 #define REQUEST_HTTP_GET 0

Definition at line 36 of file [test_client.c](#).

Referenced by [testCurlExt\(\)](#), and [testCurlExtRequest\(\)](#).

7.42.2.2 #define REQUEST_HTTP_POST 2

Definition at line 38 of file test_client.c.

7.42.2.3 #define REQUEST_HTTP_PUT 1

Definition at line 37 of file test_client.c.

7.42.3 Function Documentation**7.42.3.1 int test_client ()**

Definition at line 369 of file test_client.c.

References testConnectionAndDevices(), and testCurlExt().

Referenced by runAutomaticTests().

7.42.3.2 void test_client_byHands ()

Definition at line 383 of file test_client.c.

References testConnectionAndDevices().

7.42.3.3 int testBatch ()

Definition at line 225 of file test_client.c.

References ixmlelement_getNode(), obix_batch_create(), obix_batch_free(), obix_batch_getResult(), obix_batch_read(), obix_batch_readValue(), obix_batch_send(), obix_batch_writeValue(), OBIX_SUCCESS, OBIX_T_INT, _oBIX_BatchResult::obj, _oBIX_BatchResult::status, and _oBIX_BatchResult::value.

Referenced by testConnectionAndDevices().

7.42.3.4 int testConnectionAndDevices ()

Definition at line 290 of file test_client.c.

References FALSE, obix_dispose(), obix_openConnection(), obix_registerDevice(), OBIX_SUCCESS, obix_unregisterDevice(), printTestResult(), testBatch(), testObixLoadConfigFile(), and TRUE.

Referenced by test_client(), and test_client_byHands().

7.42.3.5 static int testCurlExt () [static]

Definition at line 112 of file test_client.c.

References curl_ext_create(), curl_ext_init(), FALSE, _CURL_EXT::inputBuffer, _CURL_EXT::outputBuffer, printTestResult(), REQUEST_HTTP_GET, REQUEST_HTTP_POST, REQUEST_HTTP_PUT, testCurlExtRequest(), and TRUE.

Referenced by test_client().

7.42.3.6 static int testCurlExtRequest (const char * testName, int requestType, CURL_EXT * handle, const char * uri, BOOL exists) [static]

Definition at line 40 of file test_client.c.

References curl_ext_get(), curl_ext_post(), curl_ext_put(), _CURL_EXT::errorBuffer, FALSE, _CURL_EXT::inputBuffer, printTestResult(), REQUEST_HTTP_GET, REQUEST_HTTP_POST, REQUEST_HTTP_PUT, and TRUE.

Referenced by testCurlExt().

7.42.3.7 static int testObixLoadConfigFile () [static]

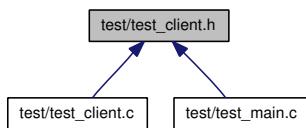
Definition at line 200 of file test_client.c.

References FALSE, obix_loadConfigFile(), OBIX_SUCCESS, printTestResult(), and TRUE.

Referenced by testConnectionAndDevices().

7.43 test/test_client.h File Reference

This graph shows which files directly or indirectly include this file:

**Functions**

- int [test_client \(\)](#)
- void [test_client_byHands \(\)](#)

7.43.1 Detailed Description**Todo**

[add description here](#)

Author:

Andrey Litvinov

Version:

1.0

Definition in file [test_client.h](#).

7.43.2 Function Documentation**7.43.2.1 int test_client ()**

Definition at line 369 of file [test_client.c](#).

References [testConnectionAndDevices\(\)](#), and [testCurlExt\(\)](#).

Referenced by [runAutomaticTests\(\)](#).

7.43.2.2 void test_client_byHands ()

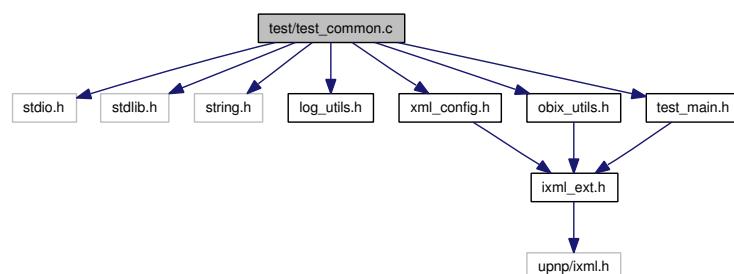
Definition at line 383 of file [test_client.c](#).

References [testConnectionAndDevices\(\)](#).

7.44 test/test_common.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <log_utils.h>
#include <xml_config.h>
#include <obix_utils.h>
#include "test_main.h"
```

Include dependency graph for [test_common.c](#):



Functions

- int [test_common\(\)](#)
- void [test_common_byHands\(\)](#)
- void [testLog\(\)](#)
- int [testObix_reltimes_fromLong\(\)](#)
- int [testObix_reltimes_parse\(\)](#)
- int [testObix_reltimes_parseToLong\(const char *reltime, int retVal, long value\)](#)
- int [testObixUtils\(\)](#)

7.44.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [test_common.c](#).

7.44.2 Function Documentation

7.44.2.1 int test_common()

Definition at line 193 of file [test_common.c](#).

References [testObixUtils\(\)](#).

Referenced by [runAutomaticTests\(\)](#).

7.44.2.2 void test_common_byHands()

Definition at line 202 of file [test_common.c](#).

References [testLog\(\)](#).

7.44.2.3 void testLog()

Definition at line 37 of file [test_common.c](#).

References [config_finishInit\(\)](#), [config_loadFile\(\)](#), [config_log\(\)](#), [log_debug](#), [log_error](#), [log_usePrintf\(\)](#), [log_warning](#), and [TRUE](#).

Referenced by [test_common_byHands\(\)](#).

7.44.2.4 int testObix_relttime_fromLong ()

Definition at line 130 of file test_common.c.

References FALSE, obix_relttime_fromLong(), printTestResult(), RELTIME_DAY, RELTIME_HOUR, RELTIME_SEC, RELTIME_YEAR, and TRUE.

Referenced by testObixUtils().

7.44.2.5 int testObix_relttime_parse ()

Definition at line 72 of file test_common.c.

References FALSE, printTestResult(), testObix_relttime_parseToLong(), and TRUE.

Referenced by testObixUtils().

7.44.2.6 int testObix_relttime_parseToLong (const char * *reltime*, int *retVal*, long *value*)

Definition at line 52 of file test_common.c.

References obix_relttime_parseToLong().

Referenced by testObix_relttime_parse().

7.44.2.7 int testObixUtils ()

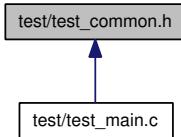
Definition at line 183 of file test_common.c.

References testObix_relttime_fromLong(), and testObix_relttime_parse().

Referenced by test_common().

7.45 test/test_common.h File Reference

This graph shows which files directly or indirectly include this file:

**Functions**

- int [test_common \(\)](#)
- void [test_common_byHands \(\)](#)

7.45.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [test_common.h](#).

7.45.2 Function Documentation

7.45.2.1 int test_common ()

Definition at line 193 of file test_common.c.

References testObixUtils().

Referenced by runAutomaticTests().

7.45.2.2 void test_common_byHands ()

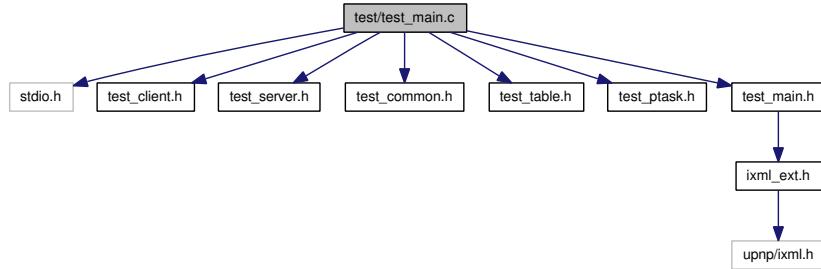
Definition at line 202 of file test_common.c.

References testLog().

7.46 test/test_main.c File Reference

```
#include <stdio.h>
#include "test_client.h"
#include "test_server.h"
#include "test_common.h"
#include "test_table.h"
#include "test_ptask.h"
#include "test_main.h"
```

Include dependency graph for test_main.c:



Functions

- int **main** (int argc, char **argv)
- void **printTestResult** (const char *name, **BOOL** successfull)
- static int **runAutomaticTests** (const char *resFolder)
- static void **runManualTests** ()

7.46.1 Function Documentation

7.46.1.1 int main (int *argc*, char ***argv*)

Definition at line 60 of file test_main.c.

References runAutomaticTests(), and runManualTests().

7.46.1.2 void printTestResult (const char * *name*, **BOOL** *successfull*)

Definition at line 52 of file test_main.c.

Referenced by test_table(), testConnectionAndDevices(), testCurlExt(), testCurlExtRequest(), testDelete(), testGenerateResponse(), testGetClosestTask(), testObix_relttime_fromLong(), testObix_relttime_parse(), testObixLoadConfigFile(), testPeriodicTask(), testPtakReschedule(), testPutHandler(), testResponse_setRightUri(), testSearch(), testSignUpHelper(), testTableGet(), testWatch(), testWatchPollChanges(), testWatchRemove(), and testWriteToDatabase().

7.46.1.3 static int runAutomaticTests (const char * *resFolder*) [static]

Definition at line 30 of file test_main.c.

References test_client(), test_common(), test_ptask(), test_server(), and test_table().

Referenced by main().

7.46.1.4 static void runManualTests () [static]

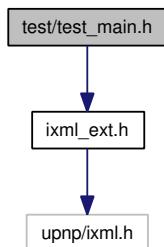
Definition at line 45 of file test_main.c.

Referenced by main().

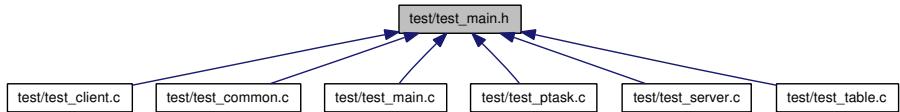
7.47 test/test_main.h File Reference

```
#include <ixml_ext.h>
```

Include dependency graph for test_main.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [printTestResult](#) (const char *name, **BOOL** successfull)

7.47.1 Function Documentation

7.47.1.1 void printTestResult (const char * *name*, **BOOL** *successfull*)

Definition at line 52 of file test_main.c.

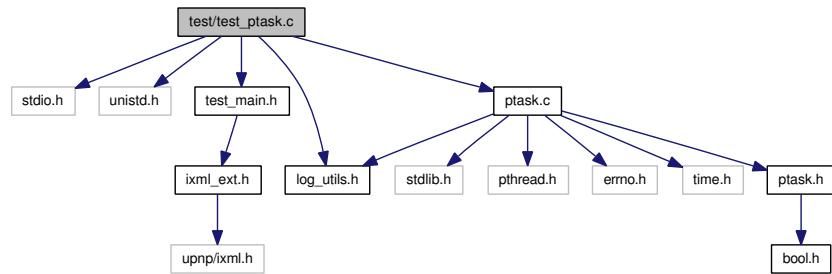
Referenced by `test_table()`, `testConnectionAndDevices()`, `testCurlExt()`, `testCurlExtRequest()`, `testDelete()`, `testGenerateResponse()`, `testGetClosestTask()`, `testObix_reltimes_fromLong()`, `testObix_reltimes_parse()`, `testObixLoadConfigFile()`, `testPeriodicTask()`, `testPtakReschedule()`, `testPutHandler()`, `testResponse_setRightUri()`, `testSearch()`, `testSignUpHelper()`, `testTableGet()`, `testWatch()`, `testWatchPollChanges()`, `testWatchRemove()`, and `testWriteToDatabase()`.

7.48 test/test_ptask.c File Reference

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

```
#include <unistd.h>
#include "test_main.h"
#include <log_utils.h>
#include <ptask.c>
```

Include dependency graph for test_ptask.c:



Functions

- void [taskStopThread](#) (void *arg)
- int [test_ptask](#) ()
- void [test_ptask_byHands](#) ()
- int [testGetClosestTask](#) (Task_Thread *thread, char *testName, int checkIds[], int iterations)
- int [testPeriodicTask](#) (Task_Thread *thread)
- int [testPtaskReschedule](#) (Task_Thread *thread, const char *testName, long period, **BOOL** add, **BOOL** shouldPass)
- void [testTask](#) (void *arg)

7.48.1 Function Documentation

7.48.1.1 void [taskStopThread](#) (void * *arg*)

Definition at line 65 of file test_ptask.c.

References ptask_dispose(), and TRUE.

Referenced by test_ptask_byHands().

7.48.1.2 int [test_ptask](#) ()

Definition at line 243 of file test_ptask.c.

References FALSE, _Task_Thread::id_gen, periodicTask_deleteRecursive(), _Task_Thread::taskExecuted, _Task_Thread::taskList, _Task_Thread::taskListMutex, _Task_Thread::taskListUpdated, testPeriodicTask(), testPtaskReschedule(), and TRUE.

Referenced by runAutomaticTests().

7.48.1.3 void test_ptask_byHands ()

Definition at line 72 of file test_ptask.c.

References EXECUTE_INDEFINITE, ptask_init(), ptask_schedule(), taskStopThread(), and testTask().

7.48.1.4 int testGetClosestTask (Task_Thread * *thread*, char * *testName*, int *checkIds*[], int *iterations*)

Definition at line 41 of file test_ptask.c.

References FALSE, _Periodic_Task::id, periodicTask_execute(), periodicTask_getClosest(), printTestResult(), _Task_Thread::taskListMutex, and TRUE.

Referenced by testPeriodicTask().

7.48.1.5 int testPeriodicTask (Task_Thread * *thread*)

Definition at line 170 of file test_ptask.c.

References EXECUTE_INDEFINITE, FALSE, _Periodic_Task::id, _Periodic_Task::next, printTestResult(), ptask_cancel(), ptask_schedule(), _Task_Thread::taskList, testGetClosestTask(), testTask(), and TRUE.

Referenced by test_ptask().

7.48.1.6 int testPtaskReschedule (Task_Thread * *thread*, const char * *testName*, long *period*, BOOL *add*, BOOL *shouldPass*)

Definition at line 90 of file test_ptask.c.

References EXECUTE_INDEFINITE, FALSE, _Periodic_Task::nextScheduledTime, periodicTask_get(), printTestResult(), ptask_cancel(), ptask_reschedule(), ptask_schedule(), testTask(), timespec_add(), timespec_cmp(), timespec_copy(), and TRUE.

Referenced by test_ptask().

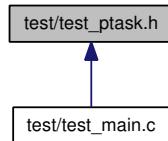
7.48.1.7 void testTask (void * *arg*)

Definition at line 35 of file test_ptask.c.

Referenced by test_ptask_byHands(), testPeriodicTask(), and testPtaskReschedule().

7.49 test/test_ptask.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- int [test_ptask\(\)](#)
- void [test_ptask_byHands\(\)](#)

7.49.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [test_ptask.h](#).

7.49.2 Function Documentation

7.49.2.1 int test_ptask()

Definition at line 243 of file [test_ptask.c](#).

References FALSE, _Task_Thread::id_gen, periodicTask_deleteRecursive(), _Task_Thread::taskExecuted, _Task_Thread::taskList, _Task_Thread::taskListMutex, _Task_Thread::taskListUpdated, testPeriodicTask(), testPtaskReschedule(), and TRUE.

Referenced by runAutomaticTests().

7.49.2.2 void test_ptask_byHands()

Definition at line 72 of file [test_ptask.c](#).

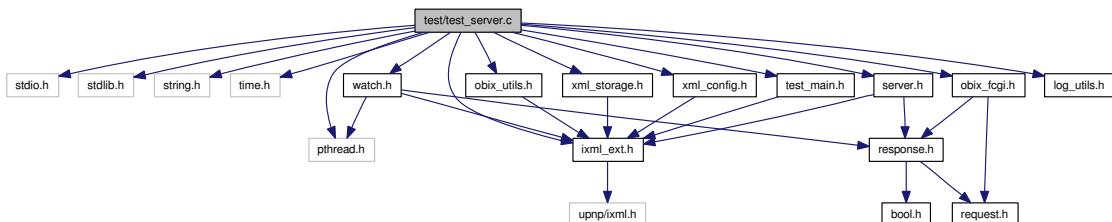
References EXECUTE_INDEFINITE, ptask_init(), ptask_schedule(), taskStopThread(), and testTask().

7.50 test/test_server.c File Reference

That's a temporary file to test various pieces of functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <pthread.h>
#include <obix_utils.h>
#include <xml_storage.h>
#include <log_utils.h>
#include <xml_config.h>
#include <ixml_ext.h>
#include <server.h>
#include <watch.h>
#include <obix_fcgi.h>
#include "test_main.h"
```

Include dependency graph for test_server.c:



Functions

- int `checkResponse` (`Response` *`response`, `BOOL` `containsError`)
- void `dummyResponseListener` (`Response` *`response`)
- int `findInResponse` (`Response` *`response`, const char *`checkString`, `BOOL` `exists`)
- static `BOOL` `isResponseSent` ()
- int `obix_client_testListener` (int `connectionId`, int `deviceId`, const char *`paramUri`, const char *`newValue`)
- void `printResponse` (`Response` *`response`)
- int `test_server` (char *`resFolder`)
- int `testDelete` (const char *`testName`, const char *`href`, `BOOL` `exists`)
- int `testGenerateResponse` (const char *`testName`, const char *`uri`, const char *`newUrl`)
- `Response` * `testPostHandler` (const char *`uri`, `IXML_Document` *`input`)
- int `testPutHandler` (const char *`testName`, const char *`uri`, const char *`data`)
- int `testResponse_setRightUri` (const char *`testName`, const char *`requestUri`, int `slashFlag`, const char *`rightUri`)
- int `testSearch` (const char *`testName`, const char *`href`, const char *`checkStr`, `BOOL` `exists`)

- int `testSignUp()`
- int `testSignUpHelper(const char *testName, const char *inputData, const char *checkUri, const char *checkString, BOOL shouldPass)`
- int `testWatch()`
- int `testWatchPollChanges(const char *testName, const char *uri, char *checkStrings[], int checkSize, BOOL exists, BOOL waitResponse)`
- int `testWatchRemove()`
- int `testWriteToDatabase(const char *testName, BOOL writeNew, const char *newData, const char *href, const char *checkString, BOOL shouldPass)`
- static `Response *waitForResponse()`

Variables

- `Response * _lastResponse`
- `BOOL _responseIsSent = FALSE`
- `pthread_mutex_t _responseMutex = PTHREAD_MUTEX_INITIALIZER`
- `pthread_cond_t _responseReceived = PTHREAD_COND_INITIALIZER`

7.50.1 Detailed Description

That's a temporary file to test various pieces of functionality.

Definition in file [test_server.c](#).

7.50.2 Function Documentation

7.50.2.1 int `checkResponse(Response * response, BOOL containsError)`

Definition at line 583 of file [test_server.c](#).

References `Response::body`, `isResponseSent()`, `Response::next`, `printResponse()`, and `Response::request`.

Referenced by `testPutHandler()`, `testSignUpHelper()`, `testWatch()`, `testWatchPollChanges()`, and `testWatchRemove()`.

7.50.2.2 void `dummyResponseListener(Response * response)`

Definition at line 45 of file [test_server.c](#).

References `_responseIsSent`, `_responseMutex`, `_responseReceived`, and `TRUE`.

Referenced by `testSignUpHelper()`, `testWatch()`, `testWatchPollChanges()`, and `testWatchRemove()`.

7.50.2.3 int `findInResponse(Response * response, const char * checkString, BOOL exists)`

Definition at line 627 of file [test_server.c](#).

References `Response::body`, and `Response::next`.

Referenced by `testWatchPollChanges()`, and `testWatchRemove()`.

7.50.2.4 static BOOL isResponseSent () [static]

Definition at line 54 of file test_server.c.

References _responseIsSent, FALSE, and TRUE.

Referenced by checkResponse().

7.50.2.5 int obix_client_testListener (int *connectionId*, int *deviceId*, const char * *paramUri*, const char * *newValue*)

Definition at line 452 of file test_server.c.

7.50.2.6 void printResponse (Response * *response*)

Definition at line 571 of file test_server.c.

References Response::body, and Response::next.

Referenced by checkResponse().

7.50.2.7 int test_server (char * *resFolder*)

Definition at line 1036 of file test_server.c.

References config_setResourceDir(), FALSE, obixWatch_init(), testDelete(), testGenerateResponse(), testResponse_setRightUri(), testSearch(), testSignUp(), testWatch(), testWriteToDatabase(), TRUE, xmldb_init(), xmldb_loadFile(), and xmldb_printDump().

Referenced by runAutomaticTests().

7.50.2.8 int testDelete (const char * *testName*, const char * *href*, BOOL *exists*)

Definition at line 200 of file test_server.c.

References FALSE, printTestResult(), TRUE, xmldb_delete(), and xmldb_get().

Referenced by test_server().

7.50.2.9 int testGenerateResponse (const char * *testName*, const char * *uri*, const char * *newUrl*)

Definition at line 242 of file test_server.c.

References Response::body, FALSE, OBIX_META, obix_server_generateResponse(), obixResponse_create(), obixResponse_free(), printTestResult(), testSearch(), TRUE, xmldb_getDOM(), and xmldb_printDump().

Referenced by test_server().

7.50.2.10 Response* testPostHandler (const char * *uri*, IXML_Document * *input*)

Definition at line 321 of file test_server.c.

7.50.2.11 int testPutHandler (const char * *testName*, const char * *uri*, const char * *data*)

Definition at line 756 of file test_server.c.

References checkResponse(), FALSE, obix_server_handlePUT(), obixResponse_create(), obixResponse_free(), printTestResult(), and TRUE.

Referenced by testWatch().

7.50.2.12 int testResponse_setRightUri (const char * *testName*, const char * *requestUri*, int *slashFlag*, const char * *rightUri*)

Definition at line 1012 of file test_server.c.

References FALSE, obixResponse_create(), obixResponse_free(), obixResponse_setRightUri(), printTestResult(), TRUE, and Response::uri.

Referenced by test_server().

7.50.2.13 int testSearch (const char * *testName*, const char * *href*, const char * *checkStr*, BOOL *exists*)

Definition at line 79 of file test_server.c.

References FALSE, printTestResult(), TRUE, and xmldb_get().

Referenced by test_server(), testGenerateResponse(), testSignUpHelper(), testWatch(), and testWatchRemove().

7.50.2.14 int testSignUp ()

Definition at line 978 of file test_server.c.

References FALSE, testSignUpHelper(), and TRUE.

Referenced by test_server().

7.50.2.15 int testSignUpHelper (const char * *testName*, const char * *inputData*, const char * *checkUri*, const char * *checkString*, BOOL *shouldPass*)

Definition at line 948 of file test_server.c.

References checkResponse(), dummyResponseListener(), FALSE, obix_server_handlePOST(), obixResponse_create(), obixResponse_free(), obixResponse_setListener(), printTestResult(), testSearch(), and TRUE.

Referenced by testSignUp().

7.50.2.16 int testWatch ()

Definition at line 770 of file test_server.c.

References checkResponse(), dummyResponseListener(), FALSE, obix_server_handlePOST(), obix_server_handlePUT(), obixResponse_create(), obixResponse_free(), obixResponse_setListener(), printTestResult(), testPutHandler(), testSearch(), testWatchPollChanges(), testWatchRemove(), and TRUE.

Referenced by test_server().

7.50.2.17 int testWatchPollChanges (const char * *testName*, const char * *uri*, char * *checkStrings*[], int *checkSize*, BOOL *exists*, BOOL *waitResponse*)

Definition at line 663 of file test_server.c.

References checkResponse(), dummyResponseListener(), FALSE, findInResponse(), obix_server_handlePOST(), obixResponse_create(), obixResponse_free(), obixResponse_setListener(), printTestResult(), TRUE, and waitForResponse().

Referenced by testWatch().

7.50.2.18 int testWatchRemove ()

Definition at line 700 of file test_server.c.

References checkResponse(), dummyResponseListener(), FALSE, findInResponse(), obix_server_handlePOST(), obixResponse_create(), obixResponse_free(), obixResponse_setListener(), printTestResult(), testSearch(), and TRUE.

Referenced by testWatch().

7.50.2.19 int testWriteToDatabase (const char * *testName*, BOOL *writeNew*, const char * *newData*, const char * *href*, const char * *checkString*, BOOL *shouldPass*)

Definition at line 116 of file test_server.c.

References FALSE, ixmlelement_freeOwnerDocument(), ixmlelement_parseBuffer(), printTestResult(), TRUE, xmldb_get(), xmldb_put(), and xmldb_updateDOM().

Referenced by test_server().

7.50.2.20 static Response* waitForResponse () [static]

Definition at line 67 of file test_server.c.

References _responseMutex, and _responseReceived.

Referenced by testWatchPollChanges().

7.50.3 Variable Documentation

7.50.3.1 Response* _lastResponse

Definition at line 41 of file test_server.c.

7.50.3.2 BOOL _responseIsSent = FALSE

Definition at line 40 of file test_server.c.

Referenced by dummyResponseListener(), and isResponseSent().

7.50.3.3 pthread_mutex_t _responseMutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 42 of file test_server.c.

Referenced by dummyResponseListener(), and waitForResponse().

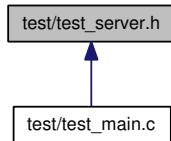
7.50.3.4 pthread_cond_t _responseReceived = PTHREAD_COND_INITIALIZER

Definition at line 43 of file test_server.c.

Referenced by dummyResponseListener(), and waitForResponse().

7.51 test/test_server.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- int [test_server](#) (const char *resFolder)

7.51.1 Detailed Description

[Todo](#)

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [test_server.h](#).

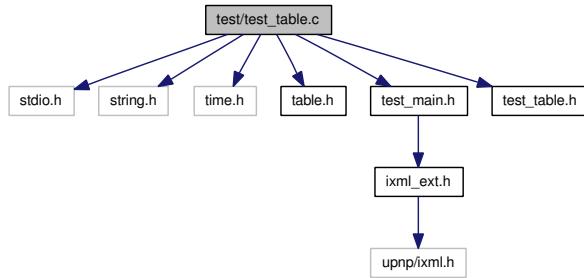
7.51.2 Function Documentation

7.51.2.1 int [test_server](#) (const char * *resFolder*)

7.52 test/test_table.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <table.h>
#include "test_main.h"
#include "test_table.h"
```

Include dependency graph for test_table.c:



Functions

- int `test_table ()`
- static int `testTableGet (Table *table, const char *key, const char *value)`

7.52.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [test_table.c](#).

7.52.2 Function Documentation

7.52.2.1 int `test_table ()`

Definition at line 57 of file [test_table.c](#).

References FALSE, printTestResult(), table_create(), table_free(), table_get(), table_put(), table_remove(), testTableGet(), and TRUE.

Referenced by runAutomaticTests().

7.52.2.2 static int `testTableGet (Table *table, const char *key, const char *value) [static]`

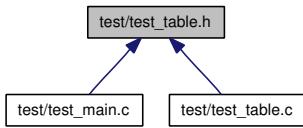
Definition at line 35 of file [test_table.c](#).

References FALSE, printTestResult(), and table_get().

Referenced by `test_table()`.

7.53 test/test_table.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- int [test_table \(\)](#)

7.53.1 Detailed Description

Todo

add description here

Author:

Andrey Litvinov

Version:

1.0

Definition in file [test_table.h](#).

7.53.2 Function Documentation

7.53.2.1 int [test_table \(\)](#)

Definition at line 57 of file [test_table.c](#).

References FALSE, printTestResult(), table_create(), table_free(), table_get(), table_put(), table_remove(), testTableGet(), and TRUE.

Referenced by runAutomaticTests().

Index

_CURL_EXT, 12
 curl, 12
 errorBuffer, 13
 inputBuffer, 13
 inputBufferFree, 13
 inputBufferSize, 13
 outputBuffer, 13
 outputPos, 13
 outputSize, 14
_Comm_Stack, 7
 closeConnection, 7
 freeConnection, 7
 initConnection, 7
 openConnection, 7
 read, 8
 readValue, 8
 registerDevice, 8
 registerListener, 8
 sendBatch, 8
 unregisterDevice, 8
 unregisterListener, 8
 writeValue, 9
_Connection, 9
 comm, 11
 deviceCount, 11
 devices, 11
 id, 11
 isConnected, 11
 maxDevices, 11
 maxListeners, 12
 type, 12
_Device, 14
 id, 15
 listenerCount, 15
 listeners, 15
_Http_Connection, 15
 batchUri, 17
 c, 17
 lobbyUri, 17
 pollInterval, 17
 pollWaitMax, 17
 pollWaitMin, 17
 serverUri, 18
 serverUriLength, 18
 signUpUri, 18
 watchAddUri, 18
 watchDeleteUri, 18
 watchLease, 18
 watchMakeUri, 19
 watchMutex, 19
_Http_Device, 20
 d, 21
 uri, 21
 uriLength, 21
_Listener, 21
 callback, 22
 connectionId, 22
 deviceId, 22
 id, 22
 paramUri, 22
_Periodic_Task, 28
 arg, 29
 executeTimes, 29
 id, 29
 isCancelled, 29
 isExecuting, 29
 next, 29
 nextScheduledTime, 30
 period, 30
 prev, 30
 task, 30
_Request, 31
 canWait, 31
 id, 31
 next, 31
 r, 32
_Table, 32
 count, 32
 keys, 32
 size, 32
 values, 33
_Target, 33
 changed, 34
 id, 34
 new, 34
 removeTask, 34
 uri, 34
 x, 34
 y, 35
_Task_Thread, 36
 id_gen, 37
 taskExecuted, 37
 taskList, 37
 taskListMutex, 37
 taskListUpdated, 37
 thread, 37

_TreeNode, 38
 child, 38
 name, 38
 neighbor, 38
_connectionCount
 obix_client.c, 93
_connections
 obix_client.c, 93
_curl_handle
 obix_http.c, 130
_curl_watch_handle
 obix_http.c, 130
_defaultInputBufferSize
 curl_ext.c, 68
_deviceData
 sensor_floor_driver.c, 62
_deviceId
 sensor_floor_driver.c, 62
_header
 curl_ext.c, 68
_initialized
 obix_http.c, 130
_lastResponse
 test_server.c, 325
_log_level
 log_utils.c, 164
_oBIX_Batch, 22
 command, 24
 commandCounter, 24
 connection, 24
 result, 24
_oBIX_BatchCmd, 25
 dataType, 26
 device, 26
 id, 26
 newValue, 26
 next, 26
 paramUri, 26
 type, 26
_oBIX_BatchResult, 27
 obj, 27
 status, 27
 value, 28
_requestIds
 obix_fcgi.c, 238
_requestList
 obix_fcgi.c, 238
_requestListMutex
 obix_fcgi.c, 238
_requestListNew
 obix_fcgi.c, 239
_requestMaxCount
 obix_fcgi.c, 239
_requestsInUse
 obix_fcgi.c, 239
 test_server.c, 325
_responseListener
 response.c, 256
_responseMutex
 test_server.c, 325
_responseReceived
 test_server.c, 325
_serverAddress
 xml_storage.c, 297
_serverAddressLength
 xml_storage.c, 297
_storage
 xml_storage.c, 297
_targetMutex
 sensor_floor_driver.c, 62
_targets
 sensor_floor_driver.c, 62
_targetsCount
 sensor_floor_driver.c, 62
_taskThread
 example_timer.c, 51
 sensor_floor_driver.c, 63
_testData
 sensor_floor_driver.c, 63
_threadLease
 watch.c, 277
_threadLongPoll
 watch.c, 277
_time
 example_timer.c, 51
_time_mutex
 example_timer.c, 51
_timerTaskId
 example_timer.c, 52
_tree
 doctree.c, 232
_use_syslog
 log_utils.c, 164
_watchThread
 obix_http.c, 130
_watches
 watch.c, 277
_watchesCount
 watch.c, 277
adapters/example_timer.c, 48
adapters/sensor_floor_driver.c, 52
addListener
 obix_http.c, 121
addResponsePart
 post_handler.c, 245
addWatchItem

obix_http.c, 121
arg
 _Periodic_Task, 29

BATCH_OUT_POSTFIX
 post_handler.c, 245
BATCH_OUT_PREFIX
 post_handler.c, 245
batchUri
 _Http_Connection, 17
body
 Response, 47
BOOL
 bool.h, 144
bool.h
 BOOL, 144
 FALSE, 143
 TRUE, 143
BOOL_H_
 ixml_ext.h, 154

c
 _Http_Connection, 17
callback
 _Listener, 22
canWait
 _Request, 31
 Response, 47
changed
 _Target, 34
checkEventsTask
 sensor_floor_driver.c, 57
checkNode
 xml_storage.c, 288
checkResponse
 test_server.c, 321
checkResponseDoc
 obix_http.c, 122
checkResponseElement
 obix_http.c, 122
checkTargets
 sensor_floor_driver.c, 57
child
 _TreeNode, 38
client/curl_ext.c, 63
client/curl_ext.h, 69
client/obix_batch.c, 72
client/obix_batch.h, 79
client/obix_client.c, 80
client/obix_client.h, 94
client/obix_comm.h, 111
client/obix_http.c, 115
client/obix_http.h, 132
client/table.c, 138

client/table.h, 140
closeConnection
 _Comm_Stack, 7
comm
 _Connection, 11
comm_closeConnection
 obix_comm.h, 112
comm_freeConnection
 obix_comm.h, 112
comm_initConnection
 obix_comm.h, 113
comm_openConnection
 obix_comm.h, 113
comm_read
 obix_comm.h, 113
comm_readValue
 obix_comm.h, 113
comm_registerDevice
 obix_comm.h, 113
comm_registerListener
 obix_comm.h, 113
comm_sendBatch
 obix_comm.h, 113
Comm_Stack
 obix_comm.h, 113
comm_unregisterDevice
 obix_comm.h, 114
comm_unregisterListener
 obix_comm.h, 114
comm_writeValue
 obix_comm.h, 114
command
 _oBIX_Batch, 24
commandCounter
 _oBIX_Batch, 24
common/bool.h, 142
common/ixml_ext.c, 144
common/ixml_ext.h, 152
common/log_utils.c, 161
common/log_utils.h, 165
common/obix_utils.c, 171
common/obix_utils.h, 183
common/ptask.c, 195
common/ptask.h, 204
common/xml_config.c, 210
common/xml_config.h, 220
compare_uri
 xml_storage.c, 288
completeWatchPollResponse
 post_handler.c, 246
CONFIG_FILE
 obix_fcgi.c, 239
config_finishInit
 xml_config.c, 212

xml_config.h, 222
config_getChildTag
 xml_config.c, 212
 xml_config.h, 223
config_getChildTagValue
 xml_config.c, 213
 xml_config.h, 223
config_getResFullPath
 xml_config.c, 213
 xml_config.h, 223
config_getTagAttrBoolValue
 xml_config.c, 214
 xml_config.h, 224
config_getTagAttributeValue
 xml_config.c, 214
 xml_config.h, 224
config_getTagAttrIntValue
 xml_config.c, 215
 xml_config.h, 225
config_getTagAttrLongValue
 xml_config.c, 215
 xml_config.h, 225
config_loadFile
 xml_config.c, 216
 xml_config.h, 226
config_log
 xml_config.c, 216
 xml_config.h, 226
config_setResourceDir
 xml_config.c, 217
 xml_config.h, 227
Connection
 obix_comm.h, 114
connection
 _oBIX_Batch, 24
connection_create
 obix_client.c, 83
connection_free
 obix_client.c, 83
connection_get
 obix_client.c, 83
 obix_comm.h, 115
CONNECTION_ID
 example_timer.c, 49
Connection_Type
 obix_comm.h, 114
connectionId
 _listener, 22
count
 Table, 32
createWatch
 obix_http.c, 122
CT_CONFIG
 xml_config.c, 217
 xml_config.h, 227
CT_CONNECTION
 obix_client.c, 93
CT_HOLD_REQUEST_MAX
 obix_fcgi.c, 239
CT_LOG
 xml_config.c, 217
 xml_config.h, 227
CT_LOG_LEVEL
 xml_config.c, 217
 xml_config.h, 227
CT_LOG_USE_SYSLOG
 xml_config.c, 218
 xml_config.h, 228
CT_LONG_POLL
 obix_http.c, 130
CT_LONG_POLL_MAX
 obix_http.c, 130
CT_LONG_POLL_MIN
 obix_http.c, 131
CT_MAX_DEVICES
 obix_client.c, 93
CT_MAX_LISTENERS
 obix_client.c, 93
CT_POLL_INTERVAL
 obix_http.c, 131
CT_SERVER_ADDRESS
 obix_http.c, 131
 server.c, 265
CT_WATCHLEASE
 obix_http.c, 131
CTA_CONNECTION_ID
 obix_client.c, 93
CTA_CONNECTION_TYPE
 obix_client.c, 93
CTA_LOBBY
 obix_http.c, 131
CTA_LOG_FACILITY
 xml_config.c, 218
 xml_config.h, 228
CTA_VALUE
 xml_config.c, 218
 xml_config.h, 228
CTAV_CONNECTION_TYPE_HTTP
 obix_client.c, 93
CTAV_LOG_FACILITY_DAEMON
 xml_config.c, 218
 xml_config.h, 228
CTAV_LOG_FACILITY_LOCAL0
 xml_config.c, 218
 xml_config.h, 228
CTAV_LOG_FACILITY_USER
 xml_config.c, 218
 xml_config.h, 229

CTAV_LOG_LEVEL_DEBUG
 xml_config.c, 219
 xml_config.h, 229
CTAV_LOG_LEVEL_ERROR
 xml_config.c, 219
 xml_config.h, 229
CTAV_LOG_LEVEL_NO
 xml_config.c, 219
 xml_config.h, 229
CTAV_LOG_LEVEL_WARNING
 xml_config.c, 219
 xml_config.h, 229
curl
 _CURL_EXT, 12
CURL_EXT
 curl_ext.h, 70
curl_ext.c
 _defaultInputBufferSize, 68
 _header, 68
 curl_ext_allocateMemory, 65
 curl_ext_create, 65
 curl_ext_dispose, 65
 curl_ext_free, 66
 curl_ext_freeMemory, 66
 curl_ext_get, 66
 curl_ext_getDOM, 66
 curl_ext_init, 66
 curl_ext_post, 66
 curl_ext_postDOM, 67
 curl_ext_put, 67
 curl_ext_putDOM, 67
DEF_INPUT_BUFFER_SIZE, 65
inputWriter, 67
outputReader, 67
parseXmlInput, 68
REQUEST_HTTP_POST, 65
REQUEST_HTTP_PUT, 65
sendRequest, 68
curl_ext.h
 CURL_EXT, 70
 curl_ext_create, 70
 curl_ext_dispose, 70
 curl_ext_free, 70
 curl_ext_get, 70
 curl_ext_getDOM, 71
 curl_ext_init, 71
 curl_ext_post, 71
 curl_ext_postDOM, 71
 curl_ext_put, 71
 curl_ext_putDOM, 72
curl_ext_allocateMemory
 curl_ext.c, 65
curl_ext_create
 curl_ext.c, 65
curl_ext.h, 70
curl_ext_dispose
 curl_ext.c, 65
 curl_ext.h, 70
curl_ext_free
 curl_ext.c, 66
 curl_ext.h, 70
curl_ext_freeMemory
 curl_ext.c, 66
curl_ext_get
 curl_ext.c, 66
 curl_ext.h, 70
curl_ext_getDOM
 curl_ext.c, 66
 curl_ext.h, 71
curl_ext_init
 curl_ext.c, 66
 curl_ext.h, 71
curl_ext_post
 curl_ext.c, 66
 curl_ext.h, 71
curl_ext_postDOM
 curl_ext.c, 67
 curl_ext.h, 71
curl_ext_put
 curl_ext.c, 67
 curl_ext.h, 71
curl_ext_putDOM
 curl_ext.c, 67
 curl_ext.h, 72
d
 _Http_Device, 21
dataType
 _oBIX_BatchCmd, 26
DEF_INPUT_BUFFER_SIZE
 curl_ext.c, 65
DEFAULT_MAX_DEVICES
 obix_client.c, 83
DEFAULT_MAX_LISTENERS
 obix_client.c, 83
DEFAULT_POLLING_INTERVAL
 obix_http.c, 118
DEFAULT_WATCHLEASE_PADDING
 obix_http.c, 118
Device
 obix_comm.h, 114
device
 _oBIX_BatchCmd, 26
DEVICE_DATA
 example_timer.c, 52
device_free
 obix_client.c, 84
device_get

obix_client.c, 84
obix_comm.h, 115
DEVICE_LIST_URI
 post_handler.c, 245
device_register
 obix_client.c, 84
device_unregister
 obix_client.c, 84
device_unregisterAllListeners
 obix_client.c, 84
DEVICE_URI_PREFIX
 xml_storage.c, 297
DEVICE_URI_PREFIX_LENGTH
 xml_storage.c, 297
deviceCount
 _Connection, 11
deviceId
 _Listener, 22
devices
 _Connection, 11
doc
 oBIX_Watch_Item, 43
doctree.c
 _tree, 232
 doctree_findNode, 231
 doctree_get, 231
 doctree_put, 231
 TreeNode, 231
doctree.h
 doctree_get, 233
 doctree_put, 233
doctree_findNode
 doctree.c, 231
doctree_get
 doctree.c, 231
 doctree.h, 233
doctree_put
 doctree.c, 231
 doctree.h, 233
doxygen/cot_main.h, 230
doxygen/libcot_desc.h, 230
dummyResponseListener
 test_server.c, 321

error
 Response, 47
ERROR_STATIC
 obix_fcgi.c, 239
errorBuffer
 _CURL_EXT, 13
eventFeedListener
 sensor_floor_driver.c, 57
example_timer.c
 _taskThread, 51

 _time, 51
 _time_mutex, 51
 _timerTaskId, 52
 CONNECTION_ID, 49
 DEVICE_DATA, 52
 getDeviceData, 50
 main, 50
 resetListener, 50
 timerTask, 51
 EXECUTE_INDEFINITE
 ptask.h, 206
executeTimes
 _Periodic_Task, 29

FALSE
 bool.h, 143
findInResponse
 test_server.c, 321
findWatchItem
 watch.c, 270
freeConnection
 _Comm_Stack, 7

generateId
 ptask.c, 197
generateWatchUri
 watch.c, 270
getAbsUri
 obix_http.c, 122
getDeviceData
 example_timer.c, 50
getHttpConnection
 obix_http.c, 123
getHttpDevice
 obix_http.c, 123
getLastSlashPosition
 xml_storage.c, 289
getLeaseTime
 watch.c, 271
getMetaInfo
 xml_storage.c, 289
 xml_storage.h, 300
getNodeByHref
 xml_storage.c, 289
getNodeByHrefRecursive
 xml_storage.c, 290
getObjectUri
 obix_http.c, 123
getRelUri
 obix_http.c, 123
getStrBatch
 obix_http.c, 123
getStrWatchIn
 obix_http.c, 124

getUriSet
 post_handler.c, 246

handlerBatch
 post_handler.c, 246

handlerError
 post_handler.c, 247

handlerSignUp
 post_handler.c, 247

handlerWatchAdd
 post_handler.c, 247

handlerWatchDelete
 post_handler.c, 247

handlerWatchLongPoll
 post_handler.c, 247

handlerWatchPollChanges
 post_handler.c, 248

handlerWatchPollHelper
 post_handler.c, 248

handlerWatchPollRefresh
 post_handler.c, 248

handlerWatchRemove
 post_handler.c, 248

handlerWatchServiceMake
 post_handler.c, 248

http_closeConnection
 obix_http.c, 124
 obix_http.h, 134

Http_Connection
 obix_http.h, 134

HTTP_CONTENT_LOCATION
 obix_fcgi.c, 240

Http_Device
 obix_http.h, 134

http_dispose
 obix_http.c, 124
 obix_http.h, 134

http_freeConnection
 obix_http.c, 124
 obix_http.h, 135

http_init
 obix_http.c, 124
 obix_http.h, 135

http_initConnection
 obix_http.c, 125
 obix_http.h, 135

http.openConnection
 obix_http.c, 125
 obix_http.h, 135

http_read
 obix_http.c, 125
 obix_http.h, 136

http_readValue
 obix_http.c, 125

obix_http.h, 136

http_registerDevice
 obix_http.c, 126
 obix_http.h, 136

http_registerListener
 obix_http.c, 126
 obix_http.h, 136

http_sendBatch
 obix_http.c, 126
 obix_http.h, 136

HTTP_STATUS_OK
 obix_fcgi.c, 240

http_unregisterDevice
 obix_http.c, 126
 obix_http.h, 137

http_unregisterListener
 obix_http.c, 126
 obix_http.h, 137

http_writeValue
 obix_http.c, 127
 obix_http.h, 137

id
 _Connection, 11
 _Device, 15
 _Listener, 22
 _Periodic_Task, 29
 _Request, 31
 _Target, 34
 _oBIX_BatchCmd, 26
 _oBIX_Watch, 40

id_gen
 _Task_Thread, 37

initConnection
 _Comm_Stack, 7

inputBuffer
 _CURL_EXT, 13

inputBufferFree
 _CURL_EXT, 13

inputBufferSize
 _CURL_EXT, 13

inputWriter
 curl_ext.c, 67

insertServerAddress
 xml_storage.c, 290

isCancelled
 _Periodic_Task, 29

isConnected
 _Connection, 11

isExecuting
 _Periodic_Task, 29

isPollWaitingMax
 _oBIX_Watch, 40

isResponseSent

test_server.c, 321
items
 oBIX_Watch, 40
ixml_ext.c
 ixmlAttr_getNode, 146
 ixmlAttr_getOwnerElement, 146
 ixmlDocument_getElementByAttrValue, 146
 ixmlDocument_getNode, 147
 ixmlDocument_getRootElement, 147
 ixmlElement_cloneWithLog, 147
 ixmlElement_copyAttributeWithLog, 148
 ixmlElement_freeOwnerDocument, 148
 ixmlElement_getNode, 149
 ixmlElement_parseBuffer, 149
 ixmlElement_removeAttributeWithLog, 150
 ixmlElement_setAttributeWithLog, 150
 ixmlNode_convertToAttr, 150
 ixmlNode_convertToElement, 151
 ixmlNode_freeOwnerDocument, 151
 ixmlNode_getElementByAttrValue, 151
 ixmlNode_parseBuffer, 152
ixml_ext.h
 BOOL_H_, 154
 ixmlAttr_getNode, 154
 ixmlAttr_getOwnerElement, 154
 ixmlDocument_getElementByAttrValue, 155
 ixmlDocument_getNode, 155
 ixmlDocument_getRootElement, 155
 ixmlElement_cloneWithLog, 156
 ixmlElement_copyAttributeWithLog, 156
 ixmlElement_freeOwnerDocument, 157
 ixmlElement_getNode, 157
 ixmlElement_parseBuffer, 158
 ixmlElement_removeAttributeWithLog, 158
 ixmlElement_setAttributeWithLog, 158
 ixmlNode_convertToAttr, 159
 ixmlNode_convertToElement, 159
 ixmlNode_freeOwnerDocument, 160
 ixmlNode_parseBuffer, 160
ixmlAttr_getNode
 ixml_ext.c, 146
 ixml_ext.h, 154
ixmlAttr_getOwnerElement
 ixml_ext.c, 146
 ixml_ext.h, 154
ixmlDocument_getElementByAttrValue
 ixml_ext.c, 146
 ixml_ext.h, 155
ixmlDocument_getNode
 ixml_ext.c, 147
 ixml_ext.h, 155
ixmlDocument_getRootElement
 ixml_ext.c, 147
 ixml_ext.h, 155
ixmlElement_cloneWithLog
 ixml_ext.c, 147
 ixml_ext.h, 156
ixmlElement_copyAttributeWithLog
 ixml_ext.c, 148
 ixml_ext.h, 156
ixmlElement_freeOwnerDocument
 ixml_ext.c, 148
 ixml_ext.h, 157
ixmlElement_getFullHref
 obix_http.c, 127
ixmlElement_getNode
 ixml_ext.c, 149
 ixml_ext.h, 157
ixmlElement_parseBuffer
 ixml_ext.c, 149
 ixml_ext.h, 158
ixmlElement_removeAttributeWithLog
 ixml_ext.c, 150
 ixml_ext.h, 158
ixmlElement_setAttributeWithLog
 ixml_ext.c, 150
 ixml_ext.h, 158
ixmlNode_convertToAttr
 ixml_ext.c, 150
 ixml_ext.h, 159
ixmlNode_convertToElement
 ixml_ext.c, 151
 ixml_ext.h, 159
ixmlNode_freeOwnerDocument
 ixml_ext.c, 151
 ixml_ext.h, 160
ixmlNode_getElementByAttrValue
 ixml_ext.c, 151
ixmlNode_parseBuffer
 ixml_ext.c, 152
 ixml_ext.h, 160

keys
 _Table, 32

LAST_EVENT_POLL_PERIOD
 sensor_floor_driver.c, 55
leaseTimerId
 oBIX_Watch, 41
Listener
 obix_comm.h, 114
listener_free
 obix_client.c, 85
listener_register
 obix_client.c, 85
listener_unregister
 obix_client.c, 85
listenerCount

_Device, 15
listeners
 _Device, 15
LISTENSOCK_FILENO
 obix_fcgi.c, 235
 server.c, 261
LISTENSOCK_FLAGS
 obix_fcgi.c, 235
 server.c, 261
loadDeviceData
 sensor_floor_driver.c, 57
loadSettings
 sensor_floor_driver.c, 58
lobbyUri
 _Http_Connection, 17
LOG_LEVEL_DEBUG
 log_utils.h, 169
LOG_LEVEL_ERROR
 log_utils.h, 169
LOG_LEVEL_NO
 log_utils.h, 169
LOG_LEVEL_WARNING
 log_utils.h, 169
log_utils.h
 LOG_LEVEL_DEBUG, 169
 LOG_LEVEL_ERROR, 169
 LOG_LEVEL_NO, 169
 LOG_LEVEL_WARNING, 169
log_debug
 log_utils.h, 167
log_debugHandler
 log_utils.c, 165
 log_utils.h, 170
log_debugPrintf
 log_utils.c, 162
log_debugSyslog
 log_utils.c, 162
log_error
 log_utils.h, 167
log_errorHandler
 log_utils.c, 165
 log_utils.h, 170
log_errorPrintf
 log_utils.c, 162
log_errorSyslog
 log_utils.c, 162
log_function
 log_utils.h, 169
LOG_LEVEL
 log_utils.h, 169
log_nothing
 log_utils.c, 163
log_setLevel
 log_utils.c, 163
 log_utils.h, 169
log_usePrintf
 log_utils.c, 163
 log_utils.h, 170
log_useSyslog
 log_utils.c, 163
 log_utils.h, 170
log_utils.c
 _log_level, 164
 _use_syslog, 164
log_debugHandler, 165
log_debugPrintf, 162
log_debugSyslog, 162
log_errorHandler, 165
log_errorPrintf, 162
log_errorSyslog, 162
log_nothing, 163
log_setLevel, 163
log_usePrintf, 163
log_useSyslog, 163
log_warningHandler, 165
log_warningPrintf, 164
log_warningSyslog, 164
setPrintf, 164
setSyslog, 164
log_utils.h
 log_debug, 167
 log_debugHandler, 170
 log_error, 167
 log_errorHandler, 170
 log_function, 169
 LOG_LEVEL, 169
 log_setLevel, 169
 log_usePrintf, 170
 log_useSyslog, 170
 log_warning, 168
 log_warningHandler, 171
log_warning
 log_utils.h, 168
log_warningHandler
 log_utils.c, 165
 log_utils.h, 171
log_warningPrintf
 log_utils.c, 164
log_warningSyslog
 log_utils.c, 164
main
 example_timer.c, 50
 obix_fcgi.c, 235
 sensor_floor_driver.c, 58
 test_main.c, 315
MAX_PARALLEL_REQUEST_DEFAULT
 obix_fcgi.c, 235

MAX_WATCH_COUNT
 watch.c, 278

maxDevices
 _Connection, 11

maxListeners
 _Connection, 12

name
 _TreeNode, 38

neighbor
 _TreeNode, 38

new
 _Target, 34

newValue
 _oBIX_BatchCmd, 26

next
 _Periodic_Task, 29
 _Request, 31
 _oBIX_BatchCmd, 26
 oBIX_Watch_Item, 43
 Response, 47

nextScheduledTime
 _Periodic_Task, 30

normalizeObixDocument
 server.c, 261

normalizeUri
 server.c, 262

obix_batch.h
 OBIX_BATCH_READ, 80
 OBIX_BATCH_READ_VALUE, 80
 OBIX_BATCH_WRITE_VALUE, 80

OBIX_BATCH_READ
 obix_batch.h, 80

OBIX_BATCH_READ_VALUE
 obix_batch.h, 80

OBIX_BATCH_WRITE_VALUE
 obix_batch.h, 80

obix_client.h
 OBIX_ERR_BAD_CONNECTION, 99
 OBIX_ERR_HTTP_LIB, 99
 OBIX_ERR_INVALID_ARGUMENT, 99
 OBIX_ERR_INVALID_STATE, 99
 OBIX_ERR_LIMIT_REACHED, 99
 OBIX_ERR_NO_MEMORY, 99
 OBIX_ERR_SERVER_ERROR, 99
 OBIX_ERR_UNKNOWN_BUG, 99
 OBIX_SUCCESS, 99
 OBIX_T_ABSTIME, 99
 OBIX_T_BOOL, 99
 OBIX_T_ENUM, 99
 OBIX_T_INT, 99
 OBIX_T_REAL, 99
 OBIX_T_RELTIME, 99

OBIX_T_STR, 99
OBIX_T_URI, 99

obix_comm.h
 OBIX_HTTP, 114

OBIX_ERR_BAD_CONNECTION
 obix_client.h, 99

OBIX_ERR_HTTP_LIB
 obix_client.h, 99

OBIX_ERR_INVALID_ARGUMENT
 obix_client.h, 99

OBIX_ERR_INVALID_STATE
 obix_client.h, 99

OBIX_ERR_LIMIT_REACHED
 obix_client.h, 99

OBIX_ERR_NO_MEMORY
 obix_client.h, 99

OBIX_ERR_SERVER_ERROR
 obix_client.h, 99

OBIX_ERR_UNKNOWN_BUG
 obix_client.h, 99

OBIX_HTTP
 obix_comm.h, 114

OBIX_SUCCESS
 obix_client.h, 99

OBIX_T_ABSTIME
 obix_client.h, 99

OBIX_T_BOOL
 obix_client.h, 99

OBIX_T_ENUM
 obix_client.h, 99

OBIX_T_INT
 obix_client.h, 99

OBIX_T_REAL
 obix_client.h, 99

OBIX_T_RELTIME
 obix_client.h, 99

OBIX_T_STR
 obix_client.h, 99

OBIX_T_URI
 obix_client.h, 99

obix_utils.h
 RELTIME_DAY, 186
 RELTIME_HOUR, 186
 RELTIME_MIN, 186
 RELTIME_MONTH, 187
 RELTIME_SEC, 186
 RELTIME_YEAR, 187

OBIX_ATTR_DISPLAY
 obix_utils.c, 176
 obix_utils.h, 188

OBIX_ATTR_DISPLAY_NAME
 obix_utils.c, 176
 obix_utils.h, 188

OBIX_ATTR_HREF

obix_utils.c, 176
obix_utils.h, 188
OBIX_ATTR_IS
 obix_utils.c, 176
 obix_utils.h, 189
OBIX_ATTR_NAME
 obix_utils.c, 176
 obix_utils.h, 189
OBIX_ATTR_NULL
 obix_utils.c, 177
 obix_utils.h, 189
OBIX_ATTR_VAL
 obix_utils.c, 177
 obix_utils.h, 189
OBIX_ATTR_WRITABLE
 obix_utils.c, 177
 obix_utils.h, 189
oBIX_Batch
 obix_client.h, 97
obix_batch.c
 obix_batch_addCommand, 74
 obix_batch_commandFree, 74
 obix_batch_create, 74
 OBIX_BATCH_EMPTY_RESULT, 74
 obix_batch_free, 75
 obix_batch_getResult, 75
 obix_batch_read, 75
 obix_batch_readValue, 76
 obix_batch_removeCommand, 77
 obix_batch_resultClear, 77
 obix_batch_resultInit, 77
 obix_batch_send, 77
 obix_batch_writeValue, 78
 strncpy, 79
obix_batch.h
 OBIX_BATCH_CMD_TYPE, 80
 oBIX_BatchCmd, 80
obix_batch_addCommand
 obix_batch.c, 74
OBIX_BATCH_CMD_TYPE
 obix_batch.h, 80
obix_batch_commandFree
 obix_batch.c, 74
obix_batch_create
 obix_batch.c, 74
 obix_client.h, 100
OBIX_BATCH_EMPTY_RESULT
 obix_batch.c, 74
obix_batch_free
 obix_batch.c, 75
 obix_client.h, 100
obix_batch_getResult
 obix_batch.c, 75
 obix_client.h, 100
obix_batch_read
 obix_batch.c, 75
 obix_client.h, 101
obix_batch_readValue
 obix_batch.c, 76
 obix_client.h, 101
obix_batch_removeCommand
 obix_batch.c, 77
 obix_client.h, 102
obix_batch_resultClear
 obix_batch.c, 77
obix_batch_resultInit
 obix_batch.c, 77
obix_batch_send
 obix_batch.c, 77
 obix_client.h, 102
OBIX_BATCH_TEMPLATE_CMD_READ
 obix_http.c, 118
OBIX_BATCH_TEMPLATE_CMD_READ_LENGTH
 obix_http.c, 119
OBIX_BATCH_TEMPLATE_CMD_WRITE
 obix_http.c, 119
OBIX_BATCH_TEMPLATE_CMD_WRITE_LENGTH
 obix_http.c, 119
OBIX_BATCH_TEMPLATE_FOOTER
 obix_http.c, 119
OBIX_BATCH_TEMPLATE_FOOTER_LENGTH
 obix_http.c, 119
OBIX_BATCH_TEMPLATE_HEADER
 obix_http.c, 119
OBIX_BATCH_TEMPLATE_HEADER_LENGTH
 obix_http.c, 120
obix_batch_writeValue
 obix_batch.c, 78
 obix_client.h, 103
oBIX_BatchCmd
 obix_batch.h, 80
oBIX_BatchResult
 obix_client.h, 98
obix_client.c
 _connectionCount, 93
 _connections, 93
 connection_create, 83
 connection_free, 83
 connection_get, 83
 CT_CONNECTION, 93
 CT_MAX_DEVICES, 93
 CT_MAX_LISTENERS, 93
 CTA_CONNECTION_ID, 93
 CTA_CONNECTION_TYPE, 93
 CTAV_CONNECTION_TYPE_HTTP, 93

DEFAULT_MAX_DEVICES, 83
DEFAULT_MAX_LISTENERS, 83
device_free, 84
device_get, 84
device_register, 84
device_unregister, 84
device_unregisterAllListeners, 84
listener_free, 85
listener_register, 85
listener_unregister, 85
obix_closeConnection, 85
obix_dispose, 86
obix_loadConfig, 86
obix_loadConfigFile, 86
obix_openConnection, 87
obix_read, 87
obix_readValue, 88
obix_registerDevice, 89
obix_registerListener, 90
obix_unregisterDevice, 91
obix_unregisterListener, 91
obix_writeValue, 92

obix_client.h
 oBIX_Batch, 97
 obix_batch_create, 100
 obix_batch_free, 100
 obix_batch_getResult, 100
 obix_batch_read, 101
 obix_batch_readValue, 101
 obix_batch_removeCommand, 102
 obix_batch_send, 102
 obix_batch_writeValue, 103
 oBIX_BatchResult, 98
 obix_closeConnection, 104
 OBIX_DATA_TYPE, 98
 obix_dispose, 104
 OBIX_ERRORCODE, 99
 obix_loadConfig, 104
 obix_loadConfigFile, 105
 obix_openConnection, 105
 obix_read, 106
 obix_readValue, 107
 obix_registerDevice, 108
 obix_registerListener, 108
 obix_unregisterDevice, 109
 obix_unregisterListener, 110
 obix_update_listener, 98
 obix_writeValue, 110

obix_client_testListener
 test_server.c, 322

obix_closeConnection
 obix_client.c, 85
 obix_client.h, 104

obix_comm.h

comm_closeConnection, 112
comm_freeConnection, 112
comm_initConnection, 113
comm_openConnection, 113
comm_read, 113
comm_readValue, 113
comm_registerDevice, 113
comm_registerListener, 113
comm_sendBatch, 113
Comm_Stack, 113
comm_unregisterDevice, 114
comm_unregisterListener, 114
comm_writeValue, 114
Connection, 114
connection_get, 115
Connection_Type, 114
Device, 114
device_get, 115
Listener, 114

OBIX_CONTRACT_ERR_BAD_URI
 obix_utils.c, 177
 obix_utils.h, 190

OBIX_CONTRACT_ERR_PERMISSION
 obix_utils.c, 177
 obix_utils.h, 190

OBIX_CONTRACT_ERR_UNSUPPORTED
 obix_utils.c, 178
 obix_utils.h, 190

OBIX_CONTRACT_LONG_POLL_WATCH
 obix_http.c, 131

OBIX_DATA_TYPE
 obix_client.h, 98

OBIX_DATA_TYPE_NAMES
 obix_http.c, 131

obix_dispose
 obix_client.c, 86
 obix_client.h, 104

OBIX_ERRORCODE
 obix_client.h, 99

obix_fcgi.c
 _requestIds, 238
 _requestList, 238
 _requestListMutex, 238
 _requestListNew, 239
 _requestMaxCount, 239
 _requestsInUse, 239
 CONFIG_FILE, 239
 CT_HOLD_REQUEST_MAX, 239
 ERROR_STATIC, 239
 HTTP_CONTENT_LOCATION, 240
 HTTP_STATUS_OK, 240
 LISTENSOCK_FILENO, 235
 LISTENSOCK_FLAGS, 235
 main, 235

MAX_PARALLEL_REQUEST_DEFAULT, 235
obix_fcgi_dumpEnvironment, 235
obix_fcgi_handleRequest, 236
obix_fcgi_init, 236
obix_fcgi_loadConfig, 236
obix_fcgi_readRequestInput, 236
obix_fcgi_sendResponse, 236
obix_fcgi_sendStaticErrorMessage, 237
obix_fcgi_shutdown, 237
obixRequest_create, 237
obixRequest_free, 237
obixRequest_get, 237
obixRequest_getHead, 238
obixRequest_release, 238
obixRequest_wait, 238
parseArguments, 238
XML_HEADER, 240
obix_fcgi.h
 obix_fcgi_dumpEnvironment, 241
 obix_fcgi_handleRequest, 242
 obix_fcgi_init, 242
 obix_fcgi_readRequestInput, 242
 obix_fcgi_sendResponse, 242
 obix_fcgi_sendStaticErrorMessage, 242
 obix_fcgi_shutdown, 242
obix_fcgi_dumpEnvironment
 obix_fcgi.c, 235
 obix_fcgi.h, 241
obix_fcgi_handleRequest
 obix_fcgi.c, 236
 obix_fcgi.h, 242
obix_fcgi_init
 obix_fcgi.c, 236
 obix_fcgi.h, 242
obix_fcgi_loadConfig
 obix_fcgi.c, 236
obix_fcgi_readRequestInput
 obix_fcgi.c, 236
 obix_fcgi.h, 242
obix_fcgi_sendResponse
 obix_fcgi.c, 236
 obix_fcgi.h, 242
obix_fcgi_sendStaticErrorMessage
 obix_fcgi.c, 237
 obix_fcgi.h, 242
obix_fcgi_shutdown
 obix_fcgi.c, 237
 obix_fcgi.h, 242
obix_http.c
 (curl_handle, 130
 (curl_watch_handle, 130
 initialized, 130
 watchThread, 130
addListener, 121
addWatchItem, 121
checkResponseDoc, 122
checkResponseElement, 122
createWatch, 122
CT_LONG_POLL, 130
CT_LONG_POLL_MAX, 130
CT_LONG_POLL_MIN, 131
CT_POLL_INTERVAL, 131
CT_SERVER_ADDRESS, 131
CT_WATCHLEASE, 131
CTA_LOBBY, 131
DEFAULT_POLLING_INTERVAL, 118
DEFAULT_WATCHLEASE_PADDING, 118
getAbsUri, 122
getHttpConnection, 123
getHttpDevice, 123
getObjectUri, 123
getRelUri, 123
getStrBatch, 123
getStrWatchIn, 124
http_closeConnection, 124
http_dispose, 124
http_freeConnection, 124
http_init, 124
http_initConnection, 125
http_openConnection, 125
http_read, 125
http_readValue, 125
http_registerDevice, 126
http_registerListener, 126
http_sendBatch, 126
http_unregisterDevice, 126
http_unregisterListener, 126
http_writeValue, 127
ixmlelement_getFullHref, 127
OBIX_BATCH_TEMPLATE_CMD_READ, 118
OBIX_BATCH_TEMPLATE_CMD_READ_LENGTH, 119
OBIX_BATCH_TEMPLATE_CMD_WRITE, 119
OBIX_BATCH_TEMPLATE_CMD_WRITE_LENGTH, 119
OBIX_BATCH_TEMPLATE_FOOTER, 119
OBIX_BATCH_TEMPLATE_FOOTER_LENGTH, 119
OBIX_BATCH_TEMPLATE_HEADER, 119
OBIX_BATCH_TEMPLATE_HEADER_LENGTH, 120
OBIX_CONTRACT_LONG_POLL_WATCH, 131
OBIX_DATA_TYPE_NAMES, 131
OBIX_HTTP_COMM_STACK, 132

OBIX_WATCH_IN_TEMPLATE_FOOTER,
 120
OBIX_WATCH_IN_TEMPLATE_FOOTER_-
 LENGTH, 120
OBIX_WATCH_IN_TEMPLATE_HEADER,
 120
OBIX_WATCH_IN_TEMPLATE_-
 HEADER_LENGTH, 120
OBIX_WATCH_IN_TEMPLATE_URI, 120
OBIX_WATCH_IN_TEMPLATE_URI_-
 LENGTH, 121
OBIX_WATCH_OUT_VALUES, 132
OBIX_WRITE_REQUEST_TEMPLATE, 121
OBIX_WRITE_REQUEST_TEMPLATE_-
 LENGTH, 121
parseElementValue, 127
parseWatchOut, 127
recreateWatch, 127
removeListener, 128
removeServerAddress, 128
removeWatch, 128
resetWatchUris, 128
setWatchLeaseTime, 129
setWatchPollWaitTime, 129
setWatchTimeParam, 129
watchPollTask, 129
writeValue, 129
obix_http.h
 http_closeConnection, 134
 Http_Connection, 134
 Http_Device, 134
 http_dispose, 134
 http_freeConnection, 135
 http_init, 135
 http_initConnection, 135
 http_openConnection, 135
 http_read, 136
 http_readValue, 136
 http_registerDevice, 136
 http_registerListener, 136
 http_sendBatch, 136
 http_unregisterDevice, 137
 http_unregisterListener, 137
 http_writeValue, 137
 OBIX_HTTP_COMM_STACK, 137
OBIX_HTTP_COMM_STACK
 obix_http.c, 132
 obix_http.h, 137
obix_loadConfig
 obix_client.c, 86
 obix_client.h, 104
obix_loadConfigFile
 obix_client.c, 86
 obix_client.h, 105
OBIX_META
 xml_storage.c, 297
 xml_storage.h, 307
OBIX_META_ATTR_OP
 server.c, 265
OBIX_META_ATTR_WATCH_TEMPLATE
 watch.c, 278
OBIX_META_WATCH_UPDATED_NO
 watch.c, 278
 watch.h, 285
OBIX_META_WATCH_UPDATED_YES
 watch.c, 278
 watch.h, 285
OBIX_NAME_BATCH
 obix_utils.c, 178
 obix_utils.h, 190
OBIX_NAME_SIGN_UP
 obix_utils.c, 178
 obix_utils.h, 190
OBIX_NAME_WATCH_ADD
 obix_utils.c, 178
 obix_utils.h, 190
OBIX_NAME_WATCH_DELETE
 obix_utils.c, 178
 obix_utils.h, 191
OBIX_NAME_WATCHLEASE
 obix_utils.c, 178
 obix_utils.h, 191
OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL
 obix_utils.c, 179
 obix_utils.h, 191
OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL_MAX
 obix_utils.c, 179
 obix_utils.h, 191
OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL_MIN
 obix_utils.c, 179
 obix_utils.h, 191
OBIX_NAME_WATCH_POLLCHANGES
 obix_utils.c, 179
 obix_utils.h, 191
OBIX_NAME_WATCH_POLLREFRESH
 obix_utils.c, 179
 obix_utils.h, 192
OBIX_NAME_WATCH_REMOVE
 obix_utils.c, 179
 obix_utils.h, 192
OBIX_NAME_WATCH_SERVICE
 obix_utils.c, 180
 obix_utils.h, 192
OBIX_NAME_WATCH_SERVICE_MAKE
 obix_utils.c, 180

obix_utils.h, 192
OBIX_OBJ
 obix_utils.c, 180
 obix_utils.h, 192
OBIX_OBJ_ABSTIME
 obix_utils.c, 180
 obix_utils.h, 192
OBIX_OBJ_BOOL
 obix_utils.c, 180
 obix_utils.h, 193
OBIX_OBJ_ENUM
 obix_utils.c, 180
 obix_utils.h, 193
OBIX_OBJ_ERR
 obix_utils.c, 180
 obix_utils.h, 193
OBIX_OBJ_ERR_TEMPLATE
 response.c, 256
OBIX_OBJ_FEED
 obix_utils.c, 181
 obix_utils.h, 193
obix_obj_implementsContract
 obix_utils.c, 174
 obix_utils.h, 187
OBIX_OBJ_INT
 obix_utils.c, 181
 obix_utils.h, 193
OBIX_OBJ_LIST
 obix_utils.c, 181
 obix_utils.h, 193
OBIX_OBJ_NULL_TEMPLATE
 obix_utils.c, 181
 obix_utils.h, 194
OBIX_OBJ_OP
 obix_utils.c, 181
 obix_utils.h, 194
OBIX_OBJ_REAL
 obix_utils.c, 181
 obix_utils.h, 194
OBIX_OBJ_REF
 obix_utils.c, 182
 obix_utils.h, 194
OBIX_OBJ_RELTIME
 obix_utils.c, 182
 obix_utils.h, 194
OBIX_OBJ_STR
 obix_utils.c, 182
 obix_utils.h, 194
OBIX_OBJ_URI
 obix_utils.c, 182
 obix_utils.h, 194
obix_openConnection
 obix_client.c, 87
 obix_client.h, 105
obix_read
 obix_client.c, 87
 obix_client.h, 106
obix_readValue
 obix_client.c, 88
 obix_client.h, 107
obix_registerDevice
 obix_client.c, 89
 obix_client.h, 108
obix_registerListener
 obix_client.c, 90
 obix_client.h, 108
obix_reltime_fromLong
 obix_utils.c, 175
 obix_utils.h, 187
obix_reltime_parseToLong
 obix_utils.c, 175
 obix_utils.h, 187
obix_response_listener
 response.h, 257
obix_server_generateErrorMessage
 server.c, 262
 server.h, 266
obix_server_generateResponse
 server.c, 263
 server.h, 266
obix_server_getPostHandler
 post_handler.c, 249
 post_handler.h, 252
obix_server_handleGET
 server.c, 263
 server.h, 267
obix_server_handlePOST
 server.c, 263
 server.h, 267
obix_server_handlePUT
 server.c, 263
 server.h, 267
obix_server_init
 server.c, 263
 server.h, 267
obix_server_invoke
 server.c, 264
 server.h, 267
obix_server_postHandler
 post_handler.h, 251
obix_server_read
 server.c, 264
 server.h, 268
obix_server_shutdown
 server.c, 264
 server.h, 268
obix_server_write
 server.c, 264

server.h, 268
OBIX_STORAGE_FILES
 xml_storage.c, 298
OBIX_STORAGE_FILES_COUNT
 xml_storage.c, 298
OBIX_SYS_ERROR_STUB
 xml_storage.c, 298
 xml_storage.h, 307
OBIX_SYS_WATCH_OUT_STUB
 xml_storage.c, 298
 xml_storage.h, 307
OBIX_SYS_WATCH_STUB
 xml_storage.c, 298
 xml_storage.h, 307
obix_unregisterDevice
 obix_client.c, 91
 obix_client.h, 109
obix_unregisterListener
 obix_client.c, 91
 obix_client.h, 110
obix_update_listener
 obix_client.h, 98
obix_utils.c
 OBIX_ATTR_DISPLAY, 176
 OBIX_ATTR_DISPLAY_NAME, 176
 OBIX_ATTR_HREF, 176
 OBIX_ATTR_IS, 176
 OBIX_ATTR_NAME, 176
 OBIX_ATTR_NULL, 177
 OBIX_ATTR_VAL, 177
 OBIX_ATTR_WRITABLE, 177
 OBIX_CONTRACT_ERR_BAD_URI, 177
 OBIX_CONTRACT_ERR_PERMISSION,
 177
 OBIX_CONTRACT_ERR_UNSUPPORTED,
 178
 OBIX_NAME_BATCH, 178
 OBIX_NAME_SIGN_UP, 178
 OBIX_NAME_WATCH_ADD, 178
 OBIX_NAME_WATCH_DELETE, 178
 OBIX_NAME_WATCHLEASE, 178
 OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL, 179
 OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL_MAX, 179
 OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL_MIN, 179
 OBIX_NAME_WATCH_POLLCHANGES,
 179
 OBIX_NAME_WATCH_POLLREFRESH,
 179
 OBIX_NAME_WATCH_REMOVE, 179
 OBIX_NAME_WATCH_SERVICE, 180
OBIX_NAME_WATCH_SERVICE_MAKE,
 180
OBIX_OBJ, 180
OBIX_OBJ_ABSTIME, 180
OBIX_OBJ_BOOL, 180
OBIX_OBJ_ENUM, 180
OBIX_OBJ_ERR, 180
OBIX_OBJ_FEED, 181
obix_obj_implementsContract, 174
OBIX_OBJ_INT, 181
OBIX_OBJ_LIST, 181
OBIX_OBJ_NULL_TEMPLATE, 181
OBIX_OBJ_OP, 181
OBIX_OBJ_REAL, 181
OBIX_OBJ_REF, 182
OBIX_OBJ_RELTIME, 182
OBIX_OBJ_STR, 182
OBIX_OBJ_URI, 182
obix_reltimes_fromLong, 175
obix_reltimes_parseToLong, 175
XML_FALSE, 182
XML_TRUE, 182
obix_utils.h
 OBIX_ATTR_DISPLAY, 188
 OBIX_ATTR_DISPLAY_NAME, 188
 OBIX_ATTR_HREF, 188
 OBIX_ATTR_IS, 189
 OBIX_ATTR_NAME, 189
 OBIX_ATTR_NULL, 189
 OBIX_ATTR_VAL, 189
 OBIX_ATTR_WRITABLE, 189
 OBIX_CONTRACT_ERR_BAD_URI, 190
 OBIX_CONTRACT_ERR_PERMISSION,
 190
 OBIX_CONTRACT_ERR_UNSUPPORTED,
 190
 OBIX_NAME_BATCH, 190
 OBIX_NAME_SIGN_UP, 190
 OBIX_NAME_WATCH_ADD, 190
 OBIX_NAME_WATCH_DELETE, 191
 OBIX_NAME_WATCHLEASE, 191
 OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL, 191
 OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL_MAX, 191
 OBIX_NAME_WATCH_POLL_WAIT_-
 INTERVAL_MIN, 191
 OBIX_NAME_WATCH_POLLCHANGES,
 191
 OBIX_NAME_WATCH_POLLREFRESH,
 192
 OBIX_NAME_WATCH_REMOVE, 192
 OBIX_NAME_WATCH_SERVICE, 192

OBIX_NAME_WATCH_SERVICE_MAKE,
 192
OBIX_OBJ, 192
OBIX_OBJ_ABSTIME, 192
OBIX_OBJ_BOOL, 193
OBIX_OBJ_ENUM, 193
OBIX_OBJ_ERR, 193
OBIX_OBJ_FEED, 193
obix_obj_implementsContract, 187
OBIX_OBJ_INT, 193
OBIX_OBJ_LIST, 193
OBIX_OBJ_NULL_TEMPLATE, 194
OBIX_OBJ_OP, 194
OBIX_OBJ_REAL, 194
OBIX_OBJ_REF, 194
OBIX_OBJ_RELTIME, 194
OBIX_OBJ_STR, 194
OBIX_OBJ_URI, 194
obix_reltimes_fromLong, 187
obix_reltimes_parseToLong, 187
RELTIME_FORMAT, 186
XML_FALSE, 195
XML_TRUE, 195
oBIX_Watch, 39
 id, 40
 isPollWaitingMax, 40
 items, 40
 leaseTimerId, 41
 pollTaskCompleted, 41
 pollTaskId, 41
 pollTaskMutex, 41
 pollWaitMax, 41
 pollWaitMin, 42
OBIX_WATCH_IN_TEMPLATE_FOOTER
 obix_http.c, 120
OBIX_WATCH_IN_TEMPLATE_FOOTER_-
 LENGTH
 obix_http.c, 120
OBIX_WATCH_IN_TEMPLATE_HEADER
 obix_http.c, 120
OBIX_WATCH_IN_TEMPLATE_HEADER_-
 LENGTH
 obix_http.c, 120
OBIX_WATCH_IN_TEMPLATE_URI
 obix_http.c, 120
OBIX_WATCH_IN_TEMPLATE_URI_LENGTH
 obix_http.c, 121
oBIX_Watch_Item, 42
 doc, 43
 next, 43
 updated, 43
 uri, 43
OBIX_WATCHLEASE_NO_CHANGE
 watch.c, 278
 watch.h, 285
OBIX_WATCH_OUT_VALUES
 obix_http.c, 132
OBIX_WRITE_REQUEST_TEMPLATE
 obix_http.c, 121
OBIX_WRITE_REQUEST_TEMPLATE_-
 LENGTH
 obix_http.c, 121
obix_writeValue
 obix_client.c, 92
 obix_client.h, 110
obixRequest_create
 obix_fcgi.c, 237
obixRequest_free
 obix_fcgi.c, 237
obixRequest_get
 obix_fcgi.c, 237
obixRequest_getHead
 obix_fcgi.c, 238
obixRequest_release
 obix_fcgi.c, 238
obixRequest_wait
 obix_fcgi.c, 238
obixResponse_add
 response.c, 254
 response.h, 257
obixResponse_create
 response.c, 254
 response.h, 258
obixResponse_free
 response.c, 254
 response.h, 258
obixResponse_isError
 response.c, 254
 response.h, 258
obixResponse_isHead
 response.c, 254
 response.h, 258
obixResponse_send
 response.c, 254
 response.h, 258
obixResponse_setError
 response.c, 255
 response.h, 259
obixResponse_setErrorFlag
 response.c, 255
 response.h, 259
obixResponse_setListener
 response.c, 255
 response.h, 259
obixResponse_setRightUri
 response.c, 255
 response.h, 259
obixResponse_setText

response.c, 255
response.h, 259
obixWatch_appendWatchItem
 watch.c, 271
 watch.h, 281
obixWatch_create
 watch.c, 271
 watch.h, 281
obixWatch_createWatchItem
 watch.c, 272
 watch.h, 281
obixWatch_delete
 watch.c, 272
 watch.h, 282
obixWatch_deleteHelper
 watch.c, 273
obixWatch_deleteWatchItem
 watch.c, 273
 watch.h, 282
obixWatch_dispose
 watch.c, 273
 watch.h, 282
obixWatch_free
 watch.c, 273
obixWatch_get
 watch.c, 273
 watch.h, 283
obixWatch_getByUri
 watch.c, 273
 watch.h, 283
obixWatch_getUri
 watch.c, 274
 watch.h, 283
obixWatch_getWatchItem
 watch.c, 274
 watch.h, 283
obixWatch_holdPoll
 watch.c, 274
 watch.h, 283
obixWatch_init
 watch.c, 274
 watch.h, 284
obixWatch_isLongPoll
 watch.c, 274
 watch.h, 284
obixWatch_isWatchUri
 watch.c, 275
 watch.h, 284
obixWatch_longPollTask
 watch.c, 275
obixWatch_notifyPollTask
 watch.c, 275
obixWatch_pollHandler
 watch.h, 280

obixWatch_processTimeUpdates
 watch.c, 275
 watch.h, 284
obixWatch_resetLeaseTimer
 watch.c, 275
 watch.h, 284
obixWatch_updateMeta
 watch.c, 276
 watch.h, 284
obixWatchItem_free
 watch.c, 276
obixWatchItem_freeRecursive
 watch.c, 276
obixWatchItem_isUpdated
 watch.c, 276
 watch.h, 285
obixWatchItem_setUpdated
 watch.c, 277
 watch.h, 285
obj
 _oBIX_BatchResult, 27
openConnection
 _Comm_Stack, 7
outputBuffer
 _CURL_EXT, 13
outputPos
 _CURL_EXT, 13
outputReader
 curl_ext.c, 67
outputSize
 _CURL_EXT, 14
paramUri
 _Listener, 22
 _oBIX_BatchCmd, 26
parseArguments
 obix_fcgi.c, 238
parseElementValue
 obix_http.c, 127
parseTarget
 sensor_floor_driver.c, 59
parseWatchOut
 obix_http.c, 127
parseXmlInput
 curl_ext.c, 68
period
 _Periodic_Task, 30
Periodic_Task
 ptask.c, 197
periodic_task
 ptask.h, 206
periodicTask_create
 ptask.c, 197
periodicTask_deleteRecursive

ptask.c, 198
periodicTask_execute
 ptask.c, 198
periodicTask_free
 ptask.c, 198
periodicTask_generateNextExecTime
 ptask.c, 198
periodicTask_get
 ptask.c, 198
periodicTask_getClosest
 ptask.c, 198
periodicTask_removeFromList
 ptask.c, 199
periodicTask_resetExecTime
 ptask.c, 199
periodicTask_setPeriod
 ptask.c, 199
pollHandler
 PollTaskParams, 45
pollInterval
 _Http_Connection, 17
pollTaskCompleted
 oBIX_Watch, 41
pollTaskId
 oBIX_Watch, 41
pollTaskMutex
 oBIX_Watch, 41
PollTaskParams, 44
 pollHandler, 45
 response, 45
 uri, 45
 watch, 45
pollWaitMax
 _Http_Connection, 17
 oBIX_Watch, 41
pollWaitMin
 _Http_Connection, 17
 oBIX_Watch, 42
pollWatchItemIterator
 post_handler.c, 249
POST_HANDLER
 post_handler.c, 250
post_handler.c
 addResponsePart, 245
 BATCH_OUT_POSTFIX, 245
 BATCH_OUT_PREFIX, 245
 completeWatchPollResponse, 246
 DEVICE_LIST_URI, 245
 getUriSet, 246
 handlerBatch, 246
 handlerError, 247
 handlerSignUp, 247
 handlerWatchAdd, 247
 handlerWatchDelete, 247
handlerWatchLongPoll, 247
handlerWatchPollChanges, 248
handlerWatchPollHelper, 248
handlerWatchPollRefresh, 248
handlerWatchRemove, 248
handlerWatchServiceMake, 248
obix_server_getPostHandler, 249
pollWatchItemIterator, 249
POST_HANDLER, 250
POST_HANDLERS_COUNT, 250
processWatchIn, 249
putDeviceReference, 249
sendErrorMessage, 249
WATCH_OUT_POSTFIX, 245
WATCH_OUT_PREFIX, 245
post_handler.h
 obix_server_getPostHandler, 252
 obix_server_postHandler, 251
POST_HANDLERS_COUNT
 post_handler.c, 250
prev
 _Periodic_Task, 30
printResponse
 test_server.c, 322
printTestResult
 test_main.c, 315
 test_main.h, 316
printXMLContents
 xml_storage.c, 290
processWatchIn
 post_handler.c, 249
ptask.c
 generateId, 197
 Periodic_Task, 197
 periodicTask_create, 197
 periodicTask_deleteRecursive, 198
 periodicTask_execute, 198
 periodicTask_free, 198
 periodicTask_generateNextExecTime, 198
 periodicTask_get, 198
 periodicTask_getClosest, 198
 periodicTask_removeFromList, 199
 periodicTask_resetExecTime, 199
 periodicTask_setPeriod, 199
 ptask_cancel, 199
 ptask_dispose, 200
 ptask_init, 200
 ptask_isScheduled, 200
 ptask_reschedule, 201
 ptask_reset, 201
 ptask_schedule, 202
 stopTask, 202
 threadCycle, 203
 timespec_add, 203

timespec_cmp, 203
timespec_copy, 203
timespec_fromMillis, 203
ptask.h
 EXECUTE_INDEFINITE, 206
 periodic_task, 206
 ptask_cancel, 206
 ptask_dispose, 207
 ptask_init, 207
 ptask_isScheduled, 208
 ptask_reschedule, 208
 ptask_reset, 209
 ptask_schedule, 209
 Task_Thread, 206
ptask_cancel
 ptask.c, 199
 ptask.h, 206
ptask_dispose
 ptask.c, 200
 ptask.h, 207
ptask_init
 ptask.c, 200
 ptask.h, 207
ptask_isScheduled
 ptask.c, 200
 ptask.h, 208
ptask_reschedule
 ptask.c, 201
 ptask.h, 208
ptask_reset
 ptask.c, 201
 ptask.h, 209
ptask_schedule
 ptask.c, 202
 ptask.h, 209
putDeviceReference
 post_handler.c, 249

r
 _Request, 32
read
 _Comm_Stack, 8
readValue
 _Comm_Stack, 8
recreateWatch
 obix_http.c, 127
registerDevice
 _Comm_Stack, 8
registerListener
 _Comm_Stack, 8
RELTIME_DAY
 obix_utils.h, 186
RELTIME_HOUR
 obix_utils.h, 186
RELTIME_MIN
 obix_utils.h, 186
RELTIME_MONTH
 obix_utils.h, 187
RELTIME_YEAR
 obix_utils.h, 187
RELTIME_FORMAT
 obix_utils.h, 186
removeListener
 obix_http.c, 128
removeMetaInfo
 xml_storage.c, 290
 xml_storage.h, 301
removeServerAddress
 obix_http.c, 128
removeTask
 _Target, 34
removeWatch
 obix_http.c, 128
Request
 request.h, 252
request
 Response, 47
request.h
 Request, 252
REQUEST_HTTP_GET
 test_client.c, 308
REQUEST_HTTP_POST
 curl_ext.c, 65
 test_client.c, 308
REQUEST_HTTP_PUT
 curl_ext.c, 65
 test_client.c, 309
resetListener
 example_timer.c, 50
resetTargets
 sensor_floor_driver.c, 59
resetWatchUris
 obix_http.c, 128
resourceFolder
 xml_config.c, 219
Response, 45
 body, 47
 canWait, 47
 error, 47
 next, 47
 request, 47
 uri, 47
response
 PollTaskParams, 45
response.c
 _responseListener, 256

OBIX_OBJ_ERR_TEMPLATE, 256
obixResponse_add, 254
obixResponse_create, 254
obixResponse_free, 254
obixResponse_isError, 254
obixResponse_isHead, 254
obixResponse_send, 254
obixResponse_setError, 255
obixResponse_setErrorFlag, 255
obixResponse_setListener, 255
obixResponse_setRightUri, 255
obixResponse_setText, 255

response.h
 obix_response_listener, 257
 obixResponse_add, 257
 obixResponse_create, 258
 obixResponse_free, 258
 obixResponse_isError, 258
 obixResponse_isHead, 258
 obixResponse_send, 258
 obixResponse_setError, 259
 obixResponse_setErrorFlag, 259
 obixResponse_setListener, 259
 obixResponse_setRightUri, 259
 obixResponse_setText, 259

result
 _oBIX_Batch, 24
runAutomaticTests
 test_main.c, 315
runManualTests
 test_main.c, 315

sendBatch
 _Comm_Stack, 8
sendErrorMessage
 post_handler.c, 249
sendFallEventUpdate
 sensor_floor_driver.c, 59
sendRequest
 curl_ext.c, 68
SENSOR_FLOOR_CONNECTION
 sensor_floor_driver.c, 56
sensor_floor_driver.c
 _deviceData, 62
 _deviceId, 62
 _targetMutex, 62
 _targets, 62
 _targetsCount, 62
 _taskThread, 63
 TestData, 63
 checkEventsTask, 57
 checkTargets, 57
 eventFeedListener, 57
 LAST_EVENT_POLL_PERIOD, 55

loadDeviceData, 57
loadSettings, 58
main, 58
parseTarget, 59
resetTargets, 59
sendFallEventUpdate, 59
SENSOR_FLOOR_CONNECTION, 56
SENSOR_FLOOR_FEED_CONNECTION,
 56
SERVER_CONNECTION, 56
Target, 56
target_get, 60
TARGET_REMOVE_TIMEOUT, 56
target_sendUpdate, 60
target_setX, 60
target_setY, 61
targetRemoveTask, 61
targetsListener, 61
testCycle, 62
SENSOR_FLOOR_FEED_CONNECTION
 sensor_floor_driver.c, 56
server.c
 CT_SERVER_ADDRESS, 265
 LISTENSOCK_FILENO, 261
 LISTENSOCK_FLAGS, 261
 normalizeObixDocument, 261
 normalizeUri, 262
 OBIX_META_ATTR_OP, 265
 obix_server_generateObixErrorMessage, 262
 obix_server_generateResponse, 263
 obix_server_handleGET, 263
 obix_server_handlePOST, 263
 obix_server_handlePUT, 263
 obix_server_init, 263
 obix_server_invoke, 264
 obix_server_read, 264
 obix_server_shutdown, 264
 obix_server_write, 264
 updateMetaWatch, 264

server.h
 obix_server_generateObixErrorMessage, 266
 obix_server_generateResponse, 266
 obix_server_handleGET, 267
 obix_server_handlePOST, 267
 obix_server_handlePUT, 267
 obix_server_init, 267
 obix_server_invoke, 267
 obix_server_read, 268
 obix_server_shutdown, 268
 obix_server_write, 268
server/doctree.c, 230
server/doctree.h, 232
server/obix_fcgi.c, 233
server/obix_fcgi.h, 240

server/post_handler.c, 243
server/post_handler.h, 250
server/request.h, 252
server/response.c, 252
server/response.h, 256
server/server.c, 260
server/server.h, 265
server/watch.c, 268
server/watch.h, 279
server/xml_storage.c, 286
server/xml_storage.h, 299
SERVER_CONNECTION
 sensor_floor_driver.c, 56
serverUri
 .Http_Connection, 18
serverUriLength
 .Http_Connection, 18
setPrintf
 log_utils.c, 164
setSyslog
 log_utils.c, 164
setWatchLeaseTime
 obix_http.c, 129
setWatchPollWaitTime
 obix_http.c, 129
setWatchTimeParam
 obix_http.c, 129
signUpUri
 .Http_Connection, 18
size
 _Table, 32
status
 _oBIX_BatchResult, 27
stopTask
 ptask.c, 202
strncpy
 obix_batch.c, 79

Table
 table.h, 141
table.c
 table_create, 138
 table_extend, 138
 table_free, 139
 table_get, 139
 table_getKeys, 139
 table_getValues, 139
 table_put, 139
 table_remove, 139
table.h
 Table, 141
 table_create, 141
 table_free, 141
 table_get, 141
table_getKeys, 141
table_getValues, 141
table_put, 141
table_remove, 142
table_create
 table.c, 138
 table.h, 141
table_extend
 table.c, 138
table_free
 table.c, 139
 table.h, 141
table_get
 table.c, 139
 table.h, 141
table_getKeys
 table.c, 139
 table.h, 141
table_getValues
 table.c, 139
 table.h, 141
table_put
 table.c, 139
 table.h, 141
table_remove
 table.c, 139
 table.h, 142
Target
 sensor_floor_driver.c, 56
target_get
 sensor_floor_driver.c, 60
TARGET_REMOVE_TIMEOUT
 sensor_floor_driver.c, 56
target_sendUpdate
 sensor_floor_driver.c, 60
target_setX
 sensor_floor_driver.c, 60
target_setY
 sensor_floor_driver.c, 61
targetRemoveTask
 sensor_floor_driver.c, 61
targetsListener
 sensor_floor_driver.c, 61
task
 _Periodic_Task, 30
Task_Thread
 ptask.h, 206
taskDeleteWatch
 watch.c, 277
taskExecuted
 _Task_Thread, 37
taskList
 _Task_Thread, 37
taskListMutex

_Task_Thread, 37
taskListUpdated
 _Task_Thread, 37
taskStopThread
 test_ptask.c, 317
test/test_client.c, 307
test/test_client.h, 310
test/test_common.c, 311
test/test_common.h, 313
test/test_main.c, 314
test/test_main.h, 316
test/test_ptask.c, 316
test/test_ptask.h, 319
test/test_server.c, 320
test/test_server.h, 326
test/test_table.c, 326
test/test_table.h, 328
test_client
 test_client.c, 309
 test_client.h, 311
test_client.c
 REQUEST_HTTP_GET, 308
 REQUEST_HTTP_POST, 308
 REQUEST_HTTP_PUT, 309
 test_client, 309
 test_client_byHands, 309
 testBatch, 309
 testConnectionAndDevices, 309
 testCurlExt, 309
 testCurlExtRequest, 310
 testObixLoadConfigFile, 310
test_client.h
 test_client, 311
 test_client_byHands, 311
test_client_byHands
 test_client.c, 309
 test_client.h, 311
test_common
 test_common.c, 312
 test_common.h, 314
test_common.c
 test_common, 312
 test_common_byHands, 312
 testLog, 312
 testObix_relttime_fromLong, 312
 testObix_relttime_parse, 313
 testObix_relttime_parseToLong, 313
 testObixUtils, 313
test_common.h
 test_common, 314
 test_common_byHands, 314
test_common_byHands
 test_common.c, 312
 test_common.h, 314
test_main.c
 main, 315
 printTestResult, 315
 runAutomaticTests, 315
 runManualTests, 315
test_main.h
 printTestResult, 316
test_ptask
 test_ptask.c, 317
 test_ptask.h, 319
test_ptask.c
 taskStopThread, 317
 test_ptask, 317
 test_ptask_byHands, 317
 testGetClosestTask, 318
 testPeriodicTask, 318
 testPtaskReschedule, 318
 testTask, 318
test_ptask.h
 test_ptask, 319
 test_ptask_byHands, 319
test_ptask_byHands
 test_ptask.c, 317
 test_ptask.h, 319
test_server
 test_server.c, 322
 test_server.h, 326
test_server.c
 _lastResponse, 325
 _responseIsSent, 325
 _responseMutex, 325
 _responseReceived, 325
 checkResponse, 321
 dummyResponseListener, 321
 findInResponse, 321
 isResponseSent, 321
 obix_client_testListener, 322
 printResponse, 322
 test_server, 322
 testDelete, 322
 testGenerateResponse, 322
 testPostHandler, 323
 testPutHandler, 323
 testResponse_setRightUri, 323
 testSearch, 323
 testSignUp, 323
 testSignUpHelper, 323
 testWatch, 324
 testWatchPollChanges, 324
 testWatchRemove, 324
 testWriteToDatabase, 324
 waitForResponse, 325
test_server.h
 test_server, 326

test_table
 test_table.c, 327
 test_table.h, 328
test_table.c
 test_table, 327
 testTableGet, 327
test_table.h
 test_table, 328
testBatch
 test_client.c, 309
testConnectionAndDevices
 test_client.c, 309
testCurlExt
 test_client.c, 309
testCurlExtRequest
 test_client.c, 310
testCycle
 sensor_floor_driver.c, 62
testDelete
 test_server.c, 322
testGenerateResponse
 test_server.c, 322
testGetClosestTask
 test_ptask.c, 318
testLog
 test_common.c, 312
testObix_relttime_fromLong
 test_common.c, 312
testObix_relttime_parse
 test_common.c, 313
testObix_relttime_parseToLong
 test_common.c, 313
testObixLoadConfigFile
 test_client.c, 310
testObixUtils
 test_common.c, 313
testPeriodicTask
 test_ptask.c, 318
testPostHandler
 test_server.c, 323
testPtTaskReschedule
 test_ptask.c, 318
testPutHandler
 test_server.c, 323
testResponse_setRightUri
 test_server.c, 323
testSearch
 test_server.c, 323
testSignUp
 test_server.c, 323
testSignUpHelper
 test_server.c, 323
testTableGet
 test_table.c, 327

testTask
 test_ptask.c, 318
testWatch
 test_server.c, 324
testWatchPollChanges
 test_server.c, 324
testWatchRemove
 test_server.c, 324
testWriteToDatabase
 test_server.c, 324
thread
 _Task_Thread, 37
threadCycle
 ptask.c, 203
timerTask
 example_timer.c, 51
timespec_add
 ptask.c, 203
timespec_cmp
 ptask.c, 203
timespec_copy
 ptask.c, 203
timespec_fromMillis
 ptask.c, 203
TreeNode
 doctree.c, 231
TRUE
 bool.h, 143
type
 _Connection, 12
 _oBIX_BatchCmd, 26
unregisterDevice
 _Comm_Stack, 8
unregisterListener
 _Comm_Stack, 8
updated
 oBIX_Watch_Item, 43
updateMetaWatch
 server.c, 264
uri
 _Http_Device, 21
 _Target, 34
 oBIX_Watch_Item, 43
 PollTaskParams, 45
 Response, 47
uriLength
 _Http_Device, 21
value
 _oBIX_BatchResult, 28
values
 _Table, 33

waitForResponse
 test_server.c, 325

watch
 PollTaskParams, 45

watch.c
 _threadLease, 277
 _threadLongPoll, 277
 _watches, 277
 _watchesCount, 277
 findWatchItem, 270
 generateWatchUri, 270
 getLeaseTime, 271
 MAX_WATCH_COUNT, 278
 OBIX_META_ATTR_WATCH_TEMPLATE,
 278
 OBIX_META_WATCH_UPDATED_NO, 278
 OBIX_META_WATCH_UPDATED_YES,
 278
 OBIX_WATCHLEASE_NO_CHANGE, 278
 obixWatch_appendWatchItem, 271
 obixWatch_create, 271
 obixWatch_createWatchItem, 272
 obixWatch_delete, 272
 obixWatch_deleteHelper, 273
 obixWatch_deleteWatchItem, 273
 obixWatch_dispose, 273
 obixWatch_free, 273
 obixWatch_get, 273
 obixWatch_getByUri, 273
 obixWatch_getUri, 274
 obixWatch_getWatchItem, 274
 obixWatch_holdPoll, 274
 obixWatch_init, 274
 obixWatch_isLongPoll, 274
 obixWatch_isWatchUri, 275
 obixWatch_longPollTask, 275
 obixWatch_notifyPollTask, 275
 obixWatch_processTimeUpdates, 275
 obixWatch_resetLeaseTimer, 275
 obixWatch_updateMeta, 276
 obixWatchItem_free, 276
 obixWatchItem_freeRecursive, 276
 obixWatchItem_isUpdated, 276
 obixWatchItem_setUpdated, 277
 taskDeleteWatch, 277
 WATCH_URI_PREFIX_LENGTH, 278
 WATCH_URI_TEMPLATE, 278

watch.h
 OBIX_META_WATCH_UPDATED_NO, 285
 OBIX_META_WATCH_UPDATED_YES,
 285
 OBIX_WATCHLEASE_NO_CHANGE, 285
 obixWatch_appendWatchItem, 281
 obixWatch_create, 281

obixWatch_createWatchItem, 281
obixWatch_delete, 282
obixWatch_deleteWatchItem, 282
obixWatch_dispose, 282
obixWatch_get, 283
obixWatch_getByUri, 283
obixWatch_getUri, 283
obixWatch_getWatchItem, 283
obixWatch_holdPoll, 283
obixWatch_init, 284
obixWatch_isLongPoll, 284
obixWatch_isWatchUri, 284
obixWatch_pollHandler, 280
obixWatch_processTimeUpdates, 284
obixWatch_resetLeaseTimer, 284
obixWatch_updateMeta, 284
obixWatchItem_isUpdated, 285
obixWatchItem_setUpdated, 285

WATCH_OUT_POSTFIX
 post_handler.c, 245

WATCH_OUT_PREFIX
 post_handler.c, 245

WATCH_URI_PREFIX_LENGTH
 watch.c, 278

WATCH_URI_TEMPLATE
 watch.c, 278

watchAddUri
 _Http_Connection, 18

watchDeleteUri
 _Http_Connection, 18

watchLease
 _Http_Connection, 18

watchMakeUri
 _Http_Connection, 19

watchMutex
 _Http_Connection, 19

watchPollChangesFullUri
 _Http_Connection, 19

watchPollTask
 obix_http.c, 129

watchPollTaskId
 _Http_Connection, 19

watchRemoveUri
 _Http_Connection, 19

watchTable
 _Http_Connection, 19

writeValue
 _Comm_Stack, 9
 obix_http.c, 129

x
 _Target, 34

xml_config.c
 config_finishInit, 212

config_getChildTag, 212
config_getChildTagValue, 213
config_getResFullPath, 213
config_getTagAttrBoolValue, 214
config_getTagAttributeValue, 214
config_getTagAttrIntValue, 215
config_getTagAttrLongValue, 215
config_loadFile, 216
config_log, 216
config_setResourceDir, 217
CT_CONFIG, 217
CT_LOG, 217
CT_LOG_LEVEL, 217
CT_LOG_USE_SYSLOG, 218
CTA_LOGFacility, 218
CTA_VALUE, 218
CTAV_LOGFacility_DAEMON, 218
CTAV_LOGFacility_LOCAL0, 218
CTAV_LOGFacility_USER, 218
CTAV_LOG_LEVEL_DEBUG, 219
CTAV_LOG_LEVEL_ERROR, 219
CTAV_LOG_LEVEL_NO, 219
CTAV_LOG_LEVEL_WARNING, 219
resourceFolder, 219

xml_config.h
config_finishInit, 222
config_getChildTag, 223
config_getChildTagValue, 223
config_getResFullPath, 223
config_getTagAttrBoolValue, 224
config_getTagAttributeValue, 224
config_getTagAttrIntValue, 225
config_getTagAttrLongValue, 225
config_loadFile, 226
config_log, 226
config_setResourceDir, 227
CT_CONFIG, 227
CT_LOG, 227
CT_LOG_LEVEL, 227
CT_LOG_USE_SYSLOG, 228
CTA_LOGFacility, 228
CTA_VALUE, 228
CTAV_LOGFacility_DAEMON, 228
CTAV_LOGFacility_LOCAL0, 228
CTAV_LOGFacility_USER, 229
CTAV_LOG_LEVEL_DEBUG, 229
CTAV_LOG_LEVEL_ERROR, 229
CTAV_LOG_LEVEL_NO, 229
CTAV_LOG_LEVEL_WARNING, 229

XML_FALSE
obix_utils.c, 182
obix_utils.h, 195

XML_HEADER
obix_fcgi.c, 240

xml_storage.c
_serverAddress, 297
_serverAddressLength, 297
_storage, 297
checkNode, 288
compare_uri, 288
DEVICE_URI_PREFIX, 297
DEVICE_URI_PREFIX_LENGTH, 297
getLastSlashPosition, 289
getMetaInfo, 289
getNodeByHref, 289
getNodeByHrefRecursive, 290
insertServerAddress, 290
OBIX_META, 297
OBIX_STORAGE_FILES, 298
OBIX_STORAGE_FILES_COUNT, 298
OBIX_SYS_ERROR_STUB, 298
OBIX_SYS_WATCH_OUT_STUB, 298
OBIX_SYS_WATCH_STUB, 298
printXMLContents, 290
removeMetaInfo, 290
xmldb_compareServerAddr, 291
xmldb_delete, 291
xmldb_deleteMeta, 291
xmldb_dispose, 291
xmldb_get, 291
xmldb_getDOM, 292
xmldb_getDump, 292
xmldb_getFullUri, 292
xmldb_getObixSysObject, 293
xmldb_getServerAddress, 293
xmldb_getServerAddressLength, 293
xmldb_init, 293
xmldb_loadFile, 293
xmldb_printDump, 294
xmldb_put, 294
xmldb_putDOM, 295
xmldb_putDOMHelper, 295
xmldb_putHelper, 295
xmldb_putMeta, 296
xmldb_updateDOM, 296
xmldb_updateMeta, 296

xml_storage.h
getMetaInfo, 300
OBIX_META, 307
OBIX_SYS_ERROR_STUB, 307
OBIX_SYS_WATCH_OUT_STUB, 307
OBIX_SYS_WATCH_STUB, 307
removeMetaInfo, 301
xmldb_compareServerAddr, 301
xmldb_delete, 301
xmldb_deleteMeta, 301
xmldb_dispose, 301
xmldb_get, 302

xmlDb_getDOM, 302
xmlDb_getDump, 302
xmlDb_getFullUri, 302
xmlDb_getObixSysObject, 303
xmlDb_getServerAddress, 303
xmlDb_getServerAddressLength, 303
xmlDb_init, 303
xmlDb_loadFile, 304
xmlDb_printDump, 304
xmlDb_put, 304
xmlDb_putDOM, 305
xmlDb_putMeta, 305
xmlDb_updateDOM, 306
xmlDb_updateMeta, 306
XML_TRUE
 obix_utils.c, 182
 obix_utils.h, 195
xmlDb_compareServerAddr
 xml_storage.c, 291
 xml_storage.h, 301
xmlDb_delete
 xml_storage.c, 291
 xml_storage.h, 301
xmlDb_deleteMeta
 xml_storage.c, 291
 xml_storage.h, 301
xmlDb_dispose
 xml_storage.c, 291
 xml_storage.h, 301
xmlDb_get
 xml_storage.c, 291
 xml_storage.h, 302
xmlDb_getDOM
 xml_storage.c, 292
 xml_storage.h, 302
xmlDb_getDump
 xml_storage.c, 292
 xml_storage.h, 302
xmlDb_getFullUri
 xml_storage.c, 292
 xml_storage.h, 302
xmlDb_getObixSysObject
 xml_storage.c, 293
 xml_storage.h, 303
xmlDb_getServerAddress
 xml_storage.c, 293
 xml_storage.h, 303
xmlDb_getServerAddressLength
 xml_storage.c, 293
 xml_storage.h, 303
xmlDb_init
 xml_storage.c, 293
 xml_storage.h, 303
xmlDb_loadFile

xml_storage.c, 293
xml_storage.h, 304
xmlDb_printDump
 xml_storage.c, 294
 xml_storage.h, 304
xmlDb_put
 xml_storage.c, 294
 xml_storage.h, 304
xmlDb_putDOM
 xml_storage.c, 295
 xml_storage.h, 305
xmlDb_putDOMHelper
 xml_storage.c, 295
xmlDb_putHelper
 xml_storage.c, 295
xmlDb_putMeta
 xml_storage.c, 296
 xml_storage.h, 305
xmlDb_updateDOM
 xml_storage.c, 296
 xml_storage.h, 306
xmlDb_updateMeta
 xml_storage.c, 296
 xml_storage.h, 306

y
 _Target, 35