

libcot
0:0:0

Generated by Doxygen 1.5.6

Tue Jun 9 18:47:42 2009

Contents

1	Main Page	1
2	Todo List	2
3	File Index	3
3.1	File List	3
4	File Documentation	3
4.1	ixml_ext.h File Reference	3
4.1.1	Detailed Description	3
4.1.2	Function Documentation	4
4.2	lwl_ext.h File Reference	6
4.2.1	Detailed Description	7
4.2.2	Define Documentation	7
4.3	obix_client.h File Reference	7
4.3.1	Detailed Description	8
4.3.2	Typedef Documentation	8
4.3.3	Enumeration Type Documentation	9
4.3.4	Function Documentation	9
4.4	obix_utils.h File Reference	13
4.4.1	Detailed Description	14
4.5	ptask.h File Reference	14
4.5.1	Detailed Description	14
4.5.2	Define Documentation	14
4.5.3	Typedef Documentation	15
4.5.4	Function Documentation	15
4.6	xml_config.h File Reference	16
4.6.1	Detailed Description	17
4.6.2	Function Documentation	17

1 Main Page

The oBIX Client Library can be used by device drivers to post device data to oBIX server and monitor value updates. In order to register new device, server should support signUp feature which is not in the oBIX specification.

Usage:

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

```
gcc -I<cot_headers> -L<cot_lib> -lcot-client <sources>
```

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.

The typical usage of library (see example at *example_timer.c*):

- Include [obix_client.h](#) header.
- Initialize library during driver 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 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, register listener for these values by calling [obix_registerListener\(\)](#). Library start automatically polling changes of subscribed values and calls corresponding [obix_update_listener\(\)](#) when receives an updated value.
- When some update of device state is received by a driver, post 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, turned off, etc.).
- Call [obix_dispose\(\)](#) when device driver is going down in order to close all connections and release resources reserved for communication with oBIX server(s).

Author:

Andrey Litvinov

2 Todo List

Global [ixmlElement_freeOwnerDocument](#) implement also [ixmlNode_freeOwnerDocument\(\)](#).

Global [OBIX_ERR_LIMIT_REACHED](#) Remove this error and enlarge arrays when needed.

Global [obix_loadConfig](#) Update description of [obix_loadConfig\(\)](#) after adding log header to the library.

3 File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

ixml_ext.h	3
lwl_ext.h	6
obix_client.h	7
obix_utils.h	13
ptask.h	14
xml_config.h (Declares configuration constants and structs)	16

4 File Documentation

4.1 [ixml_ext.h](#) File Reference

```
#include <upnp/ixml.h>
```

Functions

- `IXML_Element *` [ixmlDocument_getElementByAttrValue](#) (`IXML_Document *doc`, `const char *attrName`, `const char *attrValue`)
- `int` [ixmlElement_setAttributeWithLog](#) (`IXML_Element *element`, `const char *attrName`, `const char *attrValue`)
- `int` [ixmlElement_removeAttributeWithLog](#) (`IXML_Element *element`, `const char *attrName`)
- `IXML_Element *` [ixmlElement_cloneWithLog](#) (`IXML_Element *source`)
- `void` [ixmlElement_freeOwnerDocument](#) (`IXML_Element *element`)
- `int` [ixmlElement_copyAttributeWithLog](#) (`IXML_Element *source`, `IXML_Element *target`, `const char *attrName`, `BOOL obligatory`)
- `IXML_Element *` [ixmlAttr_getOwnerElement](#) (`IXML_Attr *attr`)

XML node types conversion @{

- `IXML_Element *` [ixmlNode_convertToElement](#) (`IXML_Node *node`)
- `IXML_Attr *` [ixmlNode_convertToAttr](#) (`IXML_Node *node`)
- `IXML_Node *` [ixmlElement_getNode](#) (`IXML_Element *element`)
- `IXML_Node *` [ixmlDocument_getNode](#) (`IXML_Document *doc`)
- `IXML_Node *` [ixmlAttr_getNode](#) (`IXML_Attr *attr`)

4.1.1 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](#).

4.1.2 Function Documentation

4.1.2.1 IXML_Node* ixmlAttr_getNode (IXML_Attr * *attr*)

Returns node which represents provided attribute.

4.1.2.2 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.

4.1.2.3 IXML_Node* ixmlDocument_getNode (IXML_Document * *doc*)

Returns node which represents provided document.

4.1.2.4 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.

Note:

Don't forget to free owner document of the clone after usage.

See also:

ixmlNode_getOwnerDocument()

Parameters:

source Element to be copied.

Returns:

NULL on error, otherwise a pointer to the new copy of the source element.

4.1.2.5 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.

Todo

implement also `ixmlNode_freeOwnerDocument()`.

Parameters:

element Element which should be freed with it's owner document.

4.1.2.6 IXML_Node* ixmlElement_getNode (IXML_Element * *element*)

Returns node which represents provided element.

4.1.2.7 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.

Parameters:

attrName Name of the attribute to be removed.

Returns:

0 on success or 1 on error.

4.1.2.8 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 on error.

Parameters:

attrName Name of the attribute to be added.

attrValue Value of the attribute.

Returns:

0 on success or 1 on error.

4.1.2.9 IXML_Attr* ixmlNode_convertToAttr (IXML_Node * *node*)

Converts node to the attribute.

Parameters:

node to be converted.

Returns:

NULL if node is not an attribute.

4.1.2.10 IXML_Element* ixmlNode_convertToElement (IXML_Node * node)

Converts node to the element.

Parameters:

node to be converted.

Returns:

NULL if node is not an element (i.e. tag).

4.2 lwl_ext.h File Reference

```
#include "ixml_ext.h"
```

Defines

Logging utilities@{

- **#define log_debug**(fmt,...)
- **#define log_warning**(fmt,...)
- **#define log_error**(fmt,...)

Typedefs

- typedef void(* **log_function**)(char *fmt,...)

Functions

- int **log_config** (IXML_Element *configTag)
- void **log_dispose** ()

Variables

- const char * **CT_LOG**
- const char * **CT_LOG_LEVEL**
- const char * **CTAV_LOG_LEVEL_DEBUG**
- const char * **CTAV_LOG_LEVEL_WARNING**
- const char * **CTAV_LOG_LEVEL_ERROR**
- const char * **CTAV_LOG_LEVEL_NO**
- const char * **CT_LOG_FILE**
- const char * **CT_LOG_FORMAT**
- const char * **CTA_LOG_PREFIX**
- const char * **CTA_LOG_DATE**
- const char * **CTA_LOG_TIME**
- const char * **CTA_LOG_LOCALE**
- const char * **CTA_LOG_PRIORITY**
- const char * **CT_LOG_NO_FLUSH**
- log_function **log_debugHandler**
- log_function **log_warningHandler**
- log_function **log_errorHandler**

4.2.1 Detailed Description

Contains definitions of logging tools.

Log system is based on *liblwl* library.

Author:

Andrey Litvinov

Version:

1.0

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

4.2.2 Define Documentation

4.2.2.1 #define log_debug(fmt, ...)

Value:

```
(*log_debugHandler) ("%s(%d): " fmt, __FILE__, \
__LINE__,
```

Definition at line 39 of file [lwl_ext.h](#).

4.2.2.2 #define log_error(fmt, ...)

Value:

```
(*log_errorHandler) ("%s(%d): " fmt, __FILE__, \
__LINE__,
```

Definition at line 45 of file [lwl_ext.h](#).

4.2.2.3 #define log_warning(fmt, ...)

Value:

```
(*log_warningHandler) ("%s(%d): " fmt, __FILE__, \

```

Definition at line 42 of file [lwl_ext.h](#).

4.3 obix_client.h File Reference

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

Typedefs

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

Enumerations

- 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 }
- 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](#) }

Functions

- int [obix_loadConfigFile](#) (const char *fileName)
- int [obix_loadConfig](#) (IXML_Element *config)
- int [obix_dispose](#) ()
- int [obix_openConnection](#) (int connectionId)
- int [obix_closeConnection](#) (int connectionId)
- int [obix_registerDevice](#) (int connectionId, const char *obixData)
- int [obix_unregisterDevice](#) (int connectionId, int deviceId)
- int [obix_writeValue](#) (int connectionId, int deviceId, const char *paramUri, const char *newValue, [OBIX_DATA_TYPE](#) dataType)
- int [obix_registerListener](#) (int connectionId, int deviceId, const char *paramUri, [obix_update_listener](#) listener)
- int [obix_unregisterListener](#) (int connectionId, int deviceId, int listenerId)

4.3.1 Detailed Description

Contains definitions for oBIX client library.

Author:

Andrey Litvinov

Version:

1.0.0

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

4.3.2 Typedef Documentation

4.3.2.1 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 121 of file obix_client.h.

4.3.3 Enumeration Type Documentation**4.3.3.1 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.

Definition at line 68 of file obix_client.h.

4.3.4 Function Documentation**4.3.4.1 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.

4.3.4.2 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.

4.3.4.3 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. Unconfigured log writes all messages to *stdout*.

Todo

Update description of [obix_loadConfig\(\)](#) after adding log header to the library.

Parameters:

config DOM structure representing a configuration XML.

Returns:

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

4.3.4.4 int obix_loadConfigFile (const char * *fileName*)

Initializes library and loads connection setting from XML file. Also sets up the logging system of the library.

Parameters:

fileName Name of the configuration file.

Returns:

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

4.3.4.5 int obix_openConnection (int *connectionId*)

Opens connection with 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.

4.3.4.6 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 should have a valid URI, 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 can specify *href* attribute but the oBIX server can modify it (for instance, add prefix of the device storage), thus URI can't be used to refer to the device record. Use device ID instead.

Parameters:

connectionId ID of the connection which should be used.

obixData oBIX object representing the new device.

Returns:

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

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

Registers listener for device parameter updates.

Overwrites existing listener if is called twice.

This method can be also used to subscribe for the updates of any 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 changing 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.

Returns:

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

4.3.4.8 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.

4.3.4.9 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.

4.3.4.10 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. 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 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.

dataType Type of data which is written to the server.

Returns:

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

4.4 obix_utils.h File Reference

Enumerations

- enum **RELTIME_FORMAT** {
 RELTIME_MILLIS, **RELTIME_SEC**, **RELTIME_MIN**, **RELTIME_HOUR**,
 RELTIME_DAY, **RELTIME_MONTH**, **RELTIME_YEAR** }

Functions

- int **obix_reftime_parseToLong** (const char *str, long *period)
- char * **obix_reftime_fromLong** (long period, RELTIME_FORMAT)

Variables

- const char * **OBIX_HREF_ERR_BAD_URI**
- const char * **OBIX_HREF_ERR_UNSUPPORTED**
- const char * **OBIX_HREF_ERR_PERMISSION**
- const char * **OBIX_OBJ**
- const char * **OBIX_OBJ_REF**
- const char * **OBIX_OBJ_OP**
- const char * **OBIX_OBJ_LIST**
- const char * **OBIX_OBJ_ERR**
- const char * **OBIX_OBJ_BOOL**
- const char * **OBIX_OBJ_INT**
- const char * **OBIX_OBJ_REAL**
- const char * **OBIX_OBJ_STR**
- const char * **OBIX_OBJ_ENUM**
- const char * **OBIX_OBJ_ABSTIME**
- const char * **OBIX_OBJ_RELTIME**
- const char * **OBIX_OBJ_URI**
- const char * **OBIX_OBJ_FEED**
- const char * **OBIX_NAME_SIGN_UP**
- const char * **OBIX_NAME_WATCH_SERVICE**
- const char * **OBIX_NAME_WATCH_SERVICE_MAKE**
- const char * **OBIX_NAME_WATCH_ADD**
- const char * **OBIX_NAME_WATCH_REMOVE**
- const char * **OBIX_NAME_WATCH_POLLCHANGES**
- const char * **OBIX_NAME_WATCH_POLLREFRESH**
- const char * **OBIX_NAME_WATCH_DELETE**
- const char * **OBIX_NAME_WATCH_LEASE**
- const char * **OBIX_OBJ_ERR_TEMPLATE**
- const char * **OBIX_OBJ_NULL_TEMPLATE**
- const char * **OBIX_ATTR_IS**
- const char * **OBIX_ATTR_NAME**
- const char * **OBIX_ATTR_HREF**
- const char * **OBIX_ATTR_VAL**
- const char * **OBIX_ATTR_WRITABLE**
- const char * **OBIX_ATTR_DISPLAY**
- const char * **OBIX_ATTR_DISPLAY_NAME**
- const char * **XML_TRUE**
- const char * **XML_FALSE**

4.4.1 Detailed Description

Contains definitions of oBIX constants: names of objects, contracts, facets, etc. (<http://obix.org/>)

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

4.5 ptask.h File Reference

Defines

- #define [EXECUTE_INDEFINITE](#) -1

Typedefs

- typedef void(* [periodic_task](#))(void *arg)
- typedef struct _Task_Thread [Task_Thread](#)

Functions

- [Task_Thread](#) * [ptask_init](#) ()
- int [ptask_dispose](#) ([Task_Thread](#) *thread)
- int [ptask_schedule](#) ([Task_Thread](#) *thread, [periodic_task](#) task, void *arg, long period, int executeTimes)
- int [ptask_reschedule](#) ([Task_Thread](#) *thread, int taskId, long period, int executeTimes)
- int [ptask_reset](#) ([Task_Thread](#) *thread, int taskId)
- int [ptask_cancel](#) ([Task_Thread](#) *thread, int taskId)

4.5.1 Detailed Description

Contains interface definition of the Periodic Task.

Periodic Task utility can be used to schedule some function to be invoked in a separate thread periodically after defined delay. A function can be scheduled to be invoked either a defined number of times (starting from 1) or until it is canceled.

Author:

Andrey Litvinov

Version:

1.0

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

4.5.2 Define Documentation

4.5.2.1 #define EXECUTE_INDEFINITE -1

Specifies that the task should be executed an indefinite number of times (until [ptask_cancel\(\)](#) is called).

Definition at line 21 of file [ptask.h](#).

4.5.3 Typedef Documentation

4.5.3.1 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 28 of file ptask.h.

4.5.3.2 typedef struct _Task_Thread Task_Thread

Represents a separate thread which can be used to schedule tasks.

Definition at line 33 of file ptask.h.

4.5.4 Function Documentation

4.5.4.1 int ptask_cancel (Task_Thread * thread, int taskId)

Removes task from the scheduled list.

Parameters:

thread Thread in which task is scheduled.

id ID of the task to be removed.

Returns:

- 0 on success;
- -1 if task with provided ID is not found.

4.5.4.2 int ptask_dispose (Task_Thread * thread)

Releases resources allocated for the provided [Task_Thread](#) instance. All scheduled tasks are canceled.

Parameters:

thread Pointer to the [Task_Thread](#) to be freed.

Returns:

0 on success, negative error code otherwise.

4.5.4.3 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.

4.5.4.4 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 then the task is executed until [ptask_cancel\(\)](#) with corresponding task ID is called.

Returns:

- ≥ 0 - ID of the scheduled task. Can be used to cancel the task.
- < 0 - Error code.

4.6 xml_config.h File Reference

Declares configuration constants and structs.

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

Functions

- IXML_Element * [config_loadFile](#) (const char *filename)
Tag attribute value 'true' (CTAV - Config Tag Attribute Value).
- IXML_Element * [config_getChildTag](#) (IXML_Element *conf, const char *tagName, BOOL obligatory)
- const char * [config_getChildTagValue](#) (IXML_Element *conf, const char *tagName, BOOL obligatory)
- const char * [config_getTagAttributeValue](#) (IXML_Element *tag, const char *attrName, BOOL obligatory)
- int [config_getTagIntAttrValue](#) (IXML_Element *tag, const char *attrName, BOOL obligatory, int defaultValue)
- long [config_getTagLongAttrValue](#) (IXML_Element *tag, const char *attrName, BOOL obligatory, long defaultValue)
- int [config_getTagBoolAttrValue](#) (IXML_Element *tag, const char *attrName, BOOL obligatory)
- void [config_finishInit](#) ()
- void [config_dispose](#) ()
- char * [config_getResFullPath](#) (const char *filename)
- void [config_setResourceDir](#) (char *path)

Variables

Config file parameter names

See more about parameters at the [OBIX_CONFIG_FILE](#).

- const char * [CT_CONFIG](#)
Main tag name (CT - Config Tag).
- const char * [CTA_VALUE](#)
Tag attribute 'value' (CTA - Config Tag Attribute).

4.6.1 Detailed Description

Declares configuration constants and structs.

Defines the names of configuration parameters as well as their possible values. These parameters can be set by user in `OBIX_CONFIG_FILE`.

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

4.6.2 Function Documentation

4.6.2.1 char* config_getResFullPath (const char *filename)

Returns the address of the resource file by adding required path to the filename. **Note:** Do not forget to release memory after using.

Returns:

address to the resource file.

4.6.2.2 int config_getTagBoolAttrValue (IXML_Element * tag, const char * attrName, BOOL obligatory)

Returns:

- 0 - False;
- 1 - True;
- -1 - Error.

4.6.2.3 int config_getTagIntAttrValue (IXML_Element * tag, const char * attrName, BOOL obligatory, int defaultValue)

Returns:

- ≥ 0 - Parsed integer attribute value;
- -1 - Error code.

4.6.2.4 long config_getTagLongAttrValue (IXML_Element * tag, const char * attrName, BOOL obligatory, long defaultValue)

Returns:

- ≥ 0 - Parsed long integer attribute value;
- -1 - Error code.

4.6.2.5 IXML_Element* config_loadFile (const char **filename*)

Tag attribute value 'true' (CTAV - Config Tag Attribute Value).

Opens config file and checks its structure

Returns:

pointer to the config xml node.

Index

config_getResFullPath
 xml_config.h, 16
config_getTagBoolAttrValue
 xml_config.h, 16
config_getTagIntAttrValue
 xml_config.h, 17
config_getTagLongAttrValue
 xml_config.h, 17
config_loadFile
 xml_config.h, 17

EXECUTE_INDEFINITE
 ptask.h, 14

ixml_ext.h, 2
 ixmlAttr_getNode, 3
 ixmlDocument_getElementByAttrValue, 3
 ixmlDocument_getNode, 3
 ixmlElement_cloneWithLog, 3
 ixmlElement_freeOwnerDocument, 4
 ixmlElement_getNode, 4
 ixmlElement_removeAttributeWithLog, 4
 ixmlElement_setAttributeWithLog, 4
 ixmlNode_convertToAttr, 4
 ixmlNode_convertToElement, 5
ixmlAttr_getNode
 ixml_ext.h, 3
ixmlDocument_getElementByAttrValue
 ixml_ext.h, 3
ixmlDocument_getNode
 ixml_ext.h, 3
ixmlElement_cloneWithLog
 ixml_ext.h, 3
ixmlElement_freeOwnerDocument
 ixml_ext.h, 4
ixmlElement_getNode
 ixml_ext.h, 4
ixmlElement_removeAttributeWithLog
 ixml_ext.h, 4
ixmlElement_setAttributeWithLog
 ixml_ext.h, 4
ixmlNode_convertToAttr
 ixml_ext.h, 4
ixmlNode_convertToElement
 ixml_ext.h, 5

log_debug
 lwl_ext.h, 6
log_error
 lwl_ext.h, 6
log_warning
 lwl_ext.h, 6
lwl_ext.h, 5
log_debug, 6
log_error, 6
log_warning, 6

obix_client.h
 OBIX_ERR_BAD_CONNECTION, 8
 OBIX_ERR_HTTP_LIB, 8
 OBIX_ERR_INVALID_ARGUMENT, 8
 OBIX_ERR_INVALID_STATE, 8
 OBIX_ERR_LIMIT_REACHED, 8
 OBIX_ERR_NO_MEMORY, 8
 OBIX_ERR_UNKNOWN_BUG, 8
 OBIX_SUCCESS, 8
OBIX_ERR_BAD_CONNECTION
 obix_client.h, 8
OBIX_ERR_HTTP_LIB
 obix_client.h, 8
OBIX_ERR_INVALID_ARGUMENT
 obix_client.h, 8
OBIX_ERR_INVALID_STATE
 obix_client.h, 8
OBIX_ERR_LIMIT_REACHED
 obix_client.h, 8
OBIX_ERR_NO_MEMORY
 obix_client.h, 8
OBIX_ERR_UNKNOWN_BUG
 obix_client.h, 8
OBIX_SUCCESS
 obix_client.h, 8
obix_client.h, 7
 obix_closeConnection, 9
 obix_dispose, 9
 OBIX_ERRORCODE, 8
 obix_loadConfig, 9
 obix_loadConfigFile, 9
 obix_openConnection, 9
 obix_registerDevice, 10
 obix_registerListener, 10
 obix_unregisterDevice, 11
 obix_unregisterListener, 11
 obix_update_listener, 8
 obix_writeValue, 11
obix_closeConnection
 obix_client.h, 9
obix_dispose
 obix_client.h, 9
OBIX_ERRORCODE
 obix_client.h, 8
obix_loadConfig

- obix_client.h, [9](#)
- obix_loadConfigFile
 - obix_client.h, [9](#)
- obix_openConnection
 - obix_client.h, [9](#)
- obix_registerDevice
 - obix_client.h, [10](#)
- obix_registerListener
 - obix_client.h, [10](#)
- obix_unregisterDevice
 - obix_client.h, [11](#)
- obix_unregisterListener
 - obix_client.h, [11](#)
- obix_update_listener
 - obix_client.h, [8](#)
- obix_utils.h, [12](#)
- obix_writeValue
 - obix_client.h, [11](#)
- periodic_task
 - ptask.h, [14](#)
- ptask.h, [13](#)
 - EXECUTE_INDEFINITE, [14](#)
 - periodic_task, [14](#)
 - ptask_cancel, [14](#)
 - ptask_dispose, [15](#)
 - ptask_init, [15](#)
 - ptask_schedule, [15](#)
 - Task_Thread, [14](#)
- ptask_cancel
 - ptask.h, [14](#)
- ptask_dispose
 - ptask.h, [15](#)
- ptask_init
 - ptask.h, [15](#)
- ptask_schedule
 - ptask.h, [15](#)
- Task_Thread
 - ptask.h, [14](#)
- xml_config.h, [15](#)
 - config_getResFullPath, [16](#)
 - config_getTagBoolAttrValue, [16](#)
 - config_getTagIntAttrValue, [17](#)
 - config_getTagLongAttrValue, [17](#)
 - config_loadFile, [17](#)