



Xively Reference Manual

C API Reference

Draft

© Marvell International, 2009-2015

August 21, 2015 17:48
Generated by Doxygen 1.8.1.1

Contents

| | | |
|----------|--|----------|
| 1 | Data Structure Index | 1 |
| 1.1 | Data Structures | 1 |
| 2 | File Index | 3 |
| 2.1 | File List | 3 |
| 3 | Data Structure Documentation | 5 |
| 3.1 | xi_context_t Struct Reference | 5 |
| 3.1.1 | Field Documentation | 5 |
| 3.1.1.1 | protocol | 5 |
| 3.1.1.2 | feed_id | 5 |
| 3.1.1.3 | layer_chain | 5 |
| 3.1.1.4 | input | 5 |
| 3.2 | xi_datapoint_t Struct Reference | 6 |
| 3.2.1 | Detailed Description | 6 |
| 3.3 | xi_datapoint_value_t Union Reference | 6 |
| 3.4 | xi_feed_t Struct Reference | 6 |
| 3.4.1 | Detailed Description | 7 |
| 3.5 | xi_response_t Struct Reference | 7 |
| 3.6 | xi_timestamp_t Struct Reference | 7 |
| 4 | File Documentation | 9 |
| 4.1 | external/libxively/src/libxively/xi_err.h File Reference | 9 |
| 4.1.1 | Detailed Description | 9 |
| 4.1.2 | Function Documentation | 9 |
| 4.1.2.1 | xi_get_last_error | 9 |
| 4.1.2.2 | xi_set_err | 9 |
| 4.2 | external/libxively/src/libxively/xively.h File Reference | 9 |
| 4.2.1 | Function Documentation | 10 |
| 4.2.1.1 | xi_get_value_type | 10 |
| 4.2.1.2 | xi_set_value_i32 | 10 |
| 4.2.1.3 | xi_get_value_i32 | 10 |

| | | |
|----------|--|----|
| 4.2.1.4 | xi_value_pointer_i32 | 10 |
| 4.2.1.5 | xi_set_value_f32 | 10 |
| 4.2.1.6 | xi_get_value_f32 | 10 |
| 4.2.1.7 | xi_value_pointer_f32 | 11 |
| 4.2.1.8 | xi_set_value_str | 11 |
| 4.2.1.9 | xi_value_pointer_str | 11 |
| 4.2.1.10 | xi_set_network_timeout | 11 |
| 4.2.1.11 | xi_create_context | 11 |
| 4.2.1.12 | xi_delete_context | 11 |
| 4.2.1.13 | xi_datastream_delete | 12 |
| 4.2.1.14 | xi_datapoint_delete | 12 |
| 4.2.1.15 | xi_datapoint_delete_range | 12 |
| 4.2.2 | Enumeration Type Documentation | 12 |
| 4.2.2.1 | xi_protocol_t | 12 |
| 4.2.2.2 | http_header_type_t | 12 |
| 4.2.2.3 | xi_value_type_t | 13 |

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

| | | |
|--------------------------------------|--|---|
| xi_context_t | <i>The context structure</i> - it's the first argument for all functions that communicate with Xively API (i.e. not helpers or utilities) | 5 |
| xi_datapoint_t | <i>Xively datapoint structure</i> - it contains value and timestamp | 6 |
| xi_datapoint_value_t | The datapoint value union | 6 |
| xi_feed_t | <i>Xively feed structure</i> - it contains a fixed array of datastream | 6 |
| xi_response_t | <i>The response structure</i> - it's the return type for all functions that communicate with Xively API (i.e. not helpers or utilities) | 7 |
| xi_timestamp_t | The datapoint timestamp | 7 |

Confidential

Chapter 2

File Index

2.1 File List

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

| | |
|--|---|
| external/libxively/src/libxively/ xi_err.h | |
| Error handling (POSIX-like) | 9 |
| external/libxively/src/libxively/ xively.h | |
| Xively C library | 9 |

Confidential

Chapter 3

Data Structure Documentation

3.1 xi_context_t Struct Reference

The context structure - it's the first argument for all functions that communicate with Xively API (*i.e. not helpers or utilities*)

```
#include <xively.h>
```

Data Fields

- char * **api_key**
- [xi_protocol_t](#) protocol
- [xi_feed_id_t](#) feed_id
- [layer_chain_t](#) layer_chain
- void * input

3.1.1 Field Documentation

3.1.1.1 xi_protocol_t xi_context_t::protocol

Xively API key

3.1.1.2 xi_feed_id_t xi_context_t::feed_id

Xively protocol

3.1.1.3 layer_chain_t xi_context_t::layer_chain

Xively feed ID

3.1.1.4 void* xi_context_t::input

Xively reference of layers

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

- [external/libxively/src/libxively/xively.h](#)

3.2 xi_datapoint_t Struct Reference

Xively datapoint structure - it contains value and timestamp

```
#include <xively.h>
```

Data Fields

- [xi_datapoint_value_t](#) **value**
- [xi_value_type_t](#) **value_type**
- [xi_timestamp_t](#) **timestamp**

3.2.1 Detailed Description

Note

A zero-valued timestamp is used by most functions as a convention to opt for server-side timestamps.

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

- [external/libxively/src/libxively/xively.h](#)

3.3 xi_datapoint_value_t Union Reference

The datapoint value union.

```
#include <xively.h>
```

Data Fields

- [int32_t](#) **i32_value**
- [float](#) **f32_value**
- [char](#) **str_value** [XI_VALUE_STRING_MAX_SIZE]

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

- [external/libxively/src/libxively/xively.h](#)

3.4 xi_feed_t Struct Reference

Xively feed structure - it contains a fixed array of datastream

```
#include <xively.h>
```

Data Fields

- [xi_feed_id_t](#) **feed_id**
- [size_t](#) **datastream_count**
- [xi_datastream_t](#) **datastreams** [XI_MAX_DATASTREAMS]

3.4.1 Detailed Description

Note

The implementation is such that user will need to know in advance how many datastreams there can be, which should be sufficient for a real-world application. It's also undesired to have some devices create dozens of datastreams due to a bug.

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

- [external/libxively/src/libxively/xively.h](#)

3.5 xi_response_t Struct Reference

The response structure - it's the return type for all functions that communicate with Xively API (*i.e. not helpers or utilities*)

```
#include <xively.h>
```

Data Fields

- `http_response_t http`

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

- [external/libxively/src/libxively/xively.h](#)

3.6 xi_timestamp_t Struct Reference

The datapoint timestamp.

```
#include <xively.h>
```

Data Fields

- `xi_time_t timestamp`
- `xi_time_t micro`

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

- [external/libxively/src/libxively/xively.h](#)

Confidential

Chapter 4

File Documentation

4.1 external/libxively/src/libxively/xi_err.h File Reference

Error handling (POSIX-like)

4.1.1 Detailed Description

Author

Olgierd Humenczuk

- Every function should return a value
- There are special values (usually 0 or -1) which indicate occurrence of an error
- User can detect and lookup errors using declarations below

4.1.2 Function Documentation

4.1.2.1 `xi_err_t xi_get_last_error (void)`

Returns

The `xi_err_t` structure which can be converted to a string using `xi_get_error_string()` method.

Warning

It resets the last error value, so it's always a good idea to make a copy of it!

4.1.2.2 `void xi_set_err (xi_err_t e)`

Note

Current implementation used a global state variable (*errno*), which is not thread-safe. If thread-safety is required, than *errno* should be made thread-local.

4.2 external/libxively/src/libxively/xively.h File Reference

Xively C library.

4.2.1 Function Documentation

4.2.1.1 `xi_value_type_t xi_get_value_type (xi_datapoint_t * p)`

Example

```
switch( xi_get_value_type(dp) ) {
    case XI_VALUE_TYPE_I32:
        printf("Got int value: %i\n", xi_get_value_i32(dp));
        break;
    case XI_VALUE_TYPE_F32:
        printf("Got float value: %f\n", xi_get_value_f32(dp));
        break;
    case XI_VALUE_TYPE_STR:
        printf("Got a string: %s\n", xi_get_value_str(dp));
        break;
    default:
        printf("Unknown value type enumerator: %i", xi_get_value_type(dp));
        break;
}
```

4.2.1.2 `xi_datapoint_t* xi_set_value_i32 (xi_datapoint_t * dp, int32_t v)`

Returns

Pointer or 0 if an error occurred

4.2.1.3 `int32_t xi_get_value_i32 (xi_datapoint_t * p)`

Warning

Only use this when you are sure the value is set and is of correct type!

4.2.1.4 `int32_t* xi_value_pointer_i32 (xi_datapoint_t * p)`

Returns

A pointer or NULL if the type doesn't match

Example

```
int32_t *v = xi_value_pointer_i32(p);
if(v == NULL) {
    printf("Not an int32_t!\n");
} else {
    printf("v=%i\n", *v);
}
```

4.2.1.5 `xi_datapoint_t* xi_set_value_f32 (xi_datapoint_t * dp, float v)`

Returns

Pointer or 0 if an error occurred

4.2.1.6 `float xi_get_value_f32 (xi_datapoint_t * p)`

Warning

Only use this when you are sure the value is set and is of correct type!

4.2.1.7 float* xi_value_pointer_f32 (xi_datapoint_t * p)**Returns**

A pointer or NULL if the type doesn't match

Example

```
float *v = xi_value_pointer_f32(p);
if(v == NULL) {
    printf("Not a float!\n");
} else {
    printf("v=%f\n", *v);
}
```

4.2.1.8 xi_datapoint_t* xi_set_value_str (xi_datapoint_t * dp, const char * v)**Returns**

Pointer or 0 if an error occurred

4.2.1.9 char* xi_value_pointer_str (xi_datapoint_t * p)**Returns**

A pointer or NULL if the type doesn't match

Example

```
char *v = xi_value_pointer_str(p);
if(v == NULL) {
    printf("Not a string!\n");
} else {
    printf("v='%s'\n", *v);
}
```

4.2.1.10 void xi_set_network_timeout (uint32_t milliseconds)**Note**

The timeout is used by the communication layer to determine whenever it should treat the lag in a connection as an error, so if your device or your connection is slow, you can try to increase the timeout for network operations. It only affects the send/rcv operations it does not work with connect but that behaviour may differ between platforms and communication layer implementations.

4.2.1.11 xi_context_t* xi_create_context (xi_protocol_t protocol, const char * api_key, xi_feed_id_t feed_id)

The purpose of this function is to allocate memory and initialise the data structures needed in order to use any other library functions.

Returns

Initialised context structure or 0 if an error occurred

4.2.1.12 void xi_delete_context (xi_context_t * context)

The purpose of this function is to free all allocated resources when the application is intending to terminate or stop using the library.

4.2.1.13 `const xi_response_t* xi_datastream_delete (xi_context_t * xi, xi_feed_id_t feed_id, const char * datastream_id)`

Warning

This function destroys the data in Xively and there is no way to restore it!

4.2.1.14 `const xi_response_t* xi_datapoint_delete (const xi_context_t * xi, xi_feed_id_t feed_id, const char * datastream_id, const xi_datapoint_t * dp)`

Warning

This function destroys the data in Xively and there is no way to restore it!

Note

You need to provide exact timestamp value to guarantee successful response from the API, i.e. it will respond with error 404 if datapoint didn't exist. If you need to determine the exact timestamp, it may be easier to call [xi_datapoint_delete_range\(\)](#) with short range instead.

4.2.1.15 `const xi_response_t* xi_datapoint_delete_range (const xi_context_t * xi, xi_feed_id_t feed_id, const char * datastream_id, const xi_timestamp_t * start, const xi_timestamp_t * end)`

Warning

This function destroys the data in Xively and there is no way to restore it!

4.2.2 Enumeration Type Documentation

4.2.2.1 enum xi_protocol_t

Note

See source code for details of what's implemented.

Enumerator

XI_HTTP `http://api.xively.com`
XI_HTTPS `https://api.xively.com`
XI_TCP `telnet api.xively.com 8081`
XI_TCPS `openssl s_client -host api.xively.com -port 8091 -tls1`
XI_WS `ws://api.xively.com:8080`
XI_WSS `wss://api.xively.com:8090`

4.2.2.2 enum http_header_type_t

Enumerator

XI_HTTP_HEADER_DATE `Date`
XI_HTTP_HEADER_CONTENT_TYPE `Content-Type`
XI_HTTP_HEADER_CONTENT_LENGTH `Content-Length`
XI_HTTP_HEADER_CONNECTION `Connection`
XI_HTTP_HEADER_X_REQUEST_ID `X-Request-Id`
XI_HTTP_HEADER_CACHE_CONTROL `Cache-Control`

XI_HTTP_HEADER_VARY Vary
XI_HTTP_HEADER_COUNT Count
XI_HTTP_HEADER_AGE Age

4.2.2.3 enum xi_value_type_t

Datapoint value types

Enumerator

XI_VALUE_TYPE_I32 32-bit signed integer
XI_VALUE_TYPE_F32 32-bit floating point number
XI_VALUE_TYPE_STR any string-encoded data

Index

external/libxively/src/libxively/xi_err.h, 9
external/libxively/src/libxively/xively.h, 9

feed_id
 xi_context_t, 5

http_header_type_t
 xively.h, 12

input
 xi_context_t, 5

layer_chain
 xi_context_t, 5

protocol
 xi_context_t, 5

XI_HTTP
 xively.h, 12

XI_HTTP_HEADER_AGE
 xively.h, 13

XI_HTTP_HEADER_CACHE_CONTROL
 xively.h, 12

XI_HTTP_HEADER_CONNECTION
 xively.h, 12

XI_HTTP_HEADER_CONTENT_LENGTH
 xively.h, 12

XI_HTTP_HEADER_CONTENT_TYPE
 xively.h, 12

XI_HTTP_HEADER_COUNT
 xively.h, 13

XI_HTTP_HEADER_DATE
 xively.h, 12

XI_HTTP_HEADER_VARY
 xively.h, 12

XI_HTTP_HEADER_X_REQUEST_ID
 xively.h, 12

XI_HTTPS
 xively.h, 12

XI_TCP
 xively.h, 12

XI_TCPS
 xively.h, 12

XI_VALUE_TYPE_F32
 xively.h, 13

XI_VALUE_TYPE_I32
 xively.h, 13

XI_VALUE_TYPE_STR
 xively.h, 13

XI_WS

 xively.h, 12

XI_WSS

 xively.h, 12

xi_context_t, 5

 feed_id, 5

 input, 5

 layer_chain, 5

 protocol, 5

xi_create_context
 xively.h, 11

xi_datapoint_delete
 xively.h, 12

xi_datapoint_delete_range
 xively.h, 12

xi_datapoint_t, 6

xi_datapoint_value_t, 6

xi_datastream_delete
 xively.h, 11

xi_delete_context
 xively.h, 11

xi_err.h
 xi_get_last_error, 9
 xi_set_err, 9

xi_feed_t, 6

xi_get_last_error
 xi_err.h, 9

xi_get_value_f32
 xively.h, 10

xi_get_value_i32
 xively.h, 10

xi_get_value_type
 xively.h, 10

xi_protocol_t
 xively.h, 12

xi_response_t, 7

xi_set_err
 xi_err.h, 9

xi_set_network_timeout
 xively.h, 11

xi_set_value_f32
 xively.h, 10

xi_set_value_i32
 xively.h, 10

xi_set_value_str
 xively.h, 11

xi_timestamp_t, 7

xi_value_pointer_f32
 xively.h, 10

xi_value_pointer_i32

xively.h, 10

xi_value_pointer_str
xively.h, 11

xi_value_type_t
xively.h, 13

xively.h

- XI_HTTP, 12
- XI_HTTP_HEADER_AGE, 13
- XI_HTTP_HEADER_CACHE_CONTROL, 12
- XI_HTTP_HEADER_CONNECTION, 12
- XI_HTTP_HEADER_CONTENT_LENGTH, 12
- XI_HTTP_HEADER_CONTENT_TYPE, 12
- XI_HTTP_HEADER_COUNT, 13
- XI_HTTP_HEADER_DATE, 12
- XI_HTTP_HEADER_VARY, 12
- XI_HTTP_HEADER_X_REQUEST_ID, 12
- XI_HTTPS, 12
- XI_TCP, 12
- XI_TCPS, 12
- XI_VALUE_TYPE_F32, 13
- XI_VALUE_TYPE_I32, 13
- XI_VALUE_TYPE_STR, 13
- XI_WS, 12
- XI_WSS, 12

xively.h

- http_header_type_t, 12
- xi_create_context, 11
- xi_datapoint_delete, 12
- xi_datapoint_delete_range, 12
- xi_datastream_delete, 11
- xi_delete_context, 11
- xi_get_value_f32, 10
- xi_get_value_i32, 10
- xi_get_value_type, 10
- xi_protocol_t, 12
- xi_set_network_timeout, 11
- xi_set_value_f32, 10
- xi_set_value_i32, 10
- xi_set_value_str, 11
- xi_value_pointer_f32, 10
- xi_value_pointer_i32, 10
- xi_value_pointer_str, 11
- xi_value_type_t, 13