

Developer manual

(ONVIF Client Library)

Happytimesoft Technology Co., LTD

Declaration

All rights reserved. No part of this publication may be excerpted, reproduced, translated, annotated or edited, in any form or by any means, without the prior written permission of the copyright owner.

Since the product version upgrade or other reasons, this manual will subsequently be updated. Unless otherwise agreed, this manual only as a guide, this manual all statements, information, recommendations do not constitute any express or implied warranties.

www.happytimesoft.com

Table of Contents

Chapter 1 Build	4
1.1 Dependent libraries.....	4
1.2 Windows Platform.....	4
1.3 Linux Platform.....	4
1.4 Android Platform.....	4
1.5 MAC Platform.....	4
1.6 IOS Platform.....	4
1.7 Embedded Platform.....	5
1.8 Compilation parameter.....	5
Chapter 2 Data Structure	6
2.1 ONVIF_DEVICE.....	6
2.2 Other Data structure.....	7
2.3 Request Data structure.....	7
2.4 Response Data structure.....	7
Chapter 3 ONVIF Version	8
Chapter 4 API Interface	9
4.1 Device discovery API.....	9
4.2 ONVIF Event API.....	10
4.3 System API.....	12
4.4 ONVIF Client API.....	13
Chapter 5 Example	14

Chapter 1 Build

1.1 Dependent libraries

If you enable HTTPS function, please first compile and install OPENSSSL library.

The download link:

<https://www.openssl.org/source/openssl-1.1.1g.tar.gz>

1.2 Windows Platform

Use VS2019 open OnvifClientLibrary.sln to build.

1.3 Linux Platform

```
cd OnvifClientLibrary
```

```
make clean
```

```
make (Compile the dynamic library)
```

```
make -f static.mk clean
```

```
make -f static.mk (Compile the static library)
```

1.4 Android Platform

```
cd OnvifClientLibrary
```

```
make -f android.mk clean
```

```
make -f android.mk
```

Note : Need to install android build environment

use QtCreator open android.pro to build

Note : QT 5.8.0 or later for android needs to be installed

1.5 MAC Platform

```
cd OnvifClientLibrary
```

```
make -f mac.mk clean
```

```
make -f mac.mk
```

1.6 IOS Platform

```
cd OnvifClientLibrary
```

```
make -f ios.mk clean
```

```
make -f ios.mk
```

Or use QtCreator open ios.pro to build

Note : QT 5.8.0 or later for IOS needs to be installed

1.7 Embedded Platform

Modify Makefile to specify cross-compiler, then make.

1.8 Compilation parameter

PROFILE_C_SUPPORT : Implementing ONVIF profile C related interfaces

Include onvif ACCESSCONTROL, DOORCONTROL service

PROFILE_G_SUPPORT : Implementing ONVIF profile G related interfaces

include onvif RECORDING, SEARCH, REPLAY services

THERMAL_SUPPORT : Implementing ONVIF thermal service interfaces

CREDENTIAL_SUPPORT : Implementing ONVIF credential service interfaces

ACCESS_RULES : Implementing ONVIF accessrules service interfaces

SCHEDULE_SUPPORT : Implementing ONVIF schedule service interfaces

RECEIVER_SUPPORT : Implementing ONVIF receiver service interfaces

IPFILTER_SUPPORT : Implementing IP filter interfaces

DEVICEIO_SUPPORT : Implementing ONVIF deviceio service interfaces

PROVISIONING_SUPPORT : Implementing ONVIF provisioning service interfaces

Chapter 2 Data Structure

2.1 ONVIF_DEVICE

The ONVIF_DEVICE structure used to store the device configuration parameters.

```
typedef struct
{
    unsigned int    local_ip;           // local ip address to connect to server, network byte
order

    DEVICE_BINFO    binfo;             // device basic information

    // request
    char            username[32]; // login user name, set by user
    char            password[32]; // login password, set by user

    BOOL            authFailed;        // when login auth failed, set by onvif stack

    ONVIF_Profile * curProfile;         // current profile pointer, the default pointer the first
profile, user can set it

    /*****/

    ONVIF_VideoSource * video_src;      // the list of video source
    ONVIF_AudioSource * audio_src;      // the list of audio source
    ONVIF_Profile * profiles;           // the list of profile
    ONVIF_VideoSourceConfiguration * video_src_cfg; // the list of video source
configuration
    ONVIF_AudioSourceConfiguration * audio_src_cfg; // the list of audio source
configuration
    ONVIF_VideoEncoderConfiguration * video_enc;  // the list of video encoder
configuration
    ONVIF_AudioEncoderConfiguration * audio_enc;  // the list of audio encoder
configuration

    ONVIF_PTZNode * ptznodes;           // the list of ptz node
    ONVIF_PTZConfiguration * ptz_cfg;    // the list of ptz configuration

    /*****/
}
```

```

    ONVIF_EVENT    events;                // event information
    onvif_DeviceInformation    DeviceInformation;    // device information
    onvif_Capabilities    Capabilities;            // device capabilities
} ONVIF_DEVICE;

```

Field	Description
local_ip	local ip address to connect to server, network byte order
binfo	device basic information
username	login user name, set by user
password	login password, set by user
authFailed	when login auth failed, set by onvif stack
curProfile	current profile pointer, the default pointer the first profile, user can set it
video_src	Video source list
audio_src	Audio source list
profiles	Media profiles list
video_src_cfg	Video source configurations list
audio_src_cfg	Audio source configurations list
video_enc	Video encoder list
audio_enc	Audio encoder list
ptznodes	PTZ nodes list
ptz_cfg	PTZ configurations list
events	Onvif event message
DeviceInformation	Device information
Capabilities	Device capabilities

2.2 Other Data structure

Other ONVIF standard data structure, please refer onvif_cm.h file

2.3 Request Data structure

ONVIF request data structure, please refer onvif_req.h file

2.4 Response Data structure

ONVIF response data structure, please refer onvif_res.h file

Chapter 3 ONVIF Version

The onvif client library implements the following ONVIF service:

ONVIF Service	Prefix	Url	version
device	tds	http://www.onvif.org/ver10/device/wsdl	23.06
event	tev	http://www.onvif.org/ver10/events/wsdl	22.06
media	trt	http://www.onvif.org/ver10/media/wsdl	21.12
media 2	tr2	http://www.onvif.org/ver20/media/wsdl	23.06
ptz	tptz	http://www.onvif.org/ver20/ptz/wsdl	22.12
image	timg	http://www.onvif.org/ver20/imaging/wsdl	22.06
analytics	tan	http://www.onvif.org/ver20/analytics/wsdl	22.06
recording control	trc	http://www.onvif.org/ver10/recording/wsdl	23.06
search	tse	http://www.onvif.org/ver10/search/wsdl	22.06
replay	trp	http://www.onvif.org/ver10/replay/wsdl	21.12
access control	tac	http://www.onvif.org/ver10/accesscontrol/wsdl	21.06
door control	tdc	http://www.onvif.org/ver10/doorcontrol/wsdl	21.06
device IO	tmd	http://www.onvif.org/ver10/deviceIO/wsdl	22.06
thermal	tth	http://www.onvif.org/ver10/thermal/wsdl	22.06
credential	tcr	http://www.onvif.org/ver10/credential/wsdl	21.06
access rules	tar	http://www.onvif.org/ver10/accessrules/wsdl	19.06
schedule	tsc	http://www.onvif.org/ver10/schedule/wsdl	18.12
receiver	trv	http://www.onvif.org/ver10/receiver/wsdl	21.12
provisioning	tpv	http://www.onvif.org/ver10/provisioning/wsdl	18.12

Chapter 4 API Interface

4.1 Device discovery API

The device discovery API in `onvif_probe.h` file.

1. `int start_probe(const char * ip, int interval)`

desc: start device discovery thread

params:

ip: start device discovery thread on the specify ip address, it can be NULL, if ip is NULL, it will use the default ip.

interval: probe interval, unit is second, default is 30s

return: 0-success, -1-failed

2. `void stop_probe()`

desc: stop device discovery thread

3. `void set_probe_interval(int interval)`

desc: set probe interval

params:

interval: probe interval, unit is second, default is 30s, if the interval less than 10, will be set to 30.

4. `void send_probe_req()`

desc: send probe request. the device discovery thread will automatically send a PROBE request at the specified interval. Calling this interface will immediately send a PROBE request.

5. `void set_probe_cb(onvif_probe_cb cb, void * pdata)`

desc: set device probed callback function

params:

cb: the callback function point

pdata: user data, when call backback pass back to user.

The callback function prototype is as follows:

```
void (* onvif_probe_cb) (DEVICE_BINFO * p_res, int msgtype, void * pdata)
```

p_res : the DEVICE_BINFO struct pointer

msgtype :

PROBE_MSGTYPE_MATCH - probe match

PROBE_MSGTYPE_HELLO - hello

PROBE_MSGTYPE_BYE - bye

note : if msgtype = PROBE_MSGTYPE_BYE, only p_res->EndpointReference field is valid, the other fields is invalid

pdata : the user data

```
6. void set_monitor_cb(char * reference, int msgtype, onvif_probe_cb cb, void * pdata)
```

desc: Set the broadcast message (ProbeMatch, Hello, Bye) callback function for a specific device.

params:

reference : endpoint reference.

msgtype : message type, PROBE_MSGTYPE_MATCH, PROBE_MSGTYPE_HELLO or PROBE_MSGTYPE_BYE.

cb: the callback function point.

pdata: user data, when call backback pass back to user.

The callback function prototype please refer to set_probe_cb.

4.2 ONVIF Event API

The ONVIF Event API in onvif_event.h file.

```
1. BOOL onvif_event_init(BOOL http_enable, const char * http_srv_addr, uint16 http_srv_port, BOOL https_enable, const char * https_srv_addr, uint16 https_srv_port, const char * cert_file, const char * key_file, int max_clients)
```

desc: init event handler, init http server and https server for receiving event notify messages.

params:

http_enable: http enable flag

http_srv_ip : http server listen address, NULL means listening on all interfaces.

http_srv_port : http server listen port.
https_enable : https enable flag.
https_srv_addr : https server listen address, NULL means listening on all interfaces.

https_srv_port : https server listen port.
cert_file : cert file for https.
key_file : key file for https.
max_clients : max support client numbers

2. void **onvif_event_deinit()**

desc: uninit event handler

3. void **onvif_set_event_notify_cb**(onvif_event_notify_cb cb, void * pdata)

desc: set event notify callback

params:

cb: the callback function pointer, set cb to NULL, disable callback.

pdata: user data, when call backback pass back to user.

The callback function prototype is as follows:

```
void (* onvif_event_notify_cb)(Notify_REQ * p_req, void * pdata)
```

p_req: the notify message

pdata: the user data

4. void **onvif_set_subscribe_disconnect_cb**(onvif_subscribe_disconnect_cb cb, void* pdata)

desc: set event subscribe disconnect callback

params:

cb: the callback function pointer, set cb to NULL, disable callback.

pdata: user data, when call backback pass back to user.

The callback function prototype is as follows:

```
void (* onvif_subscribe_disconnect_cb)(ONVIF_DEVICE * p_dev, void * pdata)
```

p_dev: the onvif device pointer

pdata: the user data

4.3 System API

1. int **log_init**(const char * log_fname)

desc: open log file

params:

log_fname: log file name.

return: 0-success, -1-failed

2. int **log_time_init**(const char * fname_prev)

desc: open log file with timestamp

params:

fname_prev: log file name prefix. Such as :

log_time_init("log"); // The format of the generated log file is

log-xxxxxxx-xxxxxx.txt

return: 0-success, -1-failed

3. void **log_set_level**(int level)

desc: set log level

params:

level: log level, as the following:

HT_LOG_TRC

HT_LOG_DBG

HT_LOG_INFO

HT_LOG_WARN

HT_LOG_ERR

HT_LOG_FATAL

4. void **log_close**()

desc: close log file

5. BOOL **sys_buf_init**(int nums)

desc: allocate systems buffer

params:

nums: the number of system buffer allocated, each system buffer size is 2048 bytes

NOTE: The allocated system buffer size is generally the maximum number of supported devices multiplied by 10.

The ONVIF CLIENT library allocates a piece of memory to receive and parse network data, so there is no need to dynamically allocate memory.

return: TRUE-success, FALSE-failed

6. void **sys_buf_deinit()**

desc: free systems buffer

7. BOOL **http_msg_buf_init**(int num)

desc: allocate http message parser buffer

params:

nums: the number of system buffer allocated, each http parser buffer size is about 300 bytes

8. void **http_msg_buf_deinit()**

desc: free http message parser buffer

4.4 ONVIF Client API

The ONVIF client API in onvif_cln.h file.

All ONVIF CLIENT APIs comply with ONVIF standard documentations, for details on the interfaces, please refer to the ONVIF standard documents:

<https://www.onvif.org/profiles/specifications/>

The API in onvif_api.h file is a simplified version of some commonly used API interfaces.

ONVIF Application Programmers Guide :

https://www.onvif.org/wp-content/uploads/2016/12/ONVIF_WG-APG-Application_Programmers_Guide-1.pdf

Chapter 5 Example

Onvif client library usage examples please refer OnvifTest.cpp, OnvifTest2.cpp and OnvifTest3.cpp files.

Onviftest.cpp : Automatically discover onvif devices and perform test steps.

OnvifTest2.cpp : Manually add devices, get device information and subscribe events.

OnvifTest3.cpp : Thread version of OnvifTest2.cpp.