WeHome SDK User Manual

Contents

About this document......1
Introduction......1
Overview......1
Architecture......2
API Reference......3
EC_Initialize......3
EC_DeInitialize......3
EC_StartSearchDev......4
EC_StopDevSearch......4
EC_GetDevList......5
EC_Login......5
EC_Logout......6
EC_SendCommand......7
EC_SendTalkData......7
EC_Subscribe......8
EC_UnSubscribe......9
EC_ResetBadge......10
EC_QueryOnlineStatus......11
startSmartConfig......11
stopSmartConfig......12
Data Structures......13
CONNECT_TYPE......13
LOGIN_RESULT......13
CMD_EXEC_RESULT......14
lpLoginResult......14
lpCmdResult......15
lpAudio_RecvData......15
lpVideo_RecvData......16
lpPBAudio_RecvData......16
lpPBVideo_RecvData......17
lpPBEnd......18
EC_INIT_INFO......18
DeviceInfo......20
OnlineQueryResultCallback......20
How to use......21
Login & Logout......21
Wi-Fi configuration......22
Device search......24
Live Streaming......25
Send command......26
Remote playback......27

Talk with remote device 28
Notifications 29
Query device online status 30
Other topics 31
Two-way audio communication 31
Notifications 31

History

Ver	Auth	Edit date	Changes
0.8.0	Dave	2018.7.23	New version

About this document

This document provides reference information for the programmers that integrate WeHome SDK to their APP to communicate with Linkwil's battery doorbells, cameras and other products. The information includes WeHome SDK's architecture, programming interface, error codes, and sequence diagram for common use case.

Note that this document is suitable for the following products:

* Doorbell M8

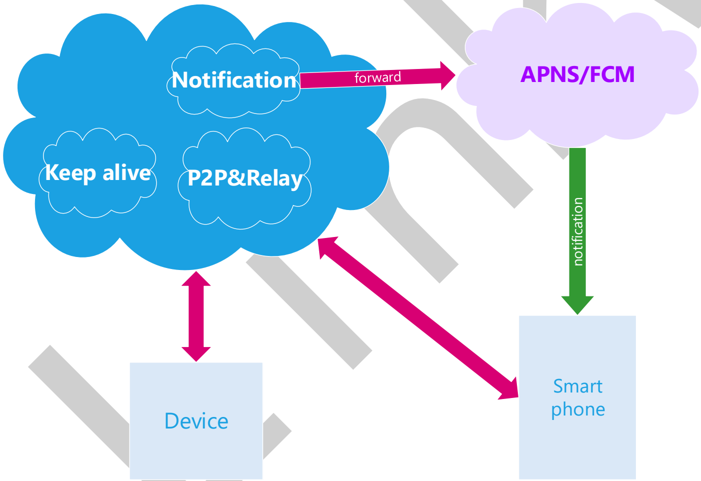
* Doorbell M6

* IP Camera A6

Introduction

Overview

Linkwil's Low Power Consumption solutions architecture:



The backend server consists of 3 parts:

- P2P & Relay server is used for data transform
- Notification server is used for notification management, such as subscribe, unsubscribe and build message for APNS & FCM.
- Keep alive server is used to wake up remote device when in deep sleep mode.

The device is low power consumption and supports fast startup even when in deep sleep mode.

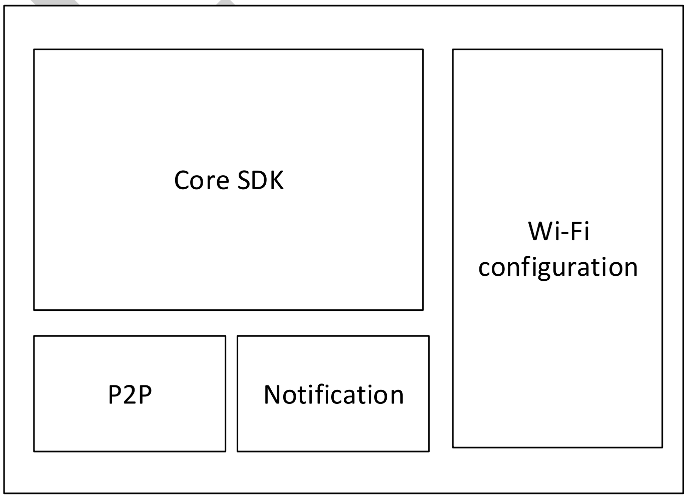
WeHome SDK provides the ability for developers to integrate Linkwil's products to their existing APP or develop a new APP rapidly. WeHome SDK is C based, and open sourced, so it is easy to port to any platform, but we still recommend you only use it for android & iOS, because only these two platforms are fully tested. WeHome SDK provides APIs to integrate all functions of Linkwil's products to your own APP, the functions include:

- SDK initialize & de-Initialize
- Wi-Fi configuration (Only available for Android & iOS)
- Device search in LAN
- Login & Logout to remote device
- Live streaming video
- Send command to remote device
- Talk with remote device
- Remote playback
- Notification subscribe & un-subscribe
- Get device online status

Architecture

As show in figure1.1, WeHome SDK consists of the following layers:

- Low level P2P communication layer
  Third-party P2P communication library, transform data between clients and remote devices reliably.
- Notification manager layer:
  Subscribe notification and un-subscribe notification on Linkwil's notification server. Note that receive notifications from Google's FCM server or Apple's APNS server is not included in WeHome SDK, it is the business of application layer.
- Wi-Fi configuration component
  This component provides the ability to setup the Wi-Fi information to a device in Wi-Fi configuration state. WeHome SDK use sound wave to configure Wi-Fi.
- Core SDK layer
  Core SDK layer is the logic layer of WeHome SDK, it is fully open-sourced under LGPL protocol.



API Reference

EC_Initialize

[Description]
Initialize SDK runtime environment and register callbacks.

[Syntax]
EASYCAM_API int EC_Initialize(EC_INIT_INFO* init);

[Parameter]

Parameter	Description	Input / Output
init	SDK init info consists of all callback functions	Input

[Return value]

Return value	Description
0	Success
-1	Already initialized

[Note]
This API should be called before anything.

EC_DeInitialize

[Description]
De-Initialize SDK and release resource.

[Syntax]
EASYCAM_API void EC_DeInitialize(void);

[Parameter]
None

[Return value]
None

[Note]

None

EC_StartSearchDev

[Description]

Start search all the device in LAN.

[Syntax]

EASYCAM_API int EC_StartSearchDev(int timeoutMS, char* bCastAddr);

[Parameter]

Parameter	Description	Input / Output
timeoutMS	Timeout value in micro-second	Input
bCastAddr	Broadcast address for LAN, such as 192.168.1.255	Input

[Return value]

Return value	Description
0	Success
-1	Already initialized

[Note]

- This API will block for some time, should not be called in UI thread
- The device can't be searched when in sleep mode

EC_StopDevSearch

[Description]

Interrupt device search.

[Syntax]

EASYCAM_API int EC_StopDevSearch(void);

[Parameter]

None

[Return value]

Return value	Description
0	Success
-1	Failed

[Note]

None

EC_GetDevList

[Description]

Get device list in LAN after device searched.

[Syntax]

EASYCAM_API int EC_GetDevList(DeviceInfo* pDevInfo, int devInfoSize);

[Parameter]

Parameter	Description	Input / Output
pDevInfo	Device information buffer	Output
devInfoSize	Device information buffer size	Input

[Return value]

Return value	Description
>=0	Device count
-1	Sdk hasn’t been inited.

[Note]

This API should be called after EC_StartSearchDev to get the device list

EC_Login

[Description]

Connect and login to a remote device.

[Syntax]

EASYCAM_API int EC_Login(const char* uid, const char* usrName,
    const char* password, const char* broadcastAddr,
    int seq, int needVideo, int needAudio, int connectType, int timeout);

[Parameter]

Parameter	Description	Input / Output
uid	UID of remote device	Input
usrName	Unused, should always be "admin"	Input
password	Access password of the remote device	Input
broadcastAddr	LAN broadcast address, eg. 192.168.1.255	Input
seq	Command sequence, should be unique during a session	Input
needVideo	If need video data after login	Input
needAudio	If need audio data after login	Input
connectType	Refer CONNECT_TYPE	Input
timeout	Timeout value unit in micro seconds	Input

[Return value]

Return value	Description
>=0	Session handle
-1	Sdk hasn't been inited.

[Note]

Session login result will be returned at callback of logIn, refer to: EC_INIT_INFO

EC_Logout

[Description]

Logout and disconnect from a remote device.

[Syntax]

EASYCAM_API int EC_Logout(int handle);

[Parameter]

Parameter	Description	Input / Output
handle	Session handle returned by logIn	Input

[Return value]

Return value	Description
>=0	Session handle
-1	Sdk hasn't been inited.

[Note]

The remote device will enter sleep mode after the session logout.

EC_SendCommand

[Description]

Send command to remote device. The command execution result will be return in command callback, refer to EC_INIT_INFO.

[Syntax]

EASYCAM_API int EC_SendCommand(int handle, char* command, int seq);

[Parameter]

Parameter	Description	Input / Output
handle	Session handle returned by logIn	Input
command	Command string in json format	Input
seq	Sequence number of this command	Input

[Return value]

Return value	Description
0	Success
-1	Sdk hasn’t been inited.

[Note]

The command will be queued in SDK, and send to remote device one by one.

EC_SendTalkData

[Description]

Send PCM audio data to remote device.

[Syntax]

EASYCAM_API int EC_SendTalkData(int handle, char* data, int dataLen, int payloadType, int seq);

[Parameter]

Parameter	Description	Input / Output
handle	Session handle returned by logIn	Input
data	PCM audio data	Input

	Data format: Sample rate: 16000 Sample size: 16bit	
dataLen	Audio data size in bytes	Input
payloadType	Unused	Input
seq	Sequence number of this command	Input

[Return value]

Return value	Description
0	Success
-1	Sdk hasn’t been inited.

[Note]

None

EC_Subscribe

[Description]

Subscribe notification from message server. Only support FCM for android and APNS for iOS. Refer to notification

[Syntax]

EASYCAM_API int EC_Subscribe(const char* uid, const char* appName, const char* agName, const char* phoneToken, unsigned int eventCh);

[Parameter]

Parameter	Description	Input / Output
uid	UID of the device	Input
appName	APP name of which want to subscribe the notification	Input
agName	For android need to be “FCM” For iOS need to be “APNS”	Input
phoneToken	FCM token string or APNS token string	Input
eventCh	Event channel returned in callback of logIn Refer to EC_INIT_INFO.	Input

[Return value]

Return value	Description
0	Success

-1	Failed

[Note]

- If you want use notification function, please contact Linkwil to register your APP's message notification information on server firstly, otherwise, this function will always fail.
- This function will be blocked when subscribe message from server, please DON'T call at UI thread.

EC_UnSubscribe

[Description]

Unsubscribe message from server. You can not receive any notifications from this device after un-subscribed.

[Syntax]

EASYCAM_API int EC_Subscribe(const char* uid, const char* appName, const char* agName, const char* phoneToken, unsigned int eventCh);

[Parameter]

Parameter	Description	Input / Output
uid	UID of the device	Input
appName	APP name of which want to subscribe the notification	Input
agName	For android need to be "FCM" For iOS need to be "APNS"	Input
phoneToken	FCM token string or APNS token string. It should be the same token when subscribe, otherwise, this API may return an error.	Input
eventCh	Event channel, it should be the same as the event channel when subscribe, otherwise, this API may return an error	Input

[Return value]

Return value	Description
0	Success
-1	Failed

[Note]

- If you want use notification function, please contact Linkwil to register your APP's message notification information on server firstly, otherwise, this function will always fail.

- This function will be blocked when subscribe message from server, please DON'T call at UI thread.

EC_ResetBadge

[Description]

Reset the badge number.

For iOS phone, you need to call this API to reset the badge number on message push server, otherwise, the badge number will always increase until max.

[Syntax]

EASYCAM_API int EC_ResetBadge(const char* uid, const char* appName, const char* agName, const char* phoneToken, unsigned int eventCh);

[Parameter]

Parameter	Description	Input / Output
uid	UID of the device	Input
appName	APP name of which want to subscribe the notification	Input
agName	For android need to be "FCM" For iOS need to be "APNS"	Input
phoneToken	FCM token string or APNS token string. It should be the same token when subscribe, otherwise, this API may return an error.	Input
eventCh	Event channel, it should be the same as the event channel when subscribe, otherwise, this API may return an error	Input

[Return value]

Return value	Description
0	Success
-1	Failed

[Note]

- If you want use notification function, please contact Linkwil to register your APP's message notification information on server firstly, otherwise, this function will always fail.
- This function will be blocked when subscribe message from server, please DON'T

call at UI thread.

EC_QueryOnlineStatus

[Description]

Query the online status of a device. The remote device need not be waked up when call this API.

[Syntax]

EASYCAM_API int EC_QueryOnlineStatus(const char* uid,
OnlineQueryResultCallback callback);

[Parameter]

Parameter	Description	Input / Output
uid	UID of the device	Input
callback	Query result callback function	Input

[Return value]

Return value	Description
0	Success
-1	Failed

[Note]

None

startSmartConfig

[Description]

Start Wi-Fi configuration.

[Syntax]

int startSmartConfig(Context context, final String wifiSsid, final String wifiPassword, final String devPassword)

[Parameter]

Parameter	Description	Input / Output
context	Context, only for android	Input
wifiSsid	Wi-Fi ssid	Input
wifiPassword	Wi-Fi ssid	Input

devPassword	Device access password	Input

[Return value]

Return value	Description
0	Success
-1	Failed

[Note]

Refer the demo for detail.

stopSmartConfig

[Description]

Sop Wi-Fi configuration.

[Syntax]

void stopSmartConfig()

[Parameter]

None

[Return value]

None

[Note]

Refer the demo for detail.

Data Structures

CONNECT_TYPE

[Description]

CONNECT_TYPE_LAN mode means it will only connect to remote device if APP and the device locate in the same LAN, CONNECT_TYPE_P2P mode means it only connect to remote by P2P mode, if the NAT stops P2P, then connection will fail. CONNECT_TYPE_RELAY mode means just use relay mode to connection to remote.

[Syntax]

```
#define CONNECT_TYPE_LAN                      (1<<0)
#define CONNECT_TYPE_P2P                      (1<<1)
#define CONNECT_TYPE_RELAY                    (1<<2)
```

[Note]

For normal use, we suggest you support both LAN, P2P and RELAY mode to ensure success ration. It will introduce more delay if you just select RELAY mode.

LOGIN_RESULT

[Description]

Login result error code.

[Syntax]

```
#define LOGIN_RESULT_SUCCESS                     (0)
#define LOGIN_RESULT_CONNECT_FAIL                (-1)
#define LOGIN_RESULT_AUTH_FAIL                   (-2)
#define LOGIN_RESULT_EXCEED_MAX_CONNECTION       (-3)
#define LOGIN_RESULT_RESULT_FORMAT_ERROR         (-4)
#define LOGIN_RESULT_FAIL_UNKOWN                 (-5)
```

[Note]

None

CMD_EXEC_RESULT

[Description]

Command execution result

[Syntax]

```
#define CMD_EXEC_RESULT_SUCCESS          (0)
#define CMD_EXEC_RESULT_FORMAT_ERROR     (-1)
#define CMD_EXEC_RESULT_SEND_FAIL        (-2)
#define CMD_EXEC_RESULT_AUTH_FAIL        (-3)
```

[Note]

None

lpLoginResult

[Description]

Login result callback function

[Syntax]

```
typedef void(*lpLoginResult)( int handle, int errorCode, int seq,
   unsigned int notificationToken, unsigned int isCharging, unsigned int batPercent);
```

[Parameter]

Parameter	Description
handle	Session handle
errorCode	Login result error code, Refer LOGIN_RESULT
seq	Sequence number equals to seq parameter in EC_Login()
notificationToken	The event channel used to subscribe notification. Refer to EC_Subscribe()
isCharging	Device’s battery is in charging or not.
batPercent	Battery remain percent, 0~100

[Note]

None

lpCmdResult

[Description]

Command execution result callback

[Syntax]

void(*lpCmdResult)(int handle, char* data, int errorCode, int seq);

[Parameter]

Parameter	Description
handle	Session handle
data	Command execution result in json format.
errorCode	Command execution result code, Refer CMD_EXEC_RESULT
seq	Sequence number of the command

[Note]

None

lpAudio_RecvData

[Description]

Live stream’s audio data callback function.

[Syntax]

void(*lpAudio_RecvData)(int handle, char *data, int len, int payloadType,
long long timestamp, int seq);

[Parameter]

Parameter	Description
handle	Session handle
data	Audio data
len	Audio data length
payloadType	Unused, always PCM data now
timestamp	Unused
seq	Sequence number

[Note]

None

lpVideo_RecvData

[Description]

Live stream's video data callback function.

[Syntax]

void(*lpVideo_RecvData)(int handle, char *data, int len, int payloadType,
long long timestamp, int seq, int frameType, int videoWidth, int videoHeight,
unsigned int wifiQuality);

[Parameter]

Parameter	Description
handle	Session handle
data	Audio data
len	Audio data length
payloadType	Unused, always H.264 now
timestamp	Unused
seq	Sequence number
FrameType	1=IDR frame 0=P frame
videoWidth	Video width
videoHeight	Video height
wifiQuality	Wi-Fi signal quality (0~100)

[Note]

None

lpPBAudio_RecvData

[Description]

Remote playback stream's audio data callback function.

[Syntax]

void(*lpPBAudio_RecvData)(int handle, char *data, int len,
int payloadType, long long timestamp, int seq, int pbSessionNo);

[Parameter]

Parameter	Description
handle	Session handle

data	Audio data
len	Audio data length
payloadType	Unused, always PCM now
timestamp	Timestamp in us
seq	Sequence number
pbSessionNo	Session number returned in EC_CMD_ID_START_PLAY_RECORD command result, Refer < WeHome SDK Commands Description.pdf> for detail

[Note]

None

lpPBVideo_RecvData

[Description]

Remote playback stream's video data callback function.

[Syntax]

```
void(*lpPBVideo_RecvData)(int handle, char *data, int len,
            int payloadType, long long timestamp, int seq, int frameType,
            int videoWidth, int videoHeight, int pbSessionNo);
```

[Parameter]

Parameter	Description
handle	Session handle
data	Audio data
len	Audio data length
payloadType	Unused, always PCM now
timestamp	Timestamp in us
seq	Sequence number
frameType	1=IDR frame 0=P frame
videoWidth	Video width
videoHeight	Video height
pbSessionNo	Session number returned in EC_CMD_ID_START_PLAY_RECORD command result, Refer <WeHome SDK Commands Description.pdf> for detail

[Note]

None

lpPBEnd

[Description]

Remote playback stream's end frame callback function. It means playing end of a video segment.

[Syntax]

void(*lpPBEnd)(int hanlde, int pbSessionNo);

[Parameter]

Parameter	Description
handle	Session handle
pbSessionNo	Session number returned in EC_CMD_ID_START_PLAY_RECORD command result, Refer <WeHome SDK Commands Description.pdf> for detail

[Note]

None

EC_INIT_INFO

[Description]

SDK initialize info

[Syntax]

```
typedef struct tagEC_INIT_INFO
{
        void(*lpLoginResult)(int handle, int errorCode, int seq,
                  unsigned int notificationToken,
                  unsigned int isCharging,
                  unsigned int batPercent);
        void(*lpCmdResult)(int handle, char* data, int errorCode, int seq);
        void(*lpAudio_RecvData)(int handle, char *data, int len, int payloadType,
                  long long timestamp, int seq);
        void(*lpVideo_RecvData)(int handle, char *data, int len, int payloadType,
                  long long timestamp, int seq, int frameType, int videoWidth, int
                  videoHeight, unsigned int wifiQuality);
        void(*lpPBAudio_RecvData)(int handle, char *data, int len,
                  int payloadType, long long timestamp, int seq, int pbSessionNo);
        void(*lpPBVideo_RecvData)(int handle, char *data, int len,
                  int payloadType, long long timestamp, int seq, int frameType,
                  int videoWidth, int videoHeight, int pbSessionNo);
        void(*lpPBEnd)(int hanlde, int pbSessionNo);
        void(*lpFileDownload_RecvData)(int handle, char* data, int len,
                  int sessionNo);
        void(*lpPIRData_RecvData)(int handle, long long timeMS,
                short adc);
}EC_INIT_INFO;
```

[Member]

Member	Description
lpLoginResult	Callback function of logIn()
lpCmdResult	Callback function of sendCommand()
lpAudio_RecvData	Callback function of audio packets
lpVideo_RecvData	Callback function of video packets
lpPBAudio_RecvData	Callback function of remote playback audio packets
lpPBVideo_RecvData	Callback function of remote playback video packets
lpPBEnd	Callback function of remote playback end frame
lpFileDownload_RecvData	Just for debug, unused
lpPIRData_RecvData	Just for debug, unused

[Note]

None

DeviceInfo

[Description]

Device information

[Syntax]

```
typedef struct tagDeviceInfo
{
    int devType; //0=Camera 1=DoorBell
    char uid[32];
    char devName[64];
    char fwVer[64];
}DeviceInfo;
```

[Member]

Member	Description
devType	Device type, 0 is Camera, 1 is Doorbell
uid	UID of the device
devName	Device name
fwVer	Firmware version name

[Note]

OnlineQueryResultCallback

[Description]

Query online status callback function

[Syntax]

typedef void(*OnlineQueryResultCallback)(int queryResult, const char* uid, int lastLoginSec);

[Parameter]

Parameter	Description
queryResult	Device type, 0 is Camera, 1 is Doorbell
uid	UID of the device
lastLoginSec	Seconds since last heartbeat to backend server

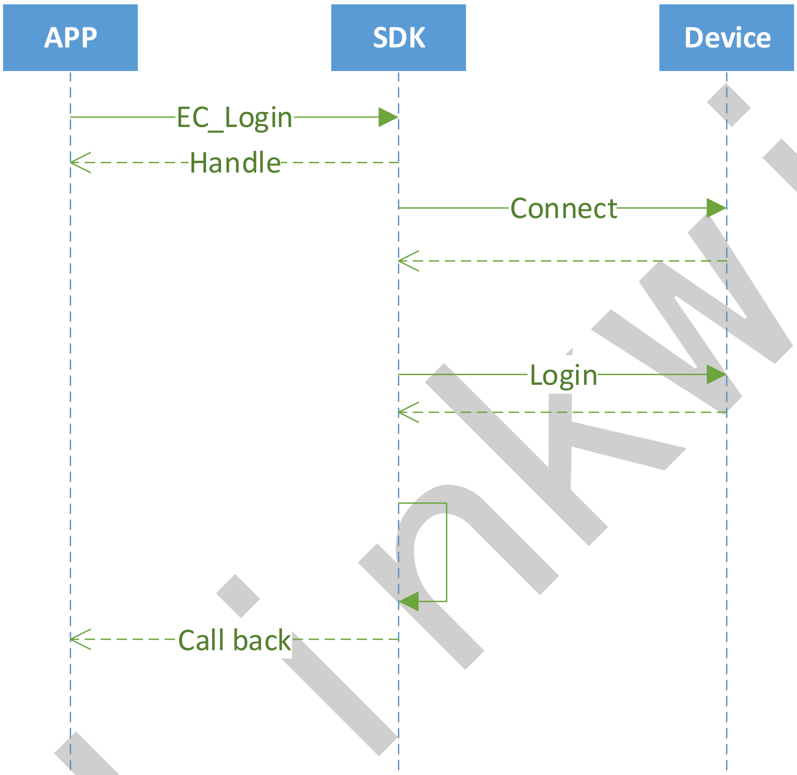
[Note]

We suggest you set the device status to offline if lastLoginSec>90s.

How to use

Login & Logout

EC_Login will create a session handle and return it immediately. Login to remote device is asynchronously processed, and the result will be returned from the login callback function registered in EC_Initialize().



Example code:

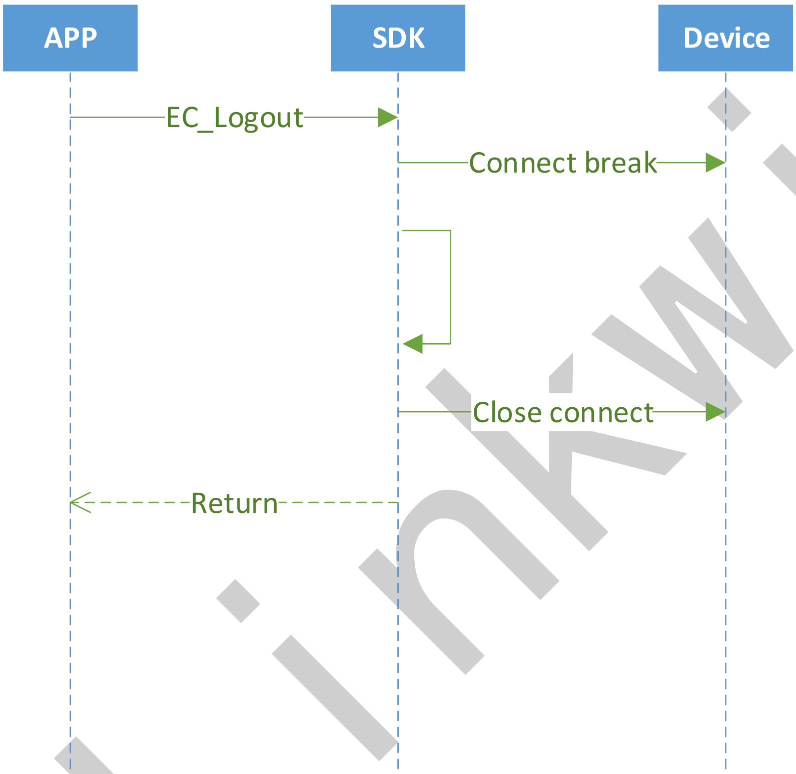
```
static void onLoginResult(int handle, int errorCode, int seq,
            unsigned int notificationToken, unsigned int isCharging,
            unsigned int batPercent)
{
    printf("Login to remote device complete\n");
}

void testLogin()
{
   int cmdSeq = 0;
```

```
    // Initialize the SDK & register callback
    ...

    // Note that login is async API, login result will in callback
    int handle = EC_Login("LBCS-000000-XXXXX", "admin", "admin",
                    "192.168.1.255", cmdSeq++, 1, 1,
                    CONNECT_TYPE_LAN|CONNECT_TYPE_P2P|CONNECT_TYPE_RELAY, 3000);
}
```

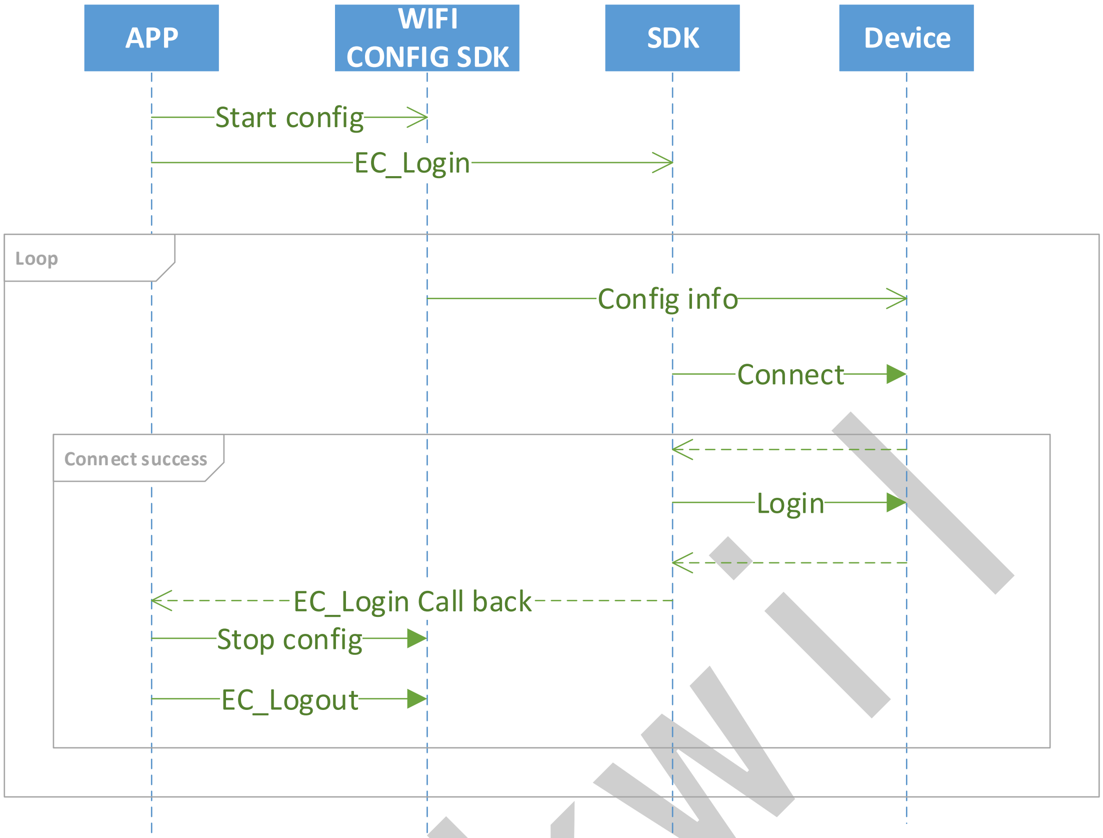
You need to call EC_Logout when you want to terminate a session, EC_Logout is synchronously processed, and will not block UI thread.



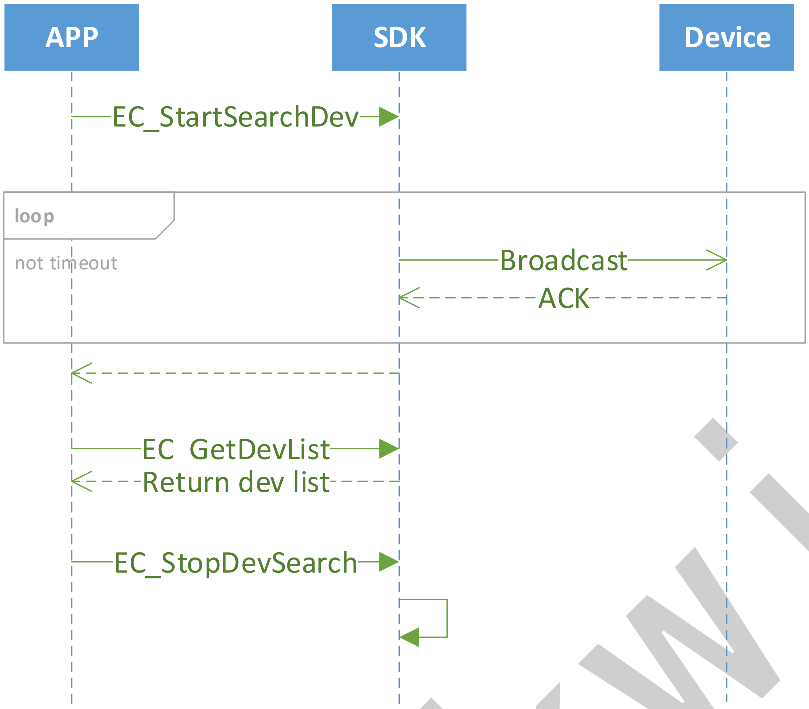
Wi-Fi configuration

WeHome SDK use sound wave to config Wi-Fi information to a device in Wi-Fi configure status, so you need make sure your smartphone can play sound and don’t mute when Wi-Fi configuring, and also need to keep environment quit when configuring.

StartConfig only need be called once, and then call EC_Login() with a reasonable timeout value(eg, 180s) to check if Wi-Fi configuration is success or timeout.

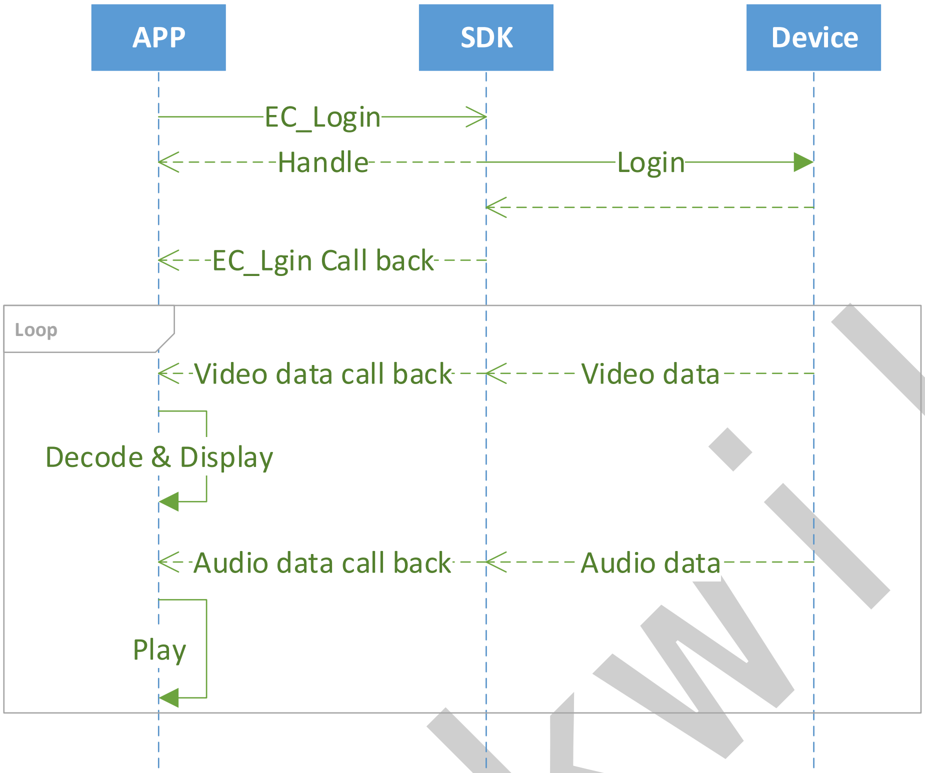

APP
WIFI CONFIG SDK
SDK
Device
Start config
EC_Login
Loop
Config info
Connect
Connect success
Login
EC_Login Call back
Stop config
EC_Logout

Device search



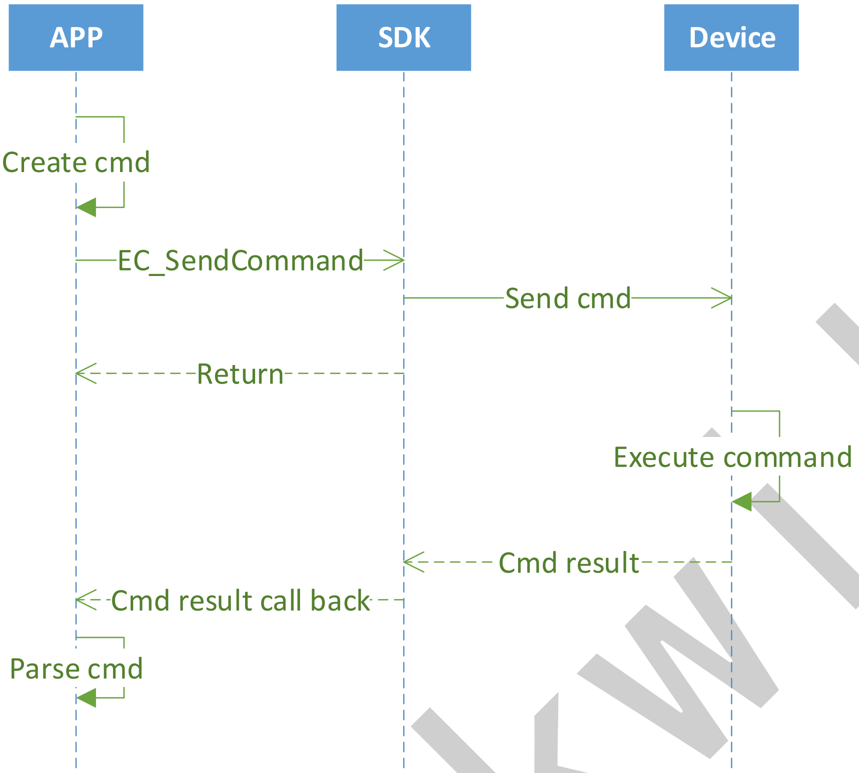
Developers can search the devices in LAN when the device is in wakeup mode.

Live Streaming

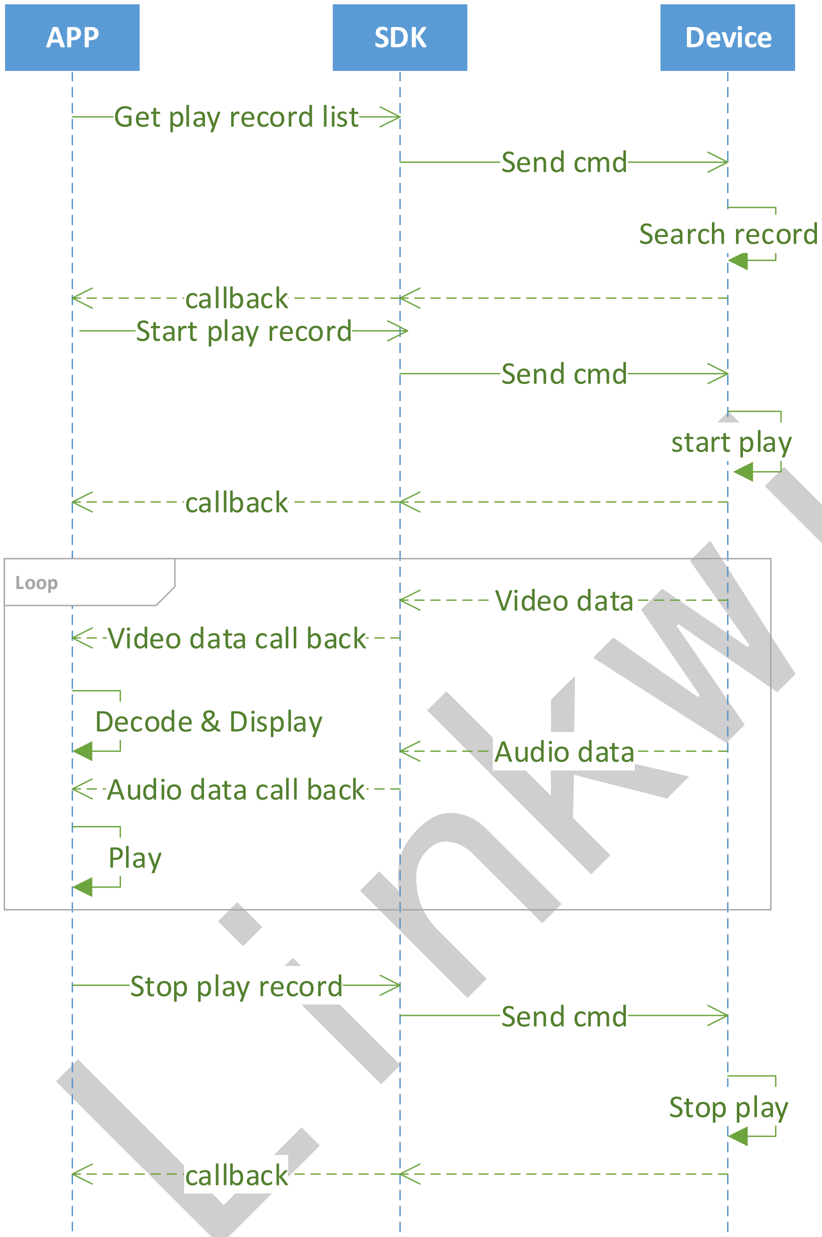


Live stream data (video & audio packets) will come from the callback function registered in EC_Initialize() frame by frame. Video packet is in H.264 format, and audio packet is in PCM format with 16000 sample rate and 16bit sample size.

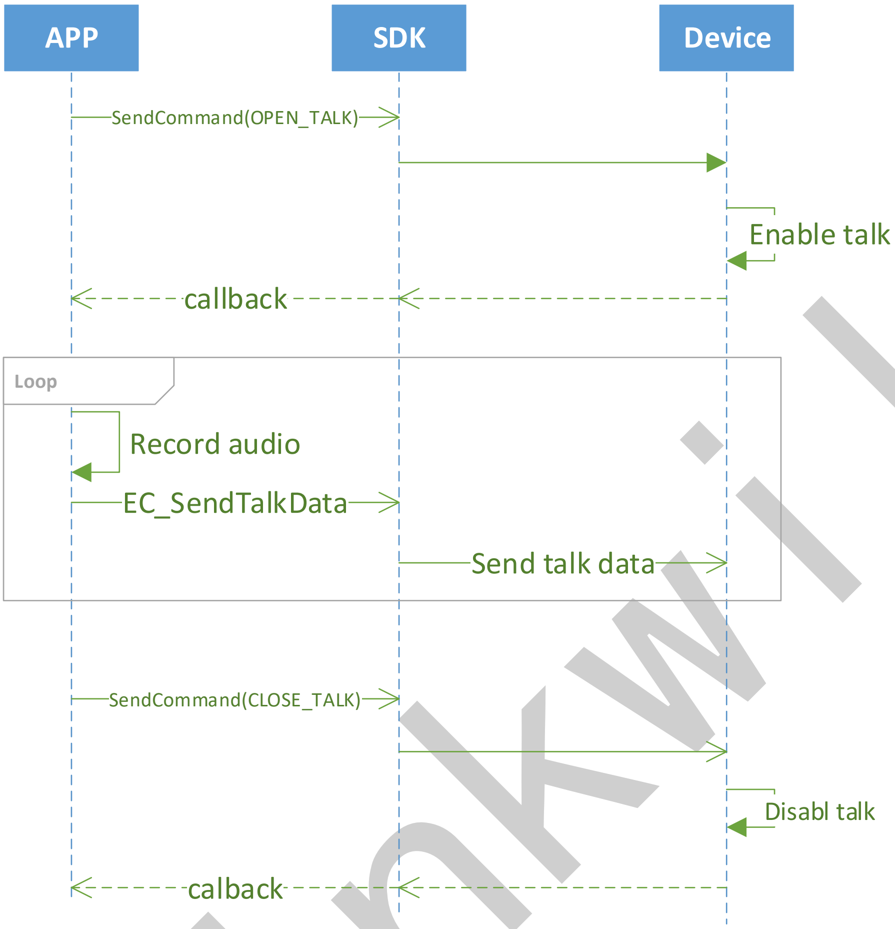
Send command



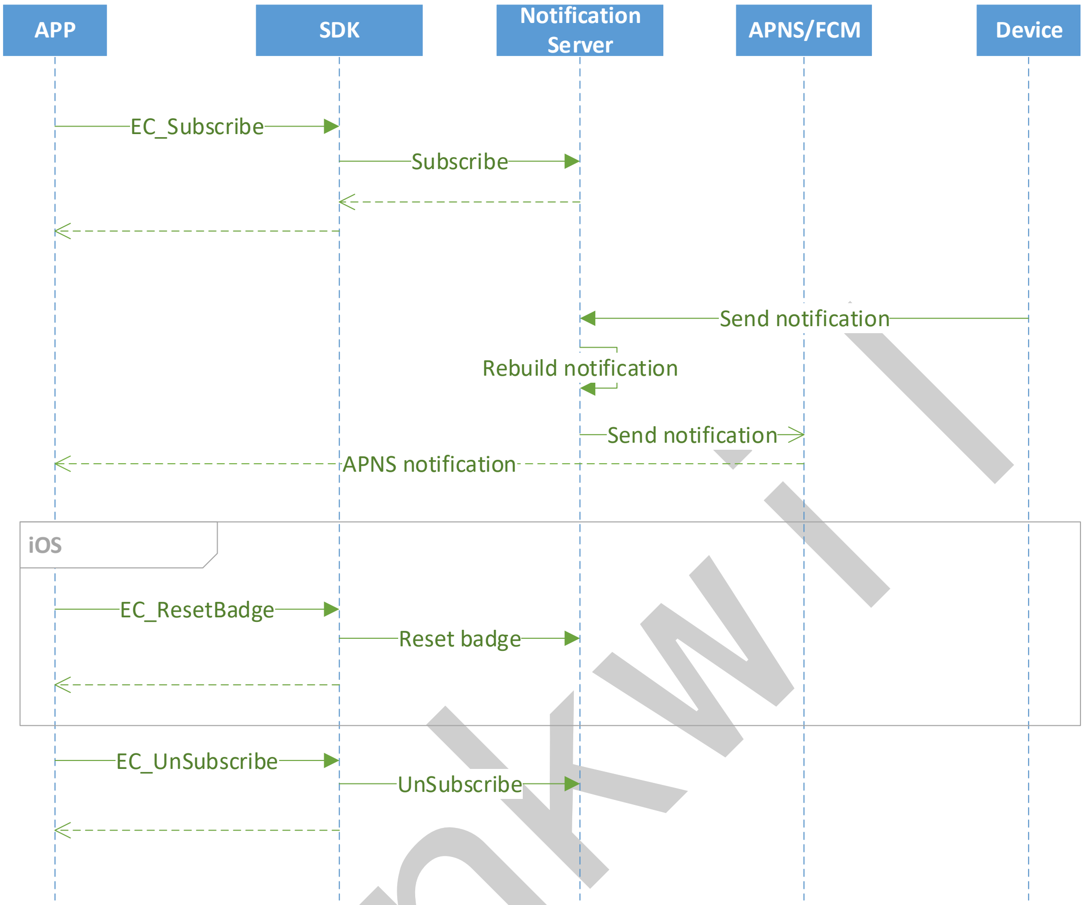
Remote playback



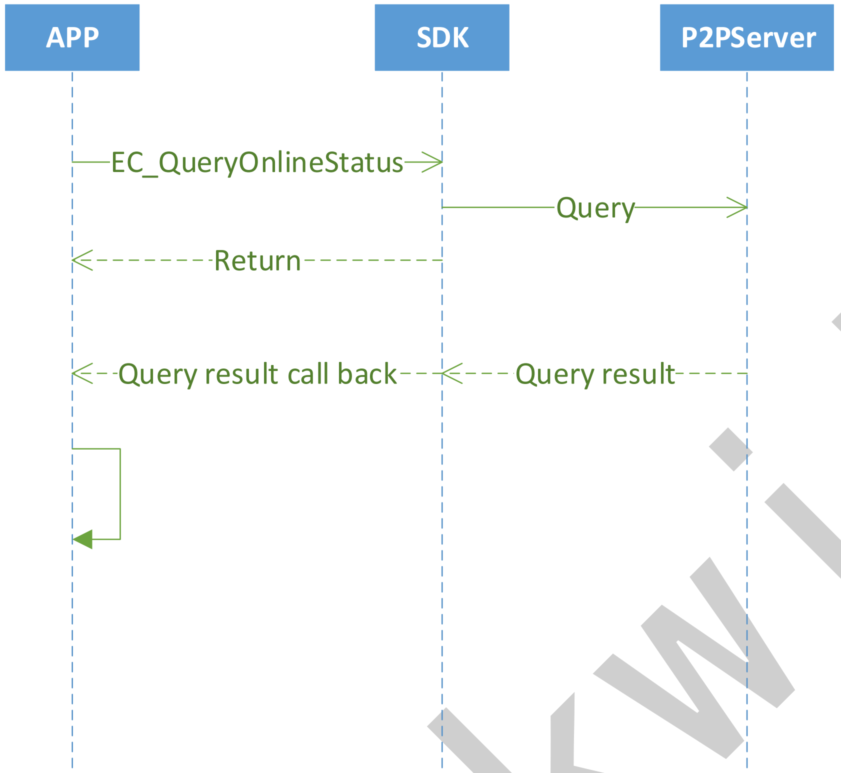
Talk with remote device



Notifications



Query device online status



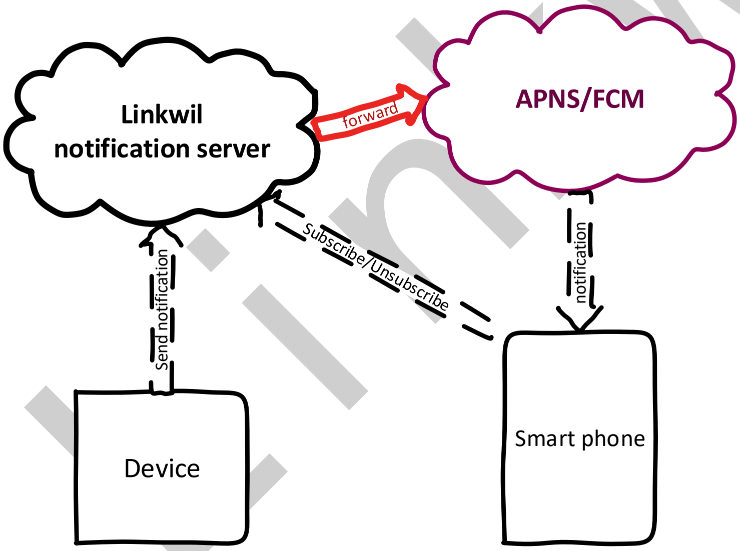
Other topics

Two-way audio communication

As we known, both APP side and device side need to support echo cancelation when talking, Linkwil's product Doorbell M8, Doorbell M6 and IP Camera A6 have already supported echo cancelation on device side, WeHome SDK doesn't contain audio echo cancelation part, developers need to implement echo cancelation at APP side by themselves if they want two-way communication.

Notifications

The device will send notifications to APP who subscribed the notification when some event triggered. The notification system works as the following diagram.



- Smart phone sends the token to Linkwil notification server to subscribe notification.
- Once some event triggered on the device, the device will send notification to Linkwil notification server.
- The notification server rebuild message and forward to APNS/FCM
- APNS or FCM send notifications to smart phone.

[Notification payload]
The decoded payload consists of the following the following fields:

Field	Description
uid	UID of the device
time	UTC time of the event
devType	Device type: 0=Camera 1=Doorbell
notificationType	Notification type: 0=Doorbell pressed 1=Motion detection 2=Unused 3=Device move alert 4=Battery low alert 5=Firmware upgrade result
recordId	Record ID of this event, can be used to play remote record by event.

[Multi language]

For android, developers can show notifications on APP according to the notification type.

For iOS, developers need to define the following ids in localizable.strings files:

```
"RING_NOTIFICATION_TITLE_LOC" = "Doorbell call";
"RING_NOTIFICATION_LOC" = "Somebody pressed your doorbell(%@)";
"PIR_NOTIFICATION_TITLE_LOC" = "Motion alert";
"PIR_NOTIFICATION_LOC" = "(%@) detected motion at:%@";
"UPGRADE_SUCCESS_NOTIFICATION_TITLE_LOC" = "Firmware upgrade result";
"UPGRADE_SUCCESS_NOTIFICATION_LOC" = "(%@) had been upgraded to version:%@";
"UPGRADE_FAIL_NOTIFICATION_TITLE_LOC" = "Firmware upgrade result";
"UPGRADE_FAIL_NOTIFICATION_LOC" = "(%@)upgrade failed, error:%@";
"UPGRADE_CREATE_TASK_FAIL_LOC" = "(%@)upgrade failed, error:Create upgrade task failed";
"UPGRADE_DOWNLOAD_FAIL_LOC" = "(%@)upgrade failed, error:Download upgrade file failed";
"UPGRADE_READ_FIRMWARE_FAIL_LOC" = "(%@)upgrade failed, error:Read upgrade file failed";
"UPGRADE_NO_NEED_UPGRADE_LOC" = "(%@)upgrade failed, error:No need upgrade";
"UPGRADE_MD5_CHECK_FAIL_LOC" = "(%@)upgrade failed, error:MD5 check failed";
"UPGRADE_BATTERY_IS_LOW_LOC" = "(%@)upgrade failed, error:Battery is low";
"START_PROTECTED_NOTIFICATION_TITLE_LOC" = "Under protection";
"START_PROTECTED_NOTIFICATION_LOC" = "(%@) is under lost protection now";
"REMOVED_NOTIFICATION_TITLE_LOC" = "Device lost alert";
"REMOVED_NOTIFICATION_LOC" = "(%@) had been removed from its bracket";
"BAT_LOW_NOTIFICATION_TITLE_LOC" = "Battery low alert";
"BAT_LOW_NOTIFICATION_LOC" = "(%@) battery is low, please charge or change battery as soon as possible";
"BAT_VERY_LOW_NOTIFICATION_TITLE_LOC" = "Battery very low alert";
"BAT_VERY_LOW_NOTIFICATION_LOC" = "(%@) battery is very low, the device will power down later, please recharge it";
```

For details, please refer to:
https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/generating_a_remote_notification