**jf -ipc-sdk development manual**

# Modification history

|  |  |  |
| --- | --- | --- |
| date | Modify the description | editor |
| 2022. 05 .08 | first draft | R&D |
| 2022. 10 .15 | Supplement some alarms and configurations, etc. | R&D |
| 2023. 06.18 | Adjust firmware upgrade related descriptions, etc. | R&D |
| 2024. 03 .14 | Supplementary content related to private file systems, etc. | R&D |
| 2024. 09 .20 | Supplemental Operations | R&D |
| 2024.10.31 | Add video intercom | R&D |
|  |  |  |

# Head record

[jf -ipc-sdk development manual 1](#_Toc161394023)

[1 Modification history 2](#_Toc161394024)

[2 Head record 3](#_Toc161394025)

[3 Preface 6](#_Toc161394026)

[3.1 Documentation purpose 6](#_Toc161394027)

[3.2 SDK Overview 6](#_Toc161394028)

[3.3 Reading object 6](#_Toc161394029)

[3.4 Glossary 6](#_Toc161394030)

[3.5 System Requirements 6](#_Toc161394031)

[4 Quick Start 7](#_Toc161394032)

[4.1 Download SDK 7](#_Toc161394033)

[4.2 Registered account 7](#_Toc161394034)

[4.3 Create product 7](#_Toc161394035)

[4.3.1 Create your own product 7](#_Toc161394036)

[4.3.2 Select attributes 7](#_Toc161394037)

[4.3.3 Check out the product 8](#_Toc161394038)

[4.4 Obtain authorization code 8](#_Toc161394039)

[4.5 Install "iCSee" app 9](#_Toc161394040)

[4.6 Start the experience 9](#_Toc161394041)

[5 Programming guide 10](#_Toc161394042)

[5.1 SDK directory structure 10](#_Toc161394043)

[5.2 SDK Structure diagram 10](#_Toc161394044)

[5.3 Development considerations 11](#_Toc161394045)

[5.4 Introduction to header files 11](#_Toc161394046)

[6 Business Process 12](#_Toc161394047)

[6.1 Initialization 12](#_Toc161394048)

[6.2 Binding 12](#_Toc161394049)

[6.2.1 WIFI QR code distribution network 12](#_Toc161394050)

[6.2.2 Wired distribution network process 13](#_Toc161394051)

[6.3 Alarm push 14](#_Toc161394052)

[6.4 PTZ 14](#_Toc161394053)

[6.5 Config 14](#_Toc161394054)

[6.6 Preview process 14](#_Toc161394055)

[6.7 Voice intercom 15](#_Toc161394056)

[6.8 Record playback 16](#_Toc161394057)

[6.8.1 Playback with FAT32 16](#_Toc161394058)

[6.8.2 Playback with private file system 16](#_Toc161394059)

[6.9 Record Storage 17](#_Toc161394060)

[6.9.1 Cloud storage 17](#_Toc161394061)

[6.9.2 Local storage 17](#_Toc161394062)

[6.10 Cloud Upgrade 17](#_Toc161394063)

[6.10.1 Ugrade call back 17](#_Toc161394064)

[6.10.2 Push upgrade progress 18](#_Toc161394065)

[6.10.3 Firmware upgrade type 18](#_Toc161394066)

[6.11 Sample code 20](#_Toc161394067)

[7 Interface definition 21](#_Toc161394068)

[7.1 jfviot\_api.h 21](#_Toc161394069)

[7.1.1 jfviot\_get\_dev\_info\_callback 21](#_Toc161394070)

[7.1.2 jfviot\_dev\_manager\_callback 21](#_Toc161394071)

[7.1.3 jfviot\_disk\_opr\_callback 22](#_Toc161394072)

[7.1.4 jfviot\_ptz\_control\_callback 22](#_Toc161394073)

[7.1.5 jfviot\_audio\_play\_cmd\_callback 23](#_Toc161394074)

[7.1.6 jfviot\_upgrade\_callback 23](#_Toc161394075)

[7.1.7 jfviot\_getconfig\_callback 24](#_Toc161394076)

[7.1.8 jfviot\_setconfig\_callback 24](#_Toc161394077)

[7.1.9 jfviot\_init\_params 25](#_Toc161394078)

[7.1.10 jfviot\_start 25](#_Toc161394079)

[7.1.11 jfviot\_stop 26](#_Toc161394080)

[7.1.12 jfviot\_start\_ip\_search 26](#_Toc161394081)

[7.1.13 Data Type 27](#_Toc161394082)

[7.2 jfviot\_bind.h 29](#_Toc161394083)

[7.2.1 jfviot\_get\_wifi\_info\_from\_link\_qrcode 29](#_Toc161394084)

[7.3 jfviot\_config.h​​ 29](#_Toc161394085)

[7.3.1 Data Type 29](#_Toc161394086)

[7.4 jfviot\_dev.h​​ 34](#_Toc161394087)

[7.4.1 Data Type 34](#_Toc161394088)

[7.5 jfviot\_disk.h​​ 37](#_Toc161394089)

[7.5.1 Data Type 37](#_Toc161394090)

[7.6 jfviot\_error\_code .h​ 38](#_Toc161394091)

[7.6.1 Data Type 38](#_Toc161394092)

[7.7 jfviot\_event.h​​ 38](#_Toc161394093)

[7.7.1 jfviot\_push\_alarm\_info 38](#_Toc161394094)

[7.7.2 Data Type 39](#_Toc161394095)

[7.8 jfviot\_media.h 41](#_Toc161394096)

[7.8.1 jfviot\_request\_I\_frame\_cb 41](#_Toc161394097)

[7.8.2 jfviot\_stream\_buffer\_init 41](#_Toc161394098)

[7.8.3 jfviot\_push\_video\_data 42](#_Toc161394099)

[7.8.4 Data Type 42](#_Toc161394100)

[7.9 jfviot\_ptz.h 49](#_Toc161394101)

[7.9.1 Data Type 49](#_Toc161394102)

[7.10 jfviot\_record.h 51](#_Toc161394103)

[7.10.1 jfviot\_get\_file\_list 51](#_Toc161394104)

[7.10.2 jfviot\_file\_search\_calendar 52](#_Toc161394105)

[7.10.3 jfviot\_file\_playback\_start 53](#_Toc161394106)

[7.10.4 jfviot\_file\_playback\_start\_by\_time 53](#_Toc161394107)

[7.10.5 jfviot\_file\_playback\_goto\_time 54](#_Toc161394108)

[7.10.6 jfviot\_file\_playback\_stop 54](#_Toc161394109)

[7.10.7 jfviot\_playback\_file\_send\_data 55](#_Toc161394110)

[7.10.8 jfviot\_file\_playback\_end 55](#_Toc161394111)

[7.10.9 Data Type 56](#_Toc161394112)

[7.11 jfviot\_ upgrade .h 57](#_Toc161394113)

[7.11.1 jfviot\_push\_upgrade\_progress 57](#_Toc161394114)

[7.11.2 Data Type 57](#_Toc161394115)

# Preface

## Documentation purpose

This document mainly introduces jf-ipc-sdk functions, framework, processes and interface definitions, etc., to assist users to quickly develop their own firmware based on jf-ipc-sdk and access our company's cloud services.

## SDK Overview

jf-ipc-sdk is a set of function interfaces. Users call relevant interfaces, pass in video data and related configuration

information, generate executable programs, and then use them on embedded devices with specific CPUs and specific operating systems. Finally, they can be used through iCSee apps, VMS and other tools to preview videos and modify configurations. If the manufacturer has the ability, they can also use our FunSDK to develop their own APP.

jf-ipc-sdk is aimed at companies that have the ability to develop camera audio and video functions, but are not good at cloud server deployment and function development, but want users to remotely view camera videos on mobile phones or PCs.

Our cloud server supports value-added functions such as cloud storage and intelligent analysis, and can share value-added benefits with equipment manufacturers using jf-ipc-sdk .

## Reading object

This document is intended for device embedded software developers who have certain development capabilities and understand the C/C++ language.

## Glossary

iCSee: A supporting APP for video preview, PTZ control and other functions.

VMS: Desktop client software for monitoring.

Local Storage: Local storage, store the video and audio in the local SD card.

Cloud Storage: Cloud storage, which stores video and audio on the cloud and can be downloaded and played back through the APP.

JFVIOT\_EN\_FS\_FAT32: A standard file system fat32.

JFVIOT\_EN\_FS\_PRIVATE\_FS: Private file system, private format, better security and higher performance.

QR code: QR code, used for wifi distribution network.

## System Requirements

Currently, the Linux operating system is compiled with gcc and arm- xm - linux compilers , and customers can provide their own compilers when needed.

# Quick Start

## [Download SDK](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=20d777495bd84b069917bfa7c5c4cfac&lang=en" \l "docs-hash-1)

Download the SDK package applicable to your device (if there is no SDK package corresponding to the compiler, please send an email to [jf\_ipc\_sdk@jftech.com](mailto:jf_ipc_sdk@jftech.com" \t "_blank) And attach contact information and download path of cross compiler)  
Download path: [https://github.com/jftek/jf-ipc-sdk](https://github.com/jftek/jf-ipc-sdk" \t "_blank)

## [Registered account](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=20d777495bd84b069917bfa7c5c4cfac&lang=en" \l "docs-hash-2)

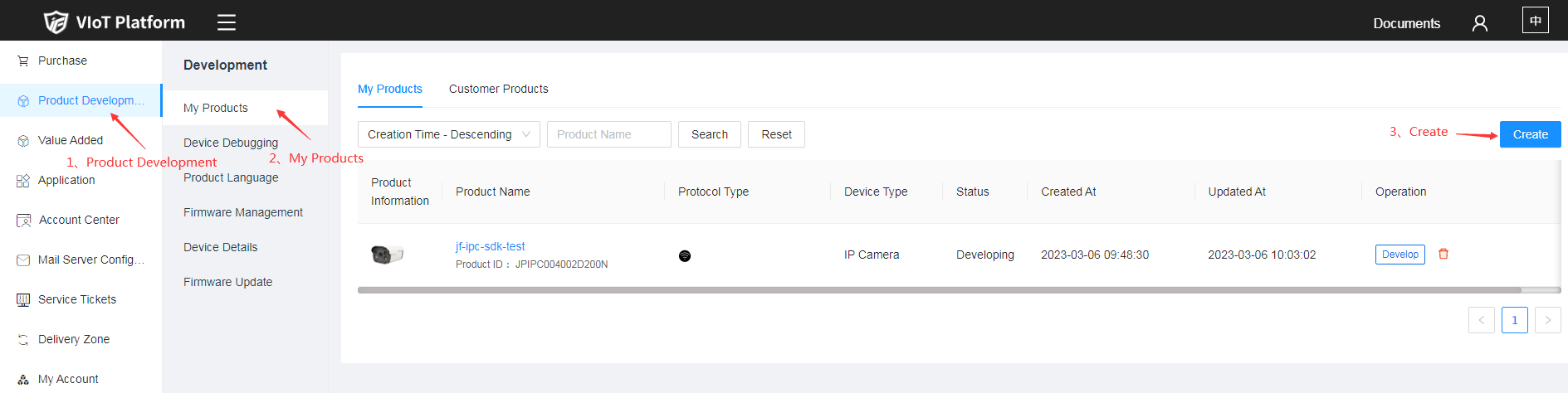
First, you need to apply for a [Jiefeng VIoT development platform](https://auth.jftech.com/" \l "/" \t "_blank) account, used to create and manage your cloud device, and more IOT extension functions.

Tips:

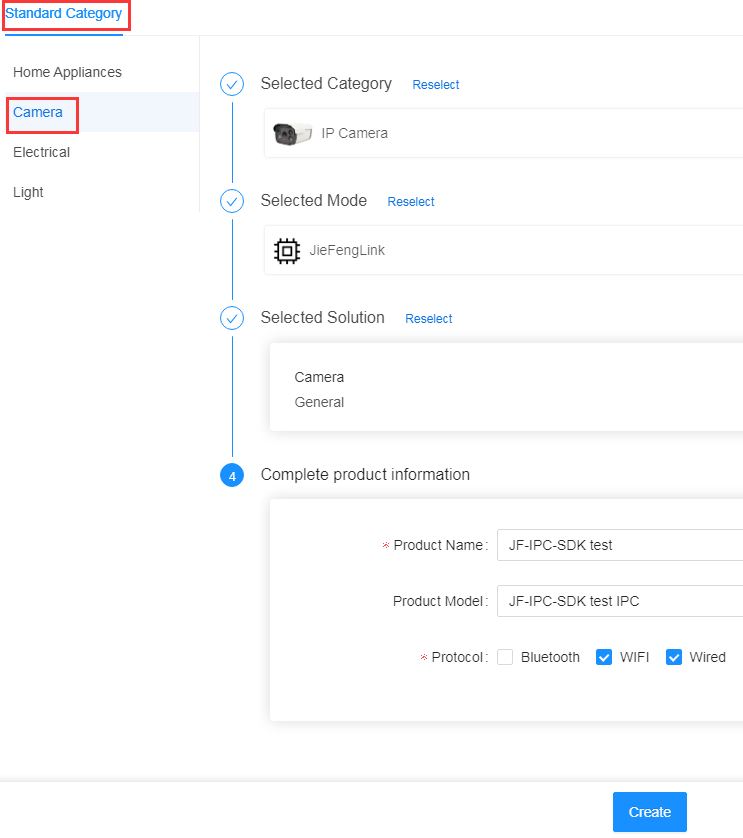
The account you registered has the highest permissions. Please keep your account properly to avoid security risks arising from the hardware products, APPs, devices or user data you create.

## [Create product](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=20d777495bd84b069917bfa7c5c4cfac&lang=en" \l "docs-hash-3)

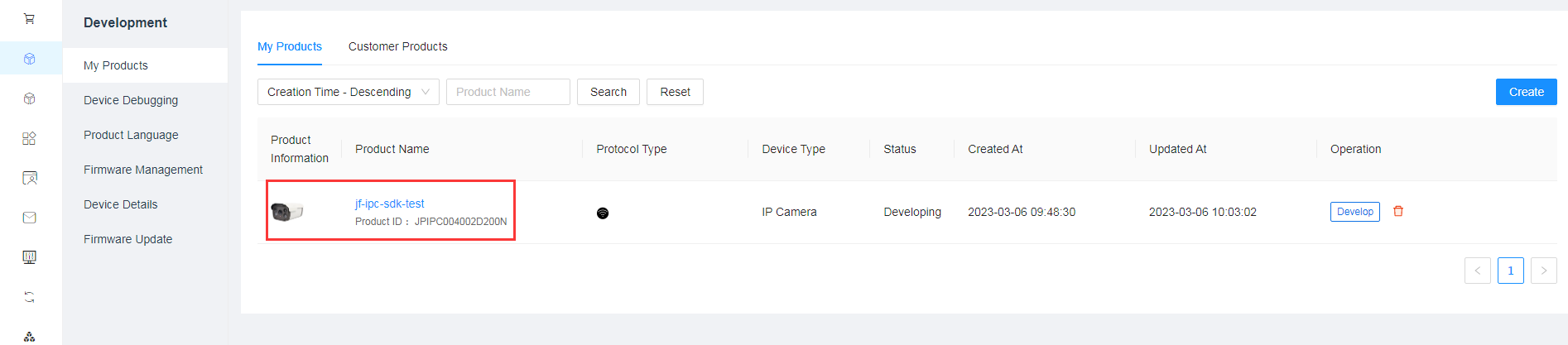
### Create your own product

<https://viot.jftech.com/product> The menu may be adjusted, please refer to the actual situation.select Product Development -> My Products -> Create  


### Select attributes

Select the Standard Category - camera, and select the basic attributes for the product  


### Check out the product

After the product is created successfully, you can see the corresponding product and product pid in the "My Products" interface  


## [Obtain authorization code](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=20d777495bd84b069917bfa7c5c4cfac&lang=en" \l "docs-hash-4)

After entering the "Develop" interface, you can apply for the product authorization code to connect to the ECS

Get authorization code for free: your corresponding product can apply for two authorization codes for free.  
Download of production documents: The authorization codes you purchased and applied for are uniformly downloaded and obtained here.Purchase authorization code: You can purchase additional authorization code for testing here or make a formal order to obtain the batch authorization code.

## Install "iCSee" app



## [Start the experience](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=20d777495bd84b069917bfa7c5c4cfac&lang=en" \l "docs-hash-6)

At this time, you only need to open the demo.c file and fill in the authorization code to start experiencing cloud services, preview the test video, and understand the App functions.

strcpy(init\_params.uuid, "");

strcpy ( init\_params.auth\_key , "");

strcpy ( init\_params.pid , "");

strcpy ( init\_params.pMac , ""); //format as 00:11:11:11:11:11

# Programming guide

## SDK directory structure

The directory structure of the SDK release package is as follows. After getting the sdk , please check the README. md first .

jf\_ipc\_sdk\_AAA\_BBB\_1.0.0

├── demo

│ ├─── demo.c //demo

│ ├─── config //config

│ ├─── Makefile //makefile

│ ├─── video.h264 //test video file

│ ├─── audio.g711a //test audio file

│ ├─── scene.jpg //alarm picure

│ └─── QrCodeInfo.txt //QrCodeInfo for wifi

│

├── include //head file

│ └── jfviot

│

├── libs //lib

│ └── libjfviot\_ipc.a

│

├── README.md //help document in english

├── README\_chn.md // help document in chinese

└── CHANGELOG.md

AAA : operating system

BBB : cross compiler

## Compile Demo And Clean

Go to the demo directory.Use make to compile to produce an application with a jfipcsdk\_demo name.

Download the jfipcsdk\_demo and the files in the demo path together in the environment where the demo program is running, and if you need to change the path of the file, you need to modify the content in demo.c to be consistent

Run the application in the same directory as jfipcsdk\_demo ./jfipcsdk\_demo If there is no permission to execute chmod +x jfipcsdk\_demo.

## SDK Structure diagram

In addition to the module for connecting to the cloud platform, the internal SDK is mainly implemented in the NetIp module. The NetIP module can also connect to the client of the LAN. Currently, there are some restrictions and communication can be made if necessary. Local​ storage When local storage is initialized to private file system, storage and playback are all implemented internally by us. When initialized to standard FAT32, recording needs to be implemented externally. This module does not work. External dependencies are mainly callbacks defined by the SDK that require external implementation.



## Development considerations

Please pay attention to the following content during the development process to avoid unnecessary problems :

1. Do not block all callback interfaces. Blocking the callback interface will affect the normal operation of the entire SDK process.

2. Minimize time-consuming operations in the callback interface. The shorter the time-consuming operation, the better .

3. When sending the video, there is no speed control, and the playback speed is controlled by the client. A pause command will be sent when necessary. When paused, the video sending data interface will be stuck.

4. Pay special attention to the iFsType field during initialization. If it is initialized to the private file system JFVIOT\_EN\_FS\_PRIVATE\_FS, then all video storage and playback related functions are completed internally by the SDK. Do not operate the hard disk externally.

## Introduction to header files

The header file defines the interface of each module. See Chapter 7 Interface Definition for details.

1. jfviot\_api.h

System initialization and registration of callback functions.

1. jfviot\_bind.h

WiFi distribution network related interfaces.

1. jfviot\_config.h

Some configurations are defined, currently mainly including encoding configuration, osd configuration, network configuration, lighting mode configuration, etc.

1. jfviot\_dev.h

Enumerations and structures related to device operations, such as restarting and restoring defaults.

1. jfviot\_disk.h

Enumerations and structures related to hard disk and file system operations .

1. jfviot\_error\_code.h

Various error codes.

1. jfviot\_event.h

Alarm event notification related interfaces and structures

1. jfviot\_media.h

Stores interfaces and structures related to real-time audio, video and intercom modules.

1. jfviot\_ptz.h

Common PTZ control related interfaces.

1. jfviot\_record.h

Video playback related functions support playback by time period and file.

1. jfviot\_upgrade.h

Upgrade related, see 6.8 for details of the process .

# Business Process

## Initialization

For system initialization, refer to [jfviot\_init\_params](#_jfviot_init_params)(). Pay special attention to the iFsType field during initialization. If initialized to a private file system, all storage, playback, and hard disk management will be completed by the SDK, which will be more efficient and more confidential, but users cannot manage related resources by themselves. . Initialization completes the registration of the main callback function.

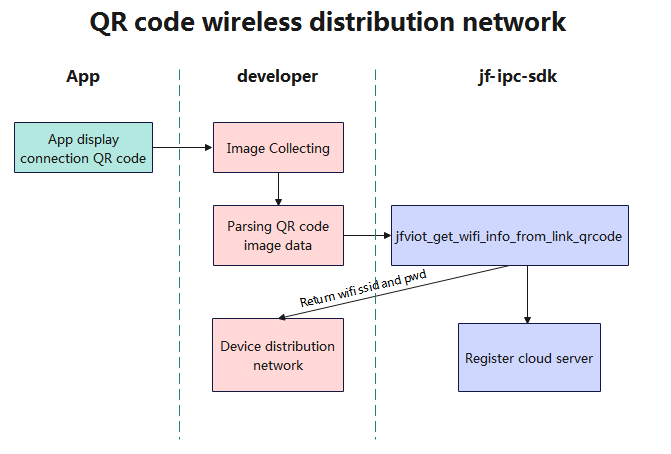
To connect to the cloud, please refer to [jfviot\_start](#_jfviot_start)().This interface will start the network function of jf-ipc-sdk and try to connect to JF Cloud.

To enable local search, please refer to [jfviot\_start\_ip\_search](#_jfviot_start_ip_search) (). Download the DeviceManager supporting tool to support the IP search and IP modification functions. The SDK will obtain or set the device IP from the JFVIOT\_EN\_CONFIG\_NETCOMMON in the setting and obtain configuration callback.

## Binding

### WIFI QR code distribution network

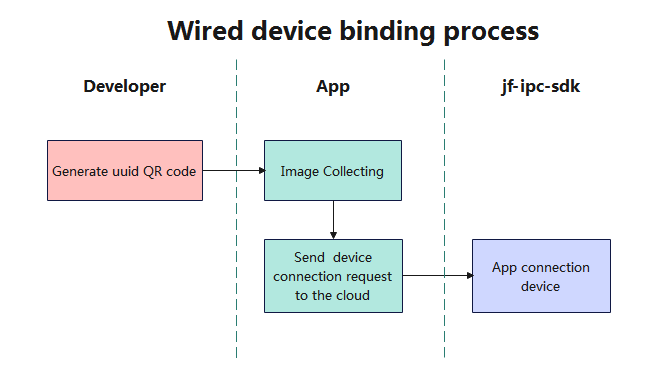
transfer jfviot\_get\_wifi\_info\_from\_link\_qrcode interface for network configuration. You need to implement the function of the device to collect and parse the App QR code image information by yourself, and pass the parsed data into this interface. This interface will return the wifi username and password, and developers need to implement the device network configuration function by themselves. After the network configuration is successful, the App will automatically bind the device to its account.



### Wired distribution network process

You need to use the iCSee app to scan the QR code generated by the device serial number to bind the device. During product production, the QR code label can be affixed to the product shell to facilitate the addition of the device to the App.

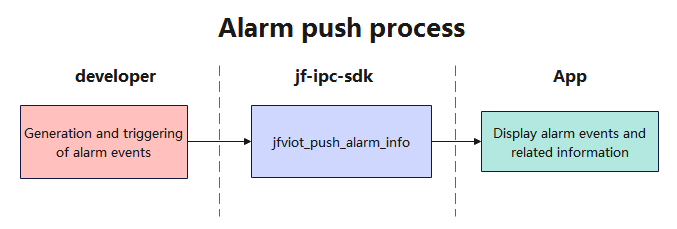
You also can use the uuid to create the QR code by some websites liking https://cli.im/.



## Alarm push

Using [jfviot\_push\_alarm\_info](#_jfviot_push_alarm_info)() interface. Developers need to obtain the alarm information by themselves, assign values according to the structure parameters and pass them into this interface.

After successfully calling this interface, the mobile phone will receive a notification and enter the "Cloud Events" interface, where you can see the pushed alarm image. If it is humanoid detection, the SDK will record the alarm video according to the alarm status. It will record when the alarm starts and stop recording 15 seconds after the alarm ends.



## PTZ

The PTZ control adopts the message callback method [jfviot\_ptz\_control\_callback](#_jfviot_ptz_control_callback)(). The App issues the control command and notifies the developer through enumeration.

Developers need to implement the rotation function of PTZ by themselves according to different enumeration values of [jfviot\_en\_ptz\_operate](#_jfviot_en_ptz_operate).

## Config

The user needs to handle the callback [jfviot\_getconfig\_callback()](#_jfviot_getconfig_callback) and [jfviot\_setconfig\_callback()](#_jfviot_setconfig_callback).

The details of the configuration are defined in the [jfviot\_config.h](#_jfviot_config.h​​) file.

## Preview process

External applications only need to directly pass in audio and video data after initialization. The application side does not need to care about how the app and SDK interact. Refer to [jfviot\_stream\_buffer\_init()](#_jfviot_stream_buffer_init) and [jfviot\_push\_video\_data()](#_jfviot_push_video_data).

The developer needs to create a thread to control the encoder to continuously provide the stream data, fill in the jfviot\_st\_stream\_params structure parameter according to the stream frame information, and pass it into the jfviot\_pass\_video\_data interface.

Developers can also choose to call jfviot\_stream\_buffer\_init before calling the stream push interface. The SDK will set the size of the stream buffer according to the parameters in it. The registered mandatory I frame callback jfviot\_request\_I\_frame\_cb will help the app get the graph display faster.

The intercom stream callback is used to receive the voice data sent by the app. The developer needs to implement the audio playback function according to the callback data.



## Voice intercom

The user only needs to handle the callback. [jfviot\_audio\_play\_cmd\_callback](#_jfviot_audio_play_cmd_callback)()



## Video intercom

The user needs to handle the callback. [jfviot\_video\_play\_cmd\_callback](#_jfviot_video_play_cmd_callback)() and [jfviot\_get\_ability\_callback](#_jfviot_get_ability_callback)()

Jfviot\_get\_ability\_callback() is a callback function used to reply to capability requests sent by the app to the device. Developers need to fill in the device capabilities in the specified JSON format and use the entire JSON string as the return value.

Jfviot\_video\_play\_cmd\_callback() is a callback function used to process data streams in video calls. Depending on the type of operation, functions can perform different operations, such as starting and stopping video calls, receiving audio and video data, etc. When using this interface, the first step is to start a video call on the app, receive the stream data (demo transmitted in structural form), and then stop recording at the end of the call.

## Record playback

### Playback with FAT32

When the file system is set to FAT32,the playback of the file format is roughly as shown in the figure below. The client first performs a query, and after getting the video file information, it begins to request file playback. Downloading and playback are the same process, but the download speed is not limited. During record playback, commands such as jump are also supported. When the file download is completed, the application must actively send an end flag.



### Playback with private file system

When the file system is set to private file system,the playback of the file format is roughly as shown in the figure below. Developers don’t need to care about anything after initialization.



## Record Storage

### Cloud storage

The application only needs to push video and audio data to the SDK . The specific cloud storage is controlled by the platform and the app. If you purchase a cloud platform package, you can enable the cloud storage function on the app.

### Local storage

When the file system is set to FAT32, local storage is implemented by the developer. When the file system is set to private file system, local storage is implemented by the SDK.

## Cloud Upgrade

### Ugrade call back

See the callback of [jfviot\_upgrade\_callback](#_jfviot_upgrade_callback).

### [Push upgrade progress](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=7dbdf2e4dfd44b16a046ff901c2e9100&lang=en" \l "docs-hash-2)

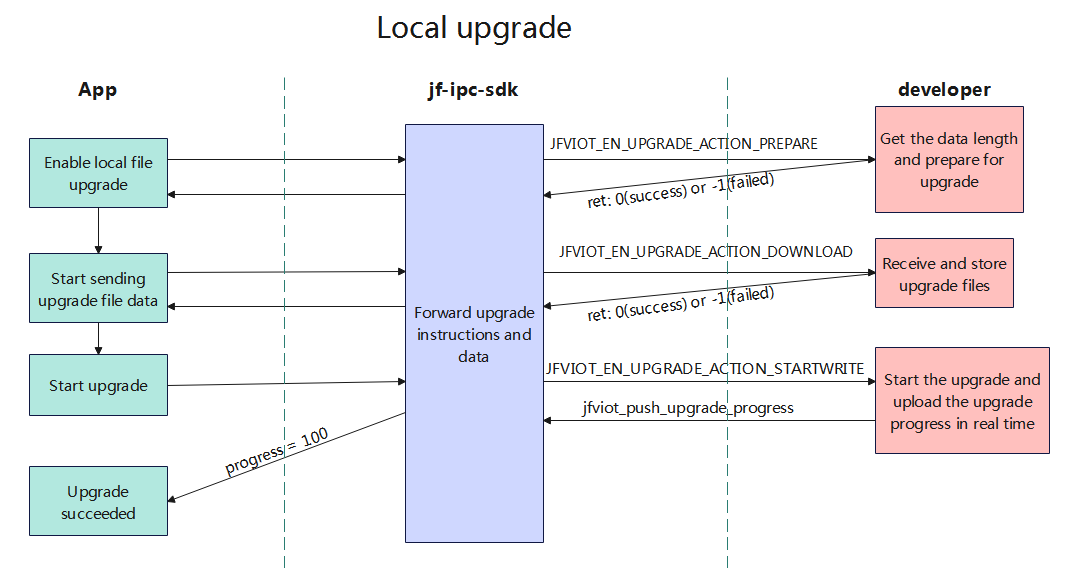
int [jfviot\_push\_upgrade\_progress](#_jfviot_push_upgrade_progress)(int progress);

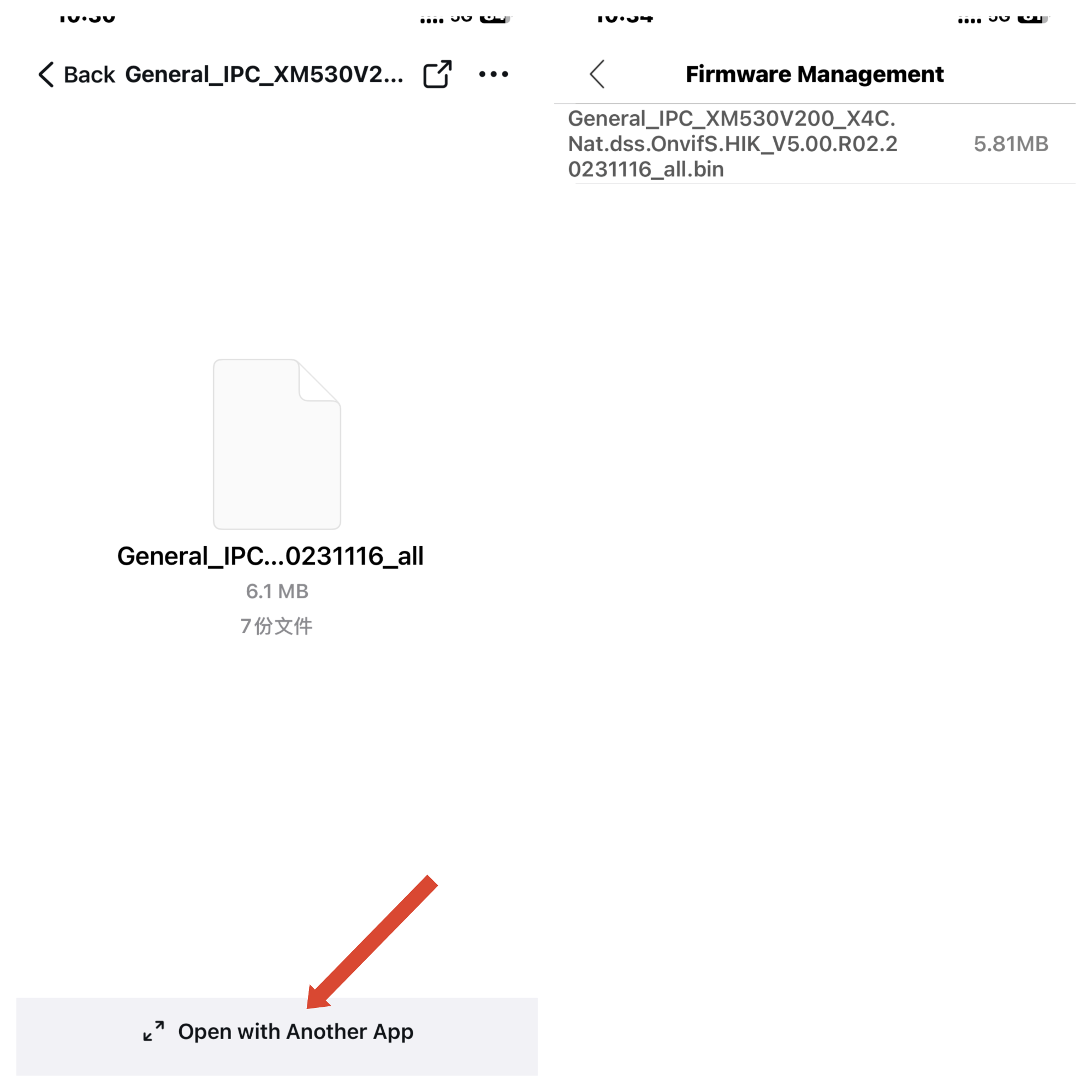
Tips:  
When "JFVIOT\_EN\_UPGRADE\_ACTION\_STARTWRITE" is recognized, it means that the device starts to upgrade. The

developer needs to continuously import the upgrade progress into this interface according to the actual upgrade situation

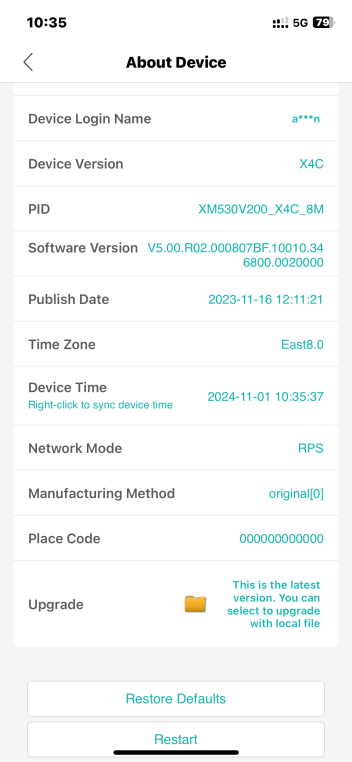
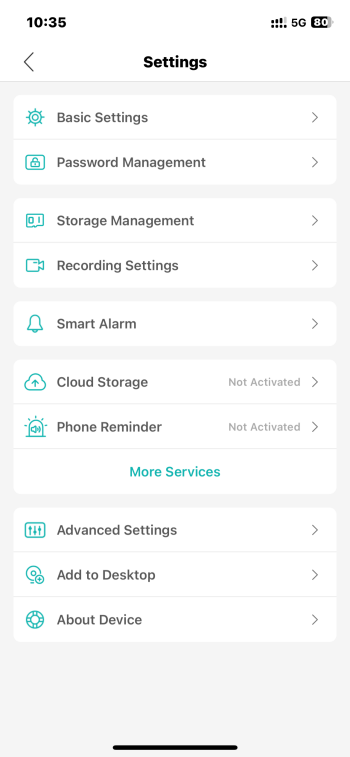
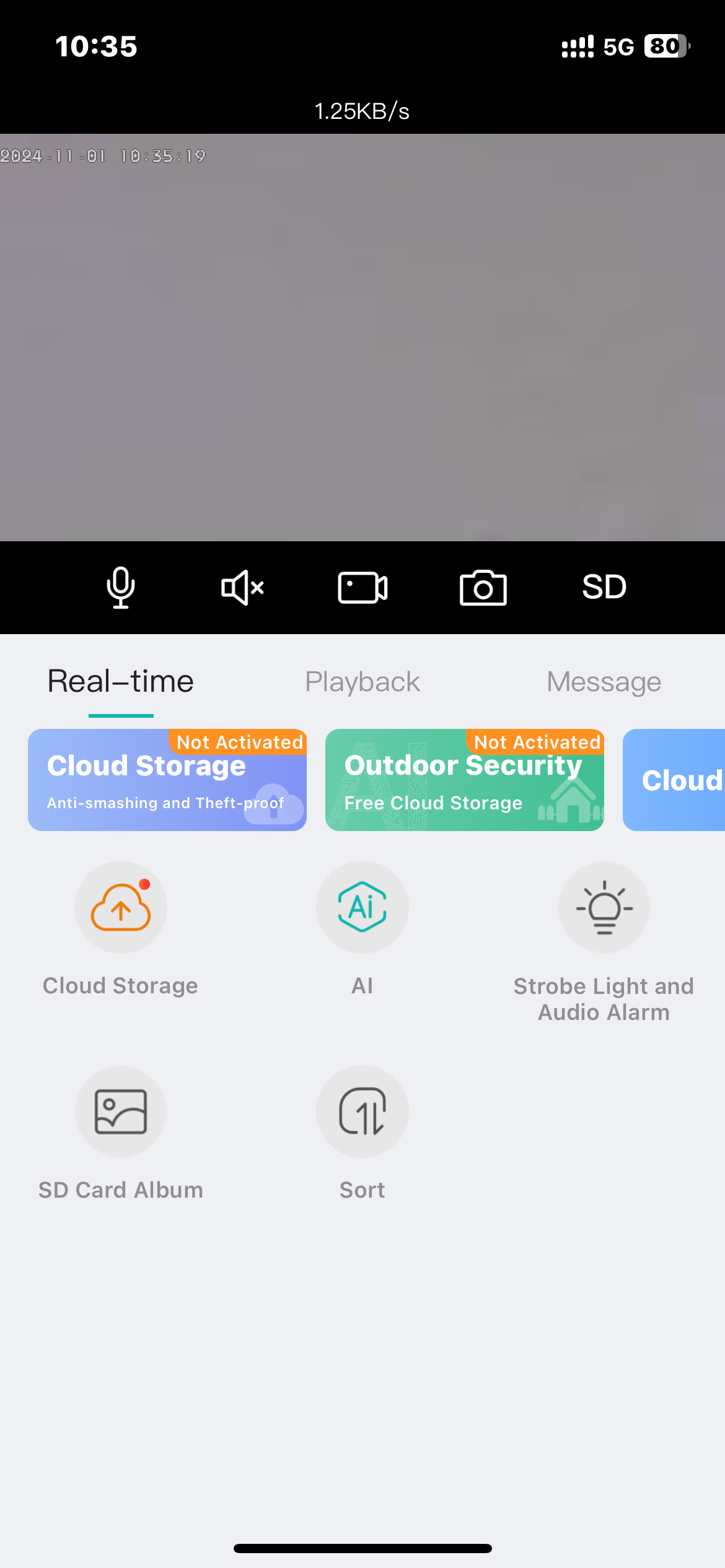
### [Firmware upgrade type](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=7dbdf2e4dfd44b16a046ff901c2e9100&lang=en" \l "docs-hash-3)

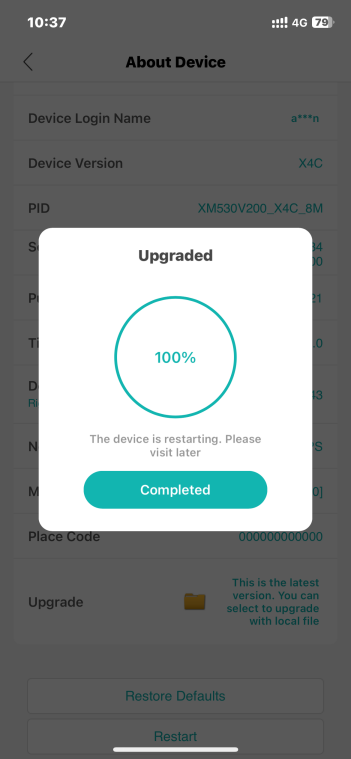
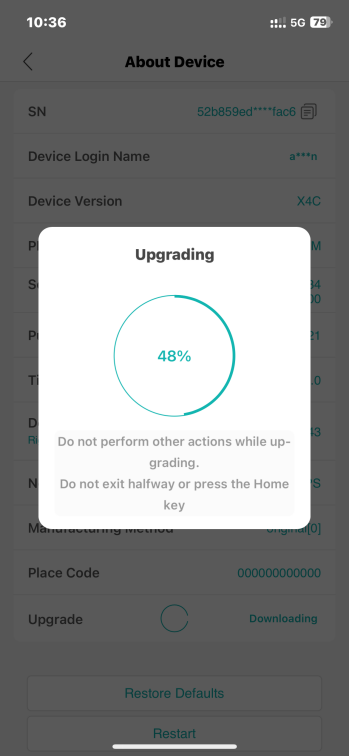
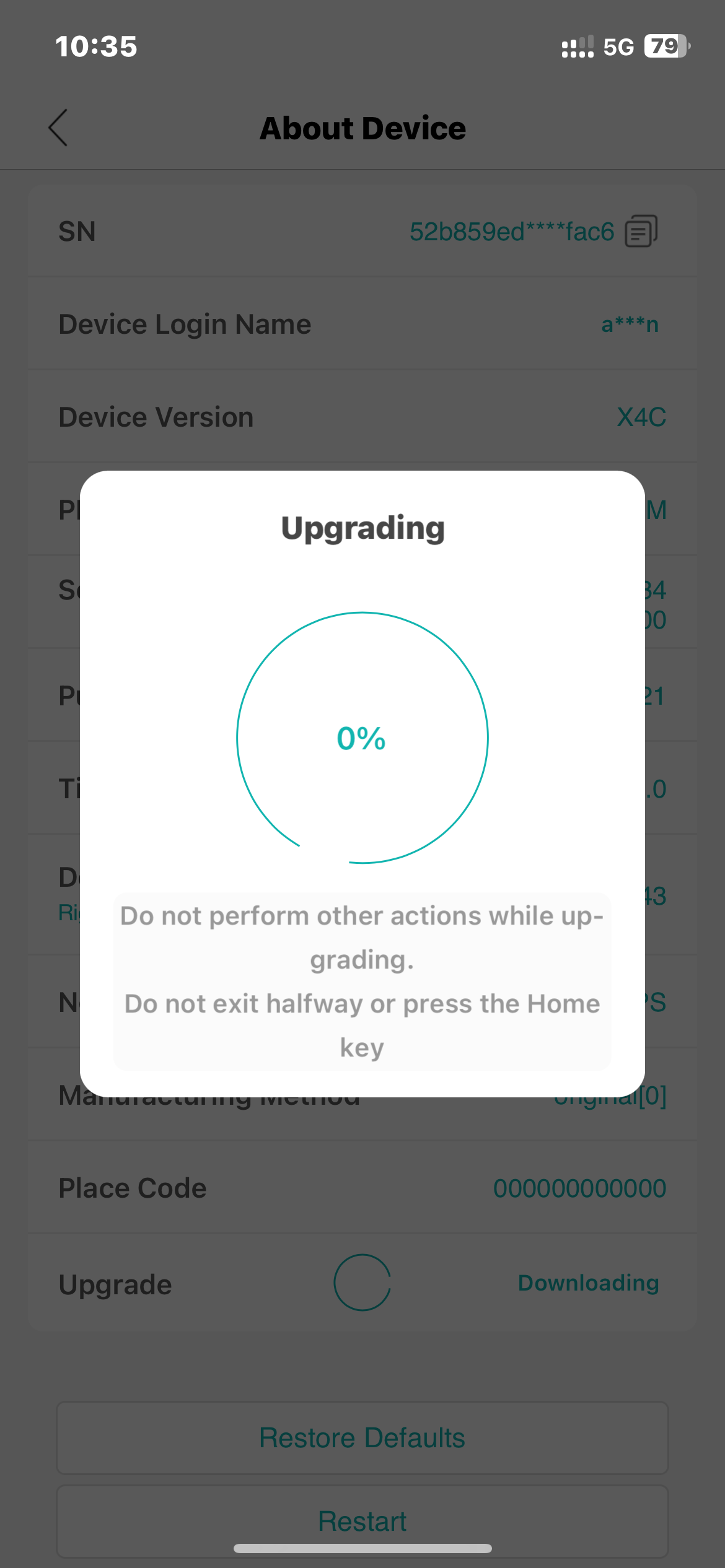
#### [Local upgrade](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=7dbdf2e4dfd44b16a046ff901c2e9100&lang=en" \l "docs-hash-4)

  
Steps:1) Prepare to upgrade the firmware, share or move the firmware to the upgrade directory of iCSee App through file sharing or PC data cable download



2) Find the matching firmware in the devices added by iCSee App and start the upgrade

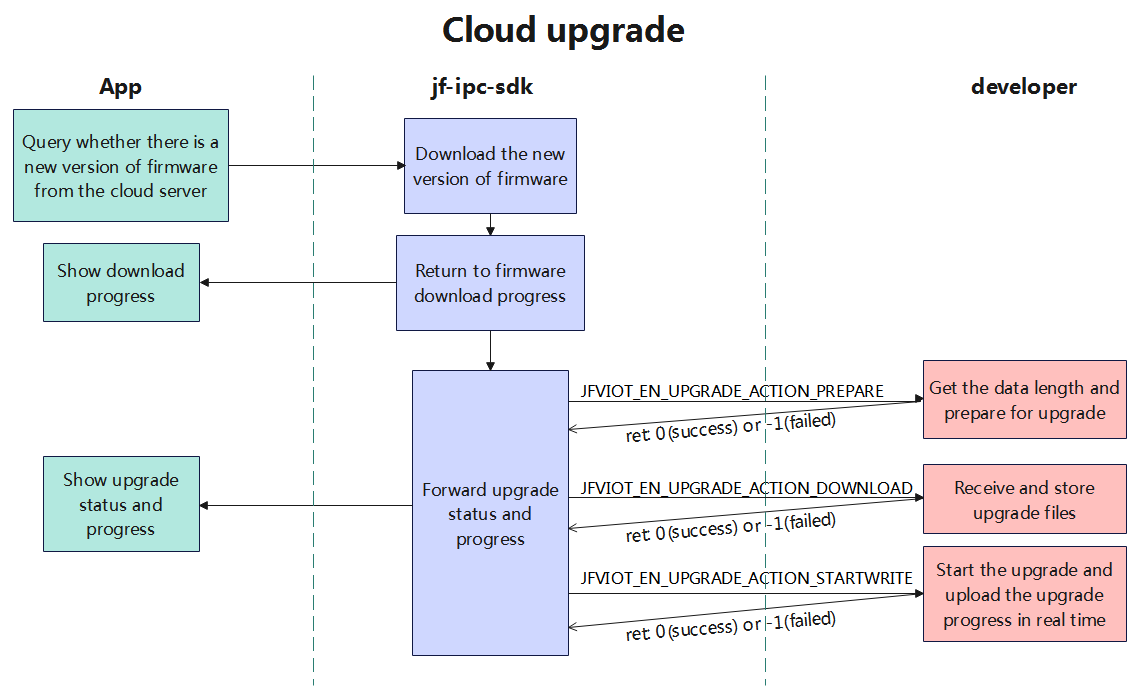




Tips:  
I. If the abort command is generated, the jfviot\_upgrade\_callback registration upgrade callback will receive JFVIOT\_EN\_UPGRADE\_ACTION\_ABORT, which you need to process according to your business needs.

II. At the beginning of the upgrade, jf-ipc-sdk will kick out all client connections and only retain one upgrade connection.

[3.2 Cloud upgrade](https://docs.jftech.com/docs?menusId=82518bf6b9a64411bb61d109bcdbb6fa&siderid=7dbdf2e4dfd44b16a046ff901c2e9100&lang=en" \l "docs-hash-5)

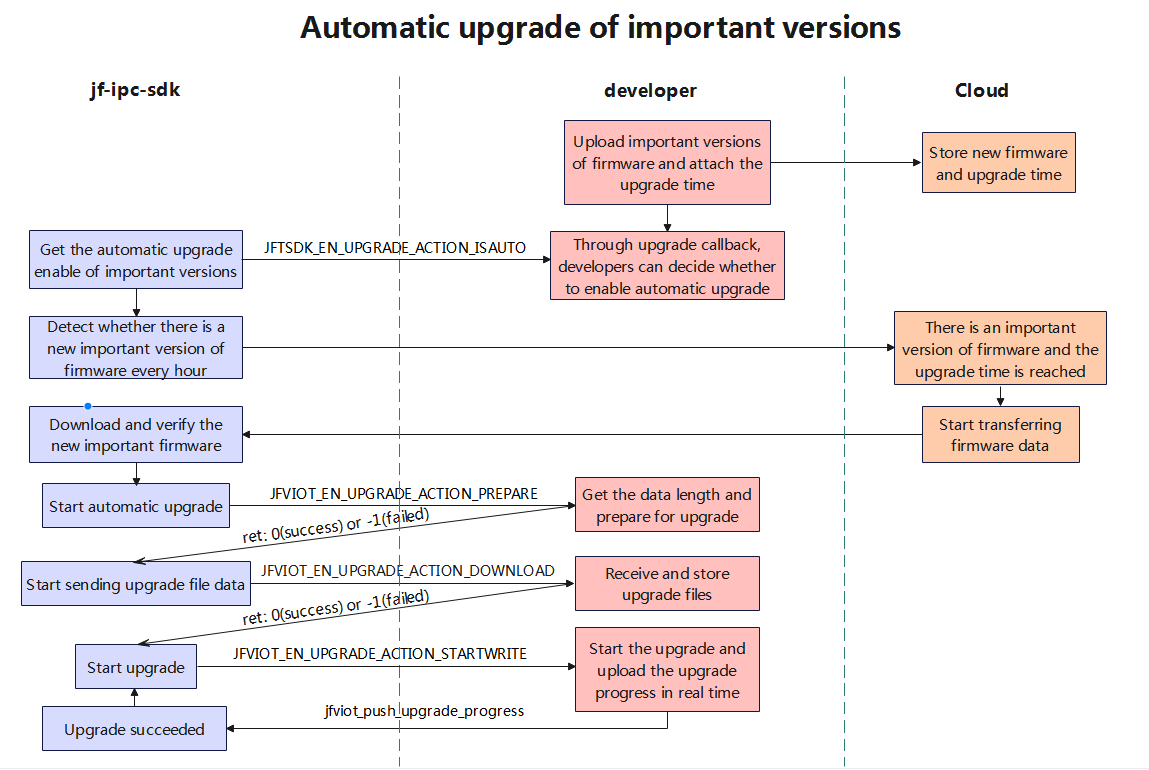
  
Steps:

Tips:  
The cloud upgrade process. The new version of firmware detection is automatically started. It is queried every 12

hours. The customer needs to decide whether to upgrade. When the upgrade is started, the App will display the progress

of the device downloading firmware and the progress of the developer returning the update.

#### Automatic upgrade of important versions

  
Steps:

Tips:   
Compared with cloud upgrade:I. The SDK will query whether the device is enabled for automatic upgrade of

important versions through JFVIOT\_EN\_UPGRADE\_ACTION\_ISAUTO.II. If an important version of firmware is detected, the

upgrade process will automatically start without manual processing by the customer . The upgrade process and processing

are consistent with the cloud upgrade.

## Sample code

See demo for details.

# Interface definition

## jfviot\_api.h

This file contains all other header files. Normal application code only needs to include this one header file. This file

mainly defines the initialization and other interfaces of the entire SDK .

### jfviot\_get\_dev\_info\_callback

【 **FUNCTION】**

Obtaining Device Information.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_get\_dev\_info\_callback)(jfviot\_en\_get\_dev\_info infoType, int channel, void\* data, void\* user\_data);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| infoType | Type of operation | input |
| channel | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| data | Data pointer, which is converted to the corresponding structure based on infoType for processing | input |
| user\_data | User data pointer, temporarily useless | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_dev\_manager\_callback

【 **FUNCTION】**

Device Management.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_dev\_manager\_callback)(jfviot\_en\_dev\_operate OprType, int channel, void\* data, void\* user\_data);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| OprType | Type of operation | input |
| channel | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| data | Data pointer, which is converted to the corresponding structure based on infoType for processing | input |
| user\_data | User data pointer, temporarily useless | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_disk\_opr\_callback

【 **FUNCTION】**

Disk Management.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_disk\_opr\_callback)(jfviot\_en\_disk\_opr\_type OprType, int channel, void\* data, void\* user\_data);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| OprType | Type of operation | input |
| channel | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| data | Data pointer, which is converted to the corresponding structure based on infoType for processing | input |
| user\_data | User data pointer, temporarily useless | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_ptz\_control\_callback

【 **FUNCTION】**

PTZ operation.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_ptz\_control\_callback)(jfviot\_en\_ptz\_operate OprType, int channel, void\* data, void\* user\_data);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| OprType | Type of operation | input |
| channel | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| data | Data pointer, which is converted to the corresponding structure based on infoType for processing | input |
| user\_data | User data pointer, temporarily useless | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_audio\_play\_cmd\_callback

【 **FUNCTION】**

Receive voice intercom data.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_audio\_play\_cmd\_callback)(jfviot\_en\_talk\_opr\_type OprType, int channel, unsigned char\* data, int len);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| OprType | Type of operation | input |
| channel | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| data | Data pointer, audio data | input |
| len | Length of data | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_video\_play\_cmd\_callback

【 **FUNCTION】**

Receive video intercom data.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_video\_play\_cmd\_callback)(jfviot\_en\_videotalk\_opr\_type OprType, unsigned int channel, unsigned char\* data, int len);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| OprType | Type of operation:  JFVIOT\_ENVNet EO\_TALK\_STAR: Start a video call.  JFVIOT\_ENVNet EO\_TALK\_STOP: Stop video call.  JFVIOT\_ENVNet EO\_TALK\_DATA: Receive video or audio data. | input |
| channel | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| data | Audio and video request parameter structure | input |
| len | Length of Structure | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_get\_ability\_callback

【 **FUNCTION】**

Device capability reporting

**【 prototype 】**

typedef const char\* (\*jfviot\_get\_ability\_callback)(jfviot\_en\_ability\_type abilityType);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| OprType | Type of operation:  JFVIOT\_EN\_ABILITY\_DECODER: App requests decoding capability from device | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| s\_Json | Json string for ability |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_upgrade\_callback

【 **FUNCTION】**

Upgrade operation.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_upgrade\_callback)(jfviot\_en\_upgrade\_action OprType, int datalen, void\* data);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| OprType | Type of operation | input |
| datalen | The size of firmware data to be store in memory | input |
| data | Data pointer, according to OprType into the corresponding structure for processing | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_getconfig\_callback

【 **FUNCTION】**

Get configuration.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_getconfig\_callback)(jfviot\_en\_config\_type configType, int channel, void\* data, void\* user\_data);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| configType | Configuration type | input |
| channel | Channel number, - 1 means channel is not distinguished, non-negative number means channel number, for DVR and NVR devices | input |
| data | Data pointer, according to configType into the corresponding structure for processing | output |
| user\_data | User data pointer, temporarily useless | Input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_setconfig\_callback

【 **FUNCTION】**

Set configuration.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_setconfig\_callback)(jfviot\_en\_config\_type configType, int channel, void\* data, void\* user\_data);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| configType | Configuration type | input |
| channel | Channel number, - 1 means channel is not distinguished, non-negative number means channel number, for DVR and NVR devices | input |
| data | Data pointer, according to configType into the corresponding structure for processing | input |
| user\_data | User data pointer, temporarily useless | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_init\_params

【 **FUNCTION】**

SDK initialization function, the first function called after startup.

**【 prototype 】**

JFVIOT\_RET jfviot\_init\_params(jfviot\_st\_init\_params\* init\_params, jfviot\_st\_callback\_funclist\* cb);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| init\_params | Device initialization parameters | input |
| cb | Pointer to the list of callback functions | input |
|  |  |  |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_start

【 **FUNCTION】**

Connecting to the Cloud Server. This interface will start the network function of jf-ipc-sdk and try to connect to JF Cloud.

**【 prototype 】**

JFVIOT\_RET jfviot\_start(const char\* pSDKVersion);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| pSDKVersion | for details,see the JFVIOT\_IPC\_SDK\_VERSION | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE 】**

See demo .

### jfviot\_stop //demo里面没有这个函数了

【 **FUNCTION 】**

During external upgrade, you can call this function to release resources. It is not allowed to call jfviot\_start again after calling it .

**【 prototype 】**

JFVIOT\_RET jfviot\_stop ();

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE 】**

jfviot\_start again after calling it .

**【 EXAMPLE 】**

NONE.

### jfviot\_start\_ip\_search

【 **FUNCTION 】**

start IP search( will listen on port 34569, to realize device discovery and IP modification) .

**【 prototype 】**

JFVIOT\_RET jfviot\_start\_ip\_ search ( );

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### Data Type

#### jfviot\_st\_init\_params

【 **illustrate** 】

The device initializes the structure

【 **definition** 】

typedef struct

{

char uuid[32];

char auth\_key[32];

char pid[20];

char pMac[32];

int iCaptrueChannels;

int iNetChannels;

char pFirmwareVersion[32];

char pFirmwareTime[24];

char pDevModel[32];

jfviot\_en\_power\_type iPowerType;

jfviot\_en\_fs\_type iFsType; //demo里面没有了

char resv[64];

}jfviot\_st\_init\_params;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| uuid | Device networking serial number |
| auth\_key | Device networking password |
| pid | Device Type ID |
| pMac | Device Mac addr |
| iCaptrueChannels | Number of local video channels |
| iNetChannels | Number of network video channels |
| pFirmwareVersion | Device firmware version, customized by customers |
| pFirmwareTime | Firmware compilation time. The format is "2022-05-11 19:23:03". |
| iPowerType | Equipment power supply type |
| iFsType | Filesystem type can be JFVIOT\_EN\_FS\_FAT32 or JFVIOT\_EN\_FS\_PRIVATE\_FS. |
| resv |  |

【 **NOTE】 //demo没有**

If iFsType is set to a JFVIOT\_EN\_FS\_PRIVATE\_FS, all storage and playback logic will be completed internally by the SDK.

pFirmwareVersion ：Only for App display

pDevModel ： Only for App display

pFirmwareTime ： Not only for app display, but also for cloud upgrade function

【**Related data types and interfaces 】**

jfviot\_init\_params()

#### jfviot\_st\_callback\_funclist

【 **illustrate** 】

Callback function pointer list

【 **definition** 】

typedef struct

{

jfviot\_get\_dev\_info\_callback cb\_get\_dev\_info;

jfviot\_dev\_manager\_callback cb\_dev\_operate;

jfviot\_disk\_opr\_callback cb\_disk\_operate;

jfviot\_audio\_play\_cmd\_callback cb\_audio\_play;

jfviot\_ptz\_control\_callback cb\_ptz\_ctrl;

jfviot\_upgrade\_callback cb\_upgrade;

jfviot\_getconfig\_callback cb\_get\_config;

jfviot\_setconfig\_callback cb\_set\_config;

} jfviot\_st\_callback\_funclist;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **parameter name** | **describe** |
| cb\_get\_dev\_info | Obtain the device running status callback |
| cb\_dev\_operate | Device operation callback |
| cb\_disk\_operate | Disk operation callback |
| cb\_audio\_play | Voice intercom callback |
| cb\_ptz\_ctrl | PTZ control callback |
| cb\_upgrade | Firmware upgrade callback |
| cb\_get\_config | Get config callback |
| cb\_set\_config | Set config callback |

【 **NOTE】**

none

【**Related data types and interfaces】**

jfviot\_init\_params()

#### jfviot\_en\_power\_type

【 **illustrate** 】

Callback function pointer list

【 **definition** 】

typedef int jfviot\_en\_power\_type;

【 **PARAMETER 】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_NORMAL\_POWER\_DEV | 0 | Normal electricity |
| JFVIOT\_EN\_NORMAL\_NORMAL\_DEV | 1 | Low power consumption |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

## jfviot\_bind.h

This file mainly contains the interface of WIFI distribution network.

### jfviot\_get\_wifi\_info\_from\_link\_qrcode

【 **FUNCTION 】**

Parse app QR code data and register it with XM Cloud server (wireless distribution network).

**【 prototype 】**

JFVIOT\_RET jfviot\_get\_wifi\_info\_from\_link\_qrcode(const char\* pdata, char ssid[65], char pwd[65]);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| pdata | decoded data | input |
| ssid | wifi ssid | output |
| pwd | wifi password | output |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE 】**

none.

**【 EXAMPLE 】**

See demo .

## jfviot\_config.h​​

This file mainly contains some configuration structures.

### Data Type

#### jfviot\_en\_config\_type

【 **illustrate** 】

config type

【 **definition** 】

typedef int jfviot\_en\_config\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_CONFIG\_NETCOMMON | 0 | Common network configuration |
| JFVIOT\_EN\_CONFIG\_VIDEOENCODE | 1 | Coding configuration |
| JFVIOT\_EN\_CONFIG\_TIMEZONE | 2 | Device time zone |
| JFVIOT\_EN\_CONFIG\_NETDNS | 3 | Device DNS |
| JFVIOT\_EN\_CONFIG\_VIDEOOSD | 4 | Video OSD overlay, including single-line channel title and time title |
| JFVIOT\_EN\_CONFIG\_VIDEOEXTOSD | 5 | Extended video multi-line OSD overlay |
| JFVIOT\_EN\_CONFIG\_CAMEAR\_PARAM | //demo 里没有 | Camera Param |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

#### jfviot\_st\_netcommon

【 **illustrate** 】

common network configuration

【 **definition** 】

typedef struct

{

char pNetIP[16]; //Wired IP, format as 192.168.1.10

char pGateWay[16]; //Wired gateway, format as 192.168.1.1

char pNetMask[16]; //Wired subnet mask, format as 255.255.255.0

}jfviot\_st\_netcommon;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| pNetIP | Wired IP, format as 192.168.1.10 |
| pGateWay | Wired gateway, format as 192.168.1.1 |
| pNetMask | Wired subnet mask, format as 255.255.255.0 |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_videoencode

【 **illustrate** 】

video encode

【 **definition** 】

typedef struct

{

unsigned int mainVideoResolution;

unsigned int mainVideoCompression;

unsigned int mainVideoFPS;

unsigned int mainAudioEnable;

unsigned int extVideoResolution;

unsigned int extVideoCompression;

unsigned int extVideoFPS;

unsigned int extAudioEnable;

}jfviot\_st\_videoencode;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| mainVideoResolution | Main stream resolution |
| mainVideoCompression | Main code stream encoding format |
| mainVideoFPS | Frame rate of main code stream (how many I frames and P frames per second) |
| mainAudioEnable | Main code stream audio acquisition enable |
| extVideoResolution | Extra stream resolution |
| extVideoCompression | Extra code stream encoding format |
| extVideoFPS | Frame rate of extra code stream (how many I frames and P frames per second) |
| extAudioEnable | Auxiliary code stream audio acquisition enable |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_timezone

【 **illustrate** 】

time zone

【 **definition** 】

typedef struct

{

int timeMin;

}jfviot\_st\_timezone;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| mainVideoResolution | The difference between UTC time and UTC time (based on the value of the global variable timezone in Linux, unit: minutes, negative number means East X zone, positive number means West X zone) |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_netdns

【 **illustrate** 】

jfviot\_st\_netdns

【 **definition** 】

typedef struct

{

char pPrimeDNS[16]; //Primary DNS

char pSpareDNS[16]; //Alternate DNS

}jfviot\_st\_netdns;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| pPrimeDNS | Primary DNS |
| pSpareDNS | Alternate DNS |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_video\_osd

【 **illustrate** 】

video osd(on-screen display)

【 **definition** 】

typedef struct

{

unsigned int isShowMainTitle;

char pMainTitle[64];

unsigned short mainTitlePosX; //

unsigned short mainTitlePosY; //

unsigned int isShowTimeTitle; //

unsigned short timeTitlePosX; //

unsigned short timeTitlePosY; //

}jfviot\_st\_video\_osd;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| isShowMainTitle | Show main title |
| pMainTitle | Main channel title (single line, UTF8 encoding) |
| mainTitlePosX | Relative position of the X-axis at the top left corner of the main title (converted to 0~10000 by comparison) |
| mainTitlePosY | Relative position of the Y-axis at the top left corner of the main title (converted to 0~10000 by comparison) |
| isShowTimeTitle | Show time title |
| timeTitlePosX | Relative position of the X-axis at the top left corner of the time title (converted to 0~10000 by comparison) |
| timeTitlePosY | Relative position of the Y-axis at the top left corner of the time title (converted to 0~10000 by comparison) |
|  |  |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_video\_extosd

【 **illustrate** 】

extra video osd (on-screen display)

【 **definition** 】

typedef struct

{

unsigned int isShowExtTitle;

char pExtTitle[6][64];

unsigned short extTitlePosX; //

unsigned short extTitlePosY; //

}jfviot\_st\_video\_extosd;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| isShowExtTitle | Show extended headers |
| pExtTitle | Extended header (multiple lines, UTF8 encoding) |
| extTitlePosX | Relative position of the X-axis at the top left corner of the expanded title (converted to 0~10000 by comparison) |
| extTitlePosY | Relative position of the Y-axis at the top left corner of the expanded title (converted to 0~10000 by comparison) |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_camera\_param //demo里面没有

【 **illustrate** 】

Camera Param

【 **definition** 】

typedef struct

{

char dayNightMode;

char rev[31];

}jfviot\_st\_camera\_param;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| dayNightMode | Day Night Mode 0 :Infrared Night Version 1:Color Night Version 2:Intelligent Night Version |
| rev | reserve |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

## jfviot\_dev.h​​

This file mainly contains device-related enumerations and structures.

### Data Type

#### jfviot\_en\_get\_dev\_info

【 **illustrate** 】

Obtaining Device Information

【 **definition** 】

typedef int jfviot\_en\_get\_dev\_info;

【 **PARAMETER】**

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_DEVINFO\_STORAGEINFO | 0 | Store information |
| JFVIOT\_EN\_DEVINFO\_WORKSTATE | 1 | Alarm and video status |
| JFVIOT\_EN\_DEVINFO\_WIFIAP | 2 | The searched wifi hotspot information, corresponding to the structure jfviot\_st\_wifi\_APS |
| JFVIOT\_EN\_DEVINFO\_NETCONNECT\_INFO | 3 | Device network connection information, corresponding to the structure jfviot\_st\_net\_connect\_info |
| JFVIOT\_EN\_DEVINFO\_ENCODEABILITY | 4 | Equipment coding capability, corresponding to jfviot\_st\_encode\_ability |
|  |  |  |
|  |  |  |

#### jfviot\_st\_wifi\_APs

【 **illustrate** 】

Obtain the searched wifi hotspot information

【 **definition** 】

typedef struct

{

unsigned int apNumbers; //

char pStrSSID[32][64]; //

int nRSSI[32]; //

}jfviot\_st\_wifi\_APs;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| pStrSSID | Hotspot Name |
| nRSSI | Strength of signal |
|  |  |
|  |  |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

#### jfviot\_st\_net\_connect\_info

【 **illustrate** 】

Obtain network connection status information

【 **definition** 】

typedef struct

{

int Eth0Status;

int WlanStatus;

int WifiSignalLevel;

char pWlanMac[24];

}jfviot\_st\_net\_connect\_info;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| Eth0Status | Wire network connection: 1: the network cable is connected. 0: The network cable is not connected |
| WlanStatus | Wireless network connection: 1: connected 0: not connected |
| WifiSignalLevel | The wifi signal strength of the currently connected router |
| pWlanMac | Wireless NIC mac address in xx:xx:xx:xx:xx format |
|  |  |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

#### jfviot\_en\_dev\_operate

【 **illustrate** 】

Device System Management

【 **definition** 】

typedef int jfviot\_en\_dev\_operate;

【 **PARAMETER】**

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_DEV\_NONE | 0 | Not define |
| JFVIOT\_EN\_DEV\_REBOOT | 1 | Device restart |
| JFVIOT\_EN\_DEV\_SHUTDOWN | 2 | Device shutdown |
| JFVIOT\_EN\_DEV\_REVERTDEFAULT | 3 | Restoring Default Settings |
| JFVIOT\_EN\_DEV\_TOROUTE | 4 | Switch to routing mode |
| JFVIOT\_EN\_DEV\_TOAP | 5 | Switch to wireless mode |
| JFVIOT\_EN\_DEV\_TOCONFIG | 6 | Switch to configuration mode |
| JFVIOT\_EN\_DEV\_HORIZONTALTOUR | 7 | Horizontal cruising |
| JFVIOT\_EN\_DEV\_WHITELIGHTCTR | 8 | White light control |
| JFVIOT\_EN\_DEV\_UTCTIME | 9 | UTC time |

#### jfviot\_st\_utc\_time

【 **illustrate** 】

UTC time.

【 **definition** 】

typedef struct

{

int year;

int month;

int day;

int wday;

int hour;

int minute;

int second;

int isdst;

}jfviot\_st\_utc\_time;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| year | year |
| month | mouth, January = 1, February = 2, and so on |
| day | day |
| wday | week, Sunday = 0, Monday = 1, and so on |
| hour | hour |
| minute | minute |
| second | second |
| isdst | Daylight Saving Time Identification |
|  |  |
|  |  |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

## jfviot\_disk.h​​

This file mainly contains enumerations and structures related to hard disk and file system operations.

### Data Type

#### jfviot\_en\_disk\_opr\_type

【 **illustrate** 】

Disk Management

【 **definition** 】

typedef int jfviot\_en\_disk\_opr\_type ;

【 **PARAMETER 】**

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_DISK\_SETTYPE | 0 | Setting Type |
| JFVIOT\_EN\_DISK\_RECOVER | 1 | Error Recovery |
| JFVIOT\_EN\_DISK\_PARTITIONS | 2 | Partition operation |
| JFVIOT\_EN\_DISK\_CLEAR | 3 | Disk Formatting |
|  |  |  |

#### jfviot\_en\_fs\_type //demo没有

【 **illustrate** 】

Disk Management

【 **definition** 】

typedef int jfviot\_en\_fs\_type;

【 **PARAMETER 】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| JFVIOT\_EN\_FS\_FAT32 | FAT32 file system |
| JFVIOT\_EN\_FS\_PRIVATE\_FS | Private file system with higher storage efficiency and higher security |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

## jfviot\_error\_code .h​

This file mainly defines various error codes.

### Data Type

#### JFVIOT\_RET

【 **illustrate** 】

Return value.

【 **definition** 】

typedef int JFVIOT\_RET;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **Value** | **describe** |
| JFVIOT\_RET\_OK | 0 | success |
| JFVIOT\_RET\_UNKNOWN\_FAILED | -1 | unknown |
| JFVIOT\_RET\_INVALID\_UUID | -2 | uuid invalid |
| JFVIOT\_RET\_INVALID\_AUTHKEY | -3 | authkey invalid |
| JFVIOT\_RET\_INVALID\_MAC | -4 | mac invalid |
| JFVIOT\_RET\_VERSION\_MISMATCH | -5 | SDK header file and SDK library do not match |
| JFVIOT\_RET\_FORMAT\_PARSE\_ERROR | -6 | data in wrong format |
| JFVIOT\_RET\_CLOUD\_THRD\_FAILED | -7 | cloud service error |
| JFVIOT\_RET\_START\_IP\_SEARCH\_FAILED | -8 | start ip search failed |
| JFVIOT\_RET\_PUSH\_FRAME\_TYPE\_ERROR | -9 | frame type does not exist |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

## jfviot\_event.h​​

This file mainly defines the interfaces and structures related to alarm event notification.

### jfviot\_push\_alarm\_info

【 **FUNCTION 】**

Alarm signal transmission

**【 prototype 】**

JFVIOT\_RET jfviot\_push\_alarm\_info(jfviot\_st\_alarm\_params\* pAlarm\_params, jfviot\_st\_alarm\_pic\_info\* pPic\_info); 【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| pAlarm\_params | Alarm information parameters | input |
| pPic\_info | Alarm picture information. If not necessary, you can fill in NULL | input |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### Data Type

#### jfviot\_en\_alarm\_type

【 **illustrate** 】

Type of alarm

【 **definition** 】

typedef int jfviot\_en\_alarm\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_LOCALIOALARM | 0 | Local IO alarm |
| JFVIOT\_EN\_MOTIONDETECT | 1 | Detection of motion |
| JFVIOT\_EN\_LOSSDETECT | 2 | Video lost |
| JFVIOT\_EN\_BLINDDETECT | 3 | Lens blocking |
| JFVIOT\_EN\_HUMANDETECT | 4 | Human detection |
| JFVIOT\_EN\_VEHICLE | 5 | Vehicle detected |
| JFVIOT\_EN\_PET | 6 | Pet detected |
| JFVIOT\_EN\_FALLDOWN | 7 | Human fall down |
| JFVIOT\_EN\_FIRE | 8 | Fire detected |
| JFVIOT\_EN\_PACKAGE | 9 | Package detected |
| JFVIOT\_EN\_HANDLED\_OBJ | 10 | Handled objecet detected |
| JFVIOT\_EN\_NON\_MOTOR | 11 | Non vehicle detected |
| JFVIOT\_EN\_PLATE\_REC | 12 | Car plate recognize |
| JFVIOT\_EN\_FACE\_DETECT | 13 | Face detected |
| JFVIOT\_EN\_CRY\_DETECT | 14 | Cry Sound Detection |
| JFVIOT\_EN\_SLEEP\_AT\_WORK | 15 | Sleep At Work |
| JFVIOT\_EN\_SLEEPER | 16 | Long lying down |
| JFVIOT\_EN\_OUT\_DOOR\_LOCK | 17 | Outside door locked |
| JFVIOT\_EN\_IN\_DOOR\_LOCK | 18 | Inside door locked |
| JFVIOT\_EN\_MANUAL\_LOCK | 19 | Manual lock |
| JFVIOT\_EN\_AUTO\_LOCK | 20 | Automatic lock |
| JFVIOT\_EN\_OUT\_DOOR\_BUTTON\_LOCK | 21 | Outside door button lock |
| JFVIOT\_EN\_OUT\_DOOR\_UNLOCK | 22 | Outside door unlocked |
| JFVIOT\_EN\_IN\_DOOR\_UNLOCK | 23 | Inside door unlocked |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_en\_action\_type

【 **illustrate** 】

Alarm status

【 **definition** 】

typedef int jfviot\_en\_action\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_EVENT\_START | 0 | Start of event |
| JFVIOT\_EN\_EVENT\_STOP | 1 | STOP of Event |
|  |  |  |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_alarm\_pic\_info

【 **illustrate** 】

Alarm picture information

【 **definition** 】

typedef struct

{

unsigned char\* pPic;

unsigned int iPicLen;

}jfviot\_st\_alarm\_pic\_info;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| pPic | Alarm picture data |
| iPicLen | Alarm picture length |
|  |  |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_alarm\_params

【 **illustrate** 】

Alarm params

【 **definition** 】

typedef struct

{

unsigned int iChannel;

jfviot\_en\_action\_type iAction;

jfviot\_en\_alarm\_type icmd;

}jfviot\_st\_alarm\_params;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| iChannel | Channel number |
| iAction | Alarm status |
| Icmd | Type of alarm |

【 **NOTE】**

none

【**Related data types and interfaces 】**

## jfviot\_media.h

This file mainly defines the interfaces and structures related to real-time audio, video and intercom modules.

### jfviot\_request\_I\_frame\_cb

【 **FUNCTION 】**

call back function to request one I frame from video decoder. set to NULL if not needed or for NON-video channel.

**【 prototype 】**

typedef void (\* jfviot\_request\_I\_frame\_ cb )( int channel, jfviot\_en\_stream\_type streamType );

【 **PARAMETER 】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| channel | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| streamType | Type of stream | input |
|  |  |  |

【 **RETURN** 】

none.

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_stream\_buffer\_init

【 **FUNCTION】**

Initializes the buffer size of the stream data.

**【 prototype 】**

JFVIOT\_RET jfviot\_stream\_buffer\_init(int iChannel, jfviot\_en\_stream\_type StreamType, jfviot\_st\_buffer\_init\_params\* pParam);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| iChannel | Channel number | input |
| StreamType | Type of stream | input |
| pParam | Initializes the stream buffer params | input |
|  |  | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_push\_video\_data

【 **FUNCTION】**

stream push function.

**【 prototype 】**

JFVIOT\_RET jfviot\_push\_video\_data(jfviot\_st\_stream\_params\* pStreamParam);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| pStreamParam | stream information parameter | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

**【 EXAMPLE】**

See demo.

### Data Type

#### jfviot\_en\_video\_compression

【 **illustrate** 】

Video compression format.

【 **definition** 】

typedef int jfviot\_en\_video\_compression;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_COMPRESSION\_H264 | 0 | H264 |
| JFVIOT\_EN\_COMPRESSION\_H265 | 1 | H265 |
| JFVIOT\_EN\_COMPRESSION\_NR | 2 | Num of compression type |
|  |  |  |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_en\_video\_resolution

【 **illustrate** 】

Resolution of video.

【 **definition** 】

typedef int jfviot\_en\_video\_resolution;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_RESOLUTION\_CIF | 0 | 352\*288 |
| JFVIOT\_EN\_RESOLUTION\_QVGA | 1 | 640\*360 |
| JFVIOT\_EN\_RESOLUTION\_D1 | 2 | 720\*576(PAL) 720\*480(NTSC) |
| JFVIOT\_EN\_RESOLUTION\_VGA | 3 | 640\*480(PAL) 640\*480(NTSC) |
| JFVIOT\_EN\_RESOLUTION\_960H | 4 | 928\*576 |
| JFVIOT\_EN\_RESOLUTION\_720P | 5 | 1280\*720 |
| JFVIOT\_EN\_RESOLUTION\_960P | 6 | 1280\*960 |
| JFVIOT\_EN\_RESOLUTION\_1080P | 7 | 1920\*1080 |
| JFVIOT\_EN\_RESOLUTION\_3M | 8 | 2048\*1536 |
| JFVIOT\_EN\_RESOLUTION\_4M | 9 | 2592\*1520 |
| JFVIOT\_EN\_RESOLUTION\_5M | 10 | 3744\*1408 |
| JFVIOT\_EN\_RESOLUTION\_6M | 11 | 3072\*2048 |
| JFVIOT\_EN\_RESOLUTION\_8M | 12 | 3264\*2448 |
| JFVIOT\_EN\_RESOLUTION\_4K | 13 | 4096\*2160 /3840\*2160 |
| JFVIOT\_EN\_RESOLUTION\_12M | 14 | 4000\*3000 |
| JFVIOT\_EN\_RESOLUTION\_NR | 15 | Num of resolution |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_en\_stream\_type

【 **illustrate** 】

Stream Type.

【 **definition** 】

typedef int jfviot\_en\_stream\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_VIDEO\_STREAM\_MAIN | 0 | Main stream |
| JFVIOT\_EN\_VIDEO\_STREAM\_EXT | 1 | Sub stream |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_time

【 **illustrate** 】

Time.

【 **definition** 】

typedef struct

{

unsigned int iYear;

unsigned int iMonth;

unsigned int iDay;

unsigned int iHour;

unsigned int iMinute;

unsigned int iSecond;

}jfviot\_st\_time;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| iYear | year |
| iMonth | month |
| iDay | day |
| iHour | hour |
| iMinute | minute |
| iSecond | second |

【 **NOTE】**

none

【**Related data types and interfaces】**

#### jfviot\_en\_video\_encode\_type

【 **illustrate** 】

Video encoding type

【 **definition** 】

typedef int jfviot\_en\_video\_encode\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_VIDEO\_ENCODE\_H264 | 0 | H264 |
| JFVIOT\_EN\_VIDEO\_ENCODE\_H265 | 1 | H265 |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_video\_stream\_params

【 **illustrate** 】

Video stream parameters.

【 **definition** 】

typedef struct

{

jfviot\_en\_video\_encode\_type eEncodeType; //

unsigned int iWidth; //

unsigned int iHeight; //

jfviot\_st\_time stDate; //

unsigned short iMilli; //

unsigned char cFrameRate; //

unsigned char resv;

}jfviot\_st\_video\_stream\_params;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| eEncodeType | Video encoding type, valid for I-frames only |
| iWidth | Wide resolution, valid for I-frames only |
| iHeight | High resolution, valid for I-frames only |
| stDate | Date format: 2022-04-29 10:00:00 This parameter is valid only for I-frames |
| iMilli | Milliseconds I-frame: indicates the current time in milliseconds. P-frame: indicates the time offset from the previous I-frame |
| cFrameRate | Frame rate, valid only for I-frames |
| resv | reserved |

【 **NOTE】**

none

【**Related data types and interfaces】**

#### jfviot\_en\_audio\_type

【 **illustrate** 】

Audio encoding type.

【 **definition** 】

typedef int jfviot\_en\_audio\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_AUDIO\_G711A | 0 | G711A |
| JFVIOT\_EN\_AUDIO\_G711U | 1 | G711U |
| JFVIOT\_EN\_AUDIO\_GAAC | 2 | GAAC |
| JFVIOT\_EN\_AUDIO\_GAMR | 3 | GAMR |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_audio\_stream\_params

【 **illustrate** 】

Audio stream parameters.

【 **definition** 】

typedef struct

{

jfviot\_en\_audio\_type eEncodeType; //

unsigned int iSamplingRate; //

unsigned short iMilli; //

unsigned short resv; //

}jfviot\_st\_audio\_stream\_params;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| eEncodeType | Audio encoding type |
| iSamplingRate | Audio sampling rate 8000/16000/44100/48000 |
| iMilli | The number of milliseconds, and the time offset of the previous I-frame |
| resv | reserved |

【 **NOTE】**

none

【**Related data types and interfaces】**

#### jfviot\_en\_frame\_type

【 **illustrate** 】

Type of frame

【 **definition** 】

typedef int jfviot\_en\_frame\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_FRAME\_I | 0 | I frame |
| JFVIOT\_EN\_FRAME\_P | 1 | P frame |
| JFVIOT\_EN\_FRAME\_AUDIO | 2 | Audio frame |
|  |  |  |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_stream\_params

【 **illustrate** 】

stream parameters.

【 **definition** 】

typedef struct

{

unsigned int iChannel; //

jfviot\_en\_stream\_type cStreamType; //Type of stream

jfviot\_en\_frame\_type eFrameType; //Type of frame

union

{

jfviot\_st\_video\_stream\_params stVideoParams; //Video frame parameters

jfviot\_st\_audio\_stream\_params stAudioParams; //Audio frame parameters

};

unsigned int iLength; //

unsigned char\* pStreamData; //stream data

}jfviot\_st\_stream\_params;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| iChannel | Channel number |
| cStreamType | Type of stream |
| eFrameType | Type of frame |
| stVideoParams | Video frame parameters |
| stAudioParams | Audio frame parameters |
| iLength | stream length |
| pStreamData | stream data |
|  |  |

【 **NOTE】**

none

【**Related data types and interfaces】**

#### jfviot\_st\_buffer\_init\_params

【 **illustrate** 】

Initializes the stream buffer params.

【 **definition** 】

typedef struct

{

unsigned int bitrate;

unsigned int fps;

unsigned int max\_buffer\_seconds;

jfviot\_request\_I\_frame\_cb requestKeyFrameCB; //force I frame callback

} jfviot\_st\_buffer\_init\_params;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| bitrate | bitrate |
| fps | frame per second |
| max\_buffer\_seconds | max buffer seconds |
| requestKeyFrameCB | force I frame callback |

【 **NOTE】**

none

【**Related data types and interfaces】**

#### jfviot\_st\_video\_encode\_ability

【 **illustrate** 】

Initializes the stream buffer params.

【 **definition** 】

typedef struct

{

unsigned int mainResolutionMask; //

unsigned int mainMaxFPS[32]; //

unsigned int mainCompressionMask; //

unsigned int extResolutionMask; //

unsigned int extMaxFPS[32]; //

unsigned int extCompressionMask; //

}jfviot\_st\_video\_encode\_ability;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| mainResolutionMask | Resolution types supported by the main stream, each bit represents a resolution, such as the enumeration jfviot\_en\_video\_resolution |
| mainMaxFPS | The maximum frame rate supported by the main stream at various resolutions, corresponding to the position of the resolution mask |
| mainCompressionMask | Type of compression format supported by the main code stream. Each bit indicates a compression format, such as enumerated jfviot\_en\_video\_compression |
| extResolutionMask | Resolution types supported by the auxiliary stream, each bit represents a resolution, such as the enumeration jfviot\_en\_video\_resolution |
| extMaxFPS | The maximum frame rate supported by the auxiliary stream at various resolutions corresponds to the position of the resolution mask |
| extCompressionMask | Type of compression format supported by the auxiliary code stream. Each bit indicates a compression format, such as the enumerated jfviot\_en\_video\_compression |

【 **NOTE】**

none

【**Related data types and interfaces】**

#### jfviot\_en\_talk\_opr\_type

【 **illustrate** 】

Type of talk operation

【 **definition** 】

typedef int jfviot\_en\_talk\_opr\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_TALK\_START | 0 | Talk start |
| JFVIOT\_EN\_TALK\_STOP | 1 | Talk stop |
| JFVIOT\_EN\_TALK\_DATA | 2 | Talk data |
|  |  |  |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

#### jfviot\_st\_video\_dec\_pram

【 **illustrate** 】

Parameters for video and audio decoding

【 **definition** 】

typedef struct

{

jfviot\_en\_video\_encode\_type eVideoEncodeType; //Video encoding type

jfviot\_en\_audio\_type eAudioEncodeType; //Audio encoding type

unsigned int iWidth; //Wide resolution

unsigned int iHeight; //High resolution

unsigned int iFPS; //video frame rate

unsigned int iSampleBits;

unsigned int iSampleRates;

}jfviot\_st\_video\_dec\_pram;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| eVideoEncodeType | Video encoding type. This indicates the codec or format used for encoding the video data. |
| eAudioEncodeType | Audio encoding type. This indicates the codec or format used for encoding the audio data. |
| iWidth | Wide resolution. This specifies the width of the video frame in pixels. |
| iHeight | High resolution. This specifies the height of the video frame in pixels. |
| iFPS | Video frame rate. This indicates the number of frames per second in the video. |
| iSampleBits | Sample bits. This specifies the number of bits per sample in the audio data. |
| iSampleRates | Sample rates. This indicates the sampling rate of the audio data, typically in Hertz (Hz). |

#### jfviot\_st\_video\_data

【 **illustrate** 】

Data of video or audio streams

【 **definition** 】

typedef struct

{

jfviot\_en\_frame\_type eFrameType;

union

{

jfviot\_st\_video\_stream\_params stVideo;

jfviot\_st\_audio\_stream\_params stAudio;

};

unsigned int iLength; //stream length

unsigned char\* pStreamData; //stream data

}jfviot\_st\_video\_data;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| eFrameType | Frame type. This indicates whether the data in the stream is a video frame, an audio frame, or some other type of frame. |
| stVideo | Video stream parameters. This contains detailed information about the video stream, such as codec-specific settings. |
| stAudio | Audio stream parameters. This contains detailed information about the audio stream, such as codec-specific settings. |
| iLength | Stream length. This specifies the length of the stream data in bytes. |
| pStreamData | Stream data. This is a pointer to the actual data of the stream, which could be video or audio data depending on the context. |

## jfviot\_ptz.h

This file mainly defines the interfaces and structures related to the gimbal.

### Data Type

#### jfviot\_en\_ptz\_operate

【 **illustrate** 】

Type of PTZ operation.

【 **definition** 】

typedef int jfviot\_en\_ptz\_operate ;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_PTZ\_NONE | 0 | none |
| JFVIOT\_EN\_PTZ\_LEFTUP | 1 | Control of direction left up |
| JFVIOT\_EN\_PTZ\_UP | 2 | Control of direction up |
| JFVIOT\_EN\_PTZ\_RIGHTUP | 3 | Control of direction right up |
| JFVIOT\_EN\_PTZ\_LEFT | 4 | Control of direction left |
| JFVIOT\_EN\_PTZ\_RIGHT | 5 | Control of direction right |
| JFVIOT\_EN\_PTZ\_LEFTDOWN | 6 | Control of direction left down |
| JFVIOT\_EN\_PTZ\_DOWN | 7 | Control of direction down |
| JFVIOT\_EN\_PTZ\_RIGHTDOWN | 8 | Control of direction right down |
| JFVIOT\_EN\_PTZ\_ZOOM\_WIDE | 9 | zoom wide |
| JFVIOT\_EN\_PTZ\_ZOOM\_TELE 10 | 10 | zoom tele |
| JFVIOT\_EN\_PTZ\_FOCUS\_FAR | 11 | focus far |
| JFVIOT\_EN\_PTZ\_FOCUS\_NEAR | 12 | focus near |
| JFVIOT\_EN\_PTZ\_IRIS\_LARGE | 13 | IRIS large |
| JFVIOT\_EN\_PTZ\_IRIS\_SMALL | 14 | IRIS small |
| JFVIOT\_EN\_PTZ\_SETPRESET | 15 | Set the current location to the preset point |
| JFVIOT\_EN\_PTZ\_CLEARPRESET | 16 | Deletes a specified preset point |
| JFVIOT\_EN\_PTZ\_GOTOPRESET | 17 | Go to the specified preset point |
| JFVIOT\_EN\_PTZ\_ADDTOUR | 18 | Add preset points to the cruise line |
| JFVIOT\_EN\_PTZ\_DELTOUR | 19 | Delete preset points from the cruise line, or delete the entire cruise line |
| JFVIOT\_EN\_PTZ\_STARTTOUR | 20 | Start a point to point cruise on a line |
| JFVIOT\_EN\_PTZ\_STOPTOUR | 21 | End a point to point cruise on a line |
| JFVIOT\_EN\_PTZ\_GOTOPOSITION | 22 | Setting the lens orientation |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_st\_ptz\_direct\_ctrl

【 **illustrate** 】

Control of direction.

【 **definition** 】

typedef struct

{

int iVertSpeed;

int iHorSpeed; //

int iCtrlStatu; //

}jfviot\_st\_ptz\_direct\_ctrl;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| iVertSpeed | Vertical velocity |
| iHorSpeed | Horizontal velocity |
| iCtrlStatu | Control states -1: Stop, Others: start |
|  |  |

【 **NOTE】**

none

【**Related data types and interfaces】**

#### jfviot\_st\_ptz\_len\_ctrl

【 **illustrate** 】

Lens control (zoom, focus, IRIS).

【 **definition** 】

typedef struct

{

int iStep; //

int iCtrlStatu; //

}jfviot\_st\_ptz\_len\_ctrl;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| iStep | Step length |
| iCtrlStatu | Control states -1: Stop, Others: start |

【 **NOTE】**

none

【**Related data types and interfaces】**

#### jfviot\_st\_ptz\_tour\_manager

【 **illustrate** 】

Point of cruising

【 **definition** 】

typedef struct

{

int iTour; //

int iPreset; //

int iStayTime; //

}jfviot\_st\_ptz\_tour\_manager;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| iTour | Cruise line number |
| iPreset | Preset point number (When the cruise line is cleared, if the value is -1, the entire cruise line is deleted; if the value is greater than or equal to 0, only the specified preset point is deleted from the line) |
| iStayTime | Residence time, in seconds |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

## jfviot\_record.h //demo里面没有

This file mainly defines the interfaces and structures related to the video playback module.

### jfviot\_get\_file\_list

【 **FUNCTION 】**

search record file call back .

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_get\_file\_list)(uint chan, uchar stream\_type,jfviot\_st\_time\* start\_time, jfviot\_st\_time\* end\_time, uint type,jfviot\_file\_info \*file,uint \*num);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| chan | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| streamType | Type of stream | input |
| start\_time | the record time start | input |
| end\_time | the record time end | input |
| type | the record type | input |
| file | filelist | output |
| num | In:the max record num out:the result of record num | in/out |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_file\_search\_calendar

【 **FUNCTION】**

search which day within this month includes recordings.

**【 prototype 】**

JFVIOT\_RET jfviot\_file\_search\_calendar(uint chan, uint year, uint month, uint type, unsigned int\* mask);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| chan | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| year | search for which year | input |
| month | search for which month | input |
| type | the record type | input |
| mask | bit 0 -> On the 1st of this month  mask bit 1 -> On the 2st of this month  ...  the value of the bit is 1 represents that there was a video recording on that day | in/out |
|  |  |  |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_file\_playback\_start

【 **FUNCTION】**

start playback of record file.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_file\_playback\_start)(PLAY\_BACK\_HANDLE handle, jfviot\_file\_info \*file,int time); 【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| handle | play back connection handle | input |
| file | the record info | input |
| month | search for which month | input |
| time | Time offset within the file | input |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_file\_playback\_start\_by\_time

【 **FUNCTION】**

start playback of record file by time.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_file\_playback\_start\_by\_time)(PLAY\_BACK\_HANDLE handle, uint chan, jfviot\_st\_time\* start\_time, jfviot\_st\_time\* end\_time, int type, uchar stream\_type);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| handle | play back connection handle | input |
| chan | Channel number: -1 indicates that the channel is not differentiated, and a non-negative number indicates the channel number. This parameter is applicable to DVR and NVR devices | input |
| start\_time | the playback want to start | input |
| end\_time | the playback want to end | input |
| type | the record type | input |
| stream\_type | 0 mainstream 1 substream | input |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_file\_playback\_goto\_time

【 **FUNCTION】**

Change plauback time.

**【 prototype 】**

JFVIOT\_RET jfviot\_file\_playback\_goto\_time(PLAY\_BACK\_HANDLE handle,jfviot\_st\_time\* time);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| handle | play back connection handle | input |
| time | the record time to start | input |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_file\_playback\_stop

【 **FUNCTION】**

stop playback of record file calback.

**【 prototype 】**

typedef JFVIOT\_RET (\*jfviot\_file\_playback\_stop)(PLAY\_BACK\_HANDLE handle,jfviot\_file\_info \*file,int time);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| handle | play back connection handle | input |
|  |  |  |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_playback\_file\_send\_data

【 **FUNCTION】**

send playback record file data.

**【 prototype 】**

JFVIOT\_RET jfviot\_playback\_file\_send\_data(PLAY\_BACK\_HANDLE handle,jfviot\_st\_stream\_params\* pStreamParam);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| handle | play back connection handle | input |
| pStreamParam | the record data that needs to be sent | input |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### jfviot\_file\_playback\_end

【 **FUNCTION】**

send playback record file end.

**【 prototype 】**

JFVIOT\_RET jfviot\_file\_playback\_end(PLAY\_BACK\_HANDLE handle);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| handle | play back connection handle | input |
|  |  |  |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### Data Type

#### jfviot\_en\_record\_type

【 **illustrate** 】

Record type.

【 **definition** 】

typedef int jfviot\_en\_record\_type;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_RECORD\_TYPE\_TIMER | 1 | normal record |
| JFVIOT\_EN\_RECORD\_TYPE\_ALARM | 2 | alarm record |
|  |  |  |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_file\_info

【 **illustrate** 】

Record file info.

【 **definition** 】

typedef struct \_jfviot\_file\_info

{

char file\_name[64];

uint file\_length ; ///< file lenth in KB

int file\_type; ///record type

jfviot\_st\_time start\_time ; ///< start\_time

jfviot\_st\_time end\_time ;

}jfviot\_file\_info;

【 **PARAMETER】**

|  |  |
| --- | --- |
| **member name** | **describe** |
| file\_name | File name |
| file\_length | file lenth in KB |
| file\_type | jfviot\_en\_record\_type |
| start\_time | Record start time |
| end\_time | Record end time |

【 **NOTE 】**

none

【 **Related data types and interfaces 】**

## jfviot\_ upgrade .h

This file mainly defines upgrade-related interfaces and structures.

### jfviot\_push\_upgrade\_progress

【 **FUNCTION 】**

Send the upgrade progress information

**【 prototype 】**

JFVIOT\_RET jfviot\_push\_upgrade\_ progress ( int progress);

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **parameter name** | **describe** | **input Output** |
| progress | <0 : check error or upgrade error, define as jfviot\_en\_upgrade\_error  0 ~ 99 : upgrading  >= 100 : upgrade finish | input |
|  |  |  |
|  |  |  |

【 **RETURN** 】

|  |  |
| --- | --- |
| **RETURN** | **Description** |
| JFVIOT\_RET | JFVIOT\_RET |

【 **NOTE】**

none.

**【 EXAMPLE】**

See demo.

### Data Type

#### jfviot\_en\_upgrade\_action

【 **illustrate** 】

Upgrade Action.

【 **definition** 】

typedef int jfviot\_en\_upgrade\_action;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_UPGRADE\_ACTION\_PREPARE | 0 | Notify new firmware info |
| JFVIOT\_EN\_UPGRADE\_ACTION\_DOWNLOAD | 1 | Download new firmware |
| JFVIOT\_EN\_UPGRADE\_ACTION\_STARTWRITE | 2 | Start write new firmware to Flash |
| JFVIOT\_EN\_UPGRADE\_ACTION\_ABORT | 3 | Interrupt upgrade |
| JFTSDK\_EN\_UPGRADE\_ACTION\_ISAUTO | 4 | Is auto upgrade |
|  |  |  |
|  |  |  |

【 **NOTE】**

none

【**Related data types and interfaces 】**

#### jfviot\_en\_upgrade\_error

【 **illustrate** 】

Type of Upgrade Result.

【 **definition** 】

typedef int jfviot\_en\_upgrade\_error;

【 **PARAMETER】**

|  |  |  |
| --- | --- | --- |
| **member name** | **value** | **describe** |
| JFVIOT\_EN\_UPGRADE\_ERROR\_NO\_ENOUGH\_MEMORY | -1 | No memory to store new firmware |
| JFVIOT\_EN\_UPGRADE\_ERROR\_INVALID\_FORMAT | -2 | The format of the upgrade file is incorrect |
| JFVIOT\_EN\_UPGRADE\_ERROR\_UPGRADE\_PART\_FAIL | -3 | A partition failed to upgrade |
| JFVIOT\_EN\_UPGRADE\_ERROR\_INVALID\_HARDWARE | -4 | The hardware model does not match |
| JFVIOT\_EN\_UPGRADE\_ERROR\_INVALID\_VENDOR | -5 | Customer information does not match |
| JFVIOT\_EN\_UPGRADE\_ERROR\_INVALID\_COMPATIBLE | -6 | Not compatible between new firmware and this device |
| JFVIOT\_EN\_UPGRADE\_ERROR\_INVALID\_VERSION | -7 | Cannot be upgraded back to the old program |
| JFVIOT\_EN\_UPGRADE\_ERROR\_INVALID\_WIFI\_DRIVER | -8 | The wifi driver in the upgrade program does not match the wifi network card currently used by the device |
| JFVIOT\_EN\_UPGRADE\_ERROR\_NETWORK\_ERR | -9 | Network error |
| JFVIOT\_EN\_UPGRADE\_ERROR\_NO\_SUPPORT\_CURFLASH | -10 | The upgrade program does not support Flash used by the device |
| JFVIOT\_EN\_UPGRADE\_ERROR\_FIRMWARE\_CRACKED | -11 | The upgrade file has been modified |

【 **NOTE】**

none

【**Related data types and interfaces 】**