International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.627.2

(03/2022)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Broadband, triple-play and advanced multimedia services – Advanced multimedia services and applications

Requirements and protocols for home surveillance systems

Recommendation ITU-T H.627.2

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Telepresence, immersive environments, virtual and extended reality	H.420–H.439
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
VEHICULAR GATEWAYS AND INTELLIGENT TRANSPORTATION SYSTEMS (ITS)	
Architecture for vehicular gateways	H.550–H.559
Vehicular gateway interfaces	H.560–H.569
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619
Advanced multimedia services and applications	**H.620–H.629**
Content delivery and ubiquitous sensor network applications	H.640–H.649
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	
General aspects	H.700–H.719
IPTV terminal devices	H.720–H.729
IPTV middleware	H.730–H.739
IPTV application event handling	H.740–H.749
IPTV metadata	H.750–H.759
IPTV multimedia application frameworks	H.760–H.769
IPTV service discovery up to consumption	H.770–H.779
Digital Signage	H.780–H.789
E-HEALTH MULTIMEDIA SYSTEMS, SERVICES AND APPLICATIONS	
Personal health systems	H.810–H.819
Interoperability compliance testing of personal health systems (HRN, PAN, LAN, TAN and WAN)	H.820–H.859
Multimedia e-health data exchange services	H.860–H.869
Safe listening	H.870–H.879

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T H.627.2

Requirements and protocols for home surveillance systems

Summary

Recommendation ITU-T H.627.2 defines the requirements and protocols for Internet protocol (IP) based network access of varied types of equipment under home security surveillance scenarios including the architecture, protocol for transmission, access and service functions as well as other relevant requirements under the home surveillance considerations.

This Recommendation specifies a way to eliminate the difference in network communication and access management of heterogeneous devices, simplify the construction complexity of home security platforms, improve the security and reliability of data transmission between home security equipment and the home surveillance platform, and finally guarantee high-qualified development of home surveillance service.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T H.627.2	2022-03-16	16	11.1002/1000/14971

Keywords

Home surveillance, interface, protocols, requirements.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Recommendation ITU-T H.627.2

Requirements and protocols for home surveillance systems

1 Scope

This Recommendation specifies the requirements and protocols for home surveillance systems (HSS) and covers the following aspects:

– Requirements for home surveillance systems;

– Reference architecture for home surveillance systems;

– Protocol interface framework for home surveillance systems;

– Relevant protocols of home surveillance systems.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T H.626] Recommendation ITU-T H.626 (V2) (2019), *Architectural requirements for video surveillance system*.

[ITU-T H.626.4] Recommendation ITU-T H.626.4 (2018), *Architecture for a point-to-point visual surveillance system*.

[IETF RFC 2818] IETF RFC 2818 (2000), *HTTP Over TLS*.

[IETF RFC 7230] IETF RFC 7230 (2014), *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

None.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 central management unit: A functional unit located in the main part of a home surveillance system. It is used for centralized system management, service operation management, etc. The main functions include management, authorization, accounting, charging, location and presence.

3.2.2 cloud storage: A functional unit locates at the central part of a home surveillance system. It is used to retrieve and store media and as well as provide a streaming media service. The main functions include media storage, media serving, media indexing, and media downloading, etc.

3.2.3 customer unit: A functional unit located at the customer part of a home surveillance system. It is used to present multimedia information (such as audio, video, image, alarm signal, etc.) to the end user.

3.2.4 home surveillance system: A telecommunication service focusing on video (including audio and image) application technology, which is used to remotely capture multimedia information (such as audio, video, image, alarm signal, etc.) and present them to the end users in a user-friendly manner. Above service is expected to be realized via a managed broadband network with ensured quality, security and reliability for home surveillance applications.

3.2.5 media distribution unit: A functional unit located in the main part of a home surveillance system. It is used to transport media from the premises unit (PU) to the customer unit (CU). The main functions include media reception, media processing, media routing, media transmission, media forwarding, media replication, etc.

3.2.6 premises unit: A functional unit located at the remote part of a home surveillance system and is used to capture multimedia information (such as audio, video, image, alarm signal, etc.) from a surveilled object.

3.2.7 service platform: A series of functional units and subsystems located at the central part of a home surveillance system. It is used to integrate all capabilities and provides home surveillance services to customers. The main functions include service-control function, media switching, distribution, storage, and control and management.

3.2.8 service control unit: A functional unit located in the main part of a home surveillance system. It is used for access service control and signal call control between the premises unit (PU) and the customer unit (CU). The main functions include access registration, access identification, authentication, authorization, call control, location, presence, and target media serving function selection.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CCS Central Control Server

CMU Centre Management Unit

CS Cloud Storage

CU Customer Unit

HSS Home Surveillance System

MDU Media Distribution Unit

MS Media Server

PTZ Pan/Tilt/Zoom

PU Premises Unit

REST Representation State Transfer

RPC Remote Procedure Call

SCU Service Control Unit

SRT Secure Reliable Transport

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

5 Conventions

In this Recommendation:

– The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.

– The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement needs not be present to claim conformance.

– The keywords "can optionally" and "may" indicate an optional requirement which is permissible, without implying any sense of being recommended. These terms are not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with the specification.

6 Message syntax and semantics

This Recommendation outlines the urgent requirements for home surveillance systems. When regarding the overall architectural framework of home surveillance systems, it references the architecture based on [ITU-T H.626] and [ITU-T H.626.4] with a few extensions. This Recommendation gives a more detailed description of protocol interfaces based on the reference architecture as well as the relevant protocols for such systems, including device management, media stream transmission, recording cloud storage (CS) and their related functions.

7 Requirements for home surveillance systems (HSS)

This Recommendation is targeted at providing a series of comprehensive requirements and protocols for a home surveillance system that is implemented in public IP based networks. The home surveillance system (HSS) is required to provide some advanced services such as remote real-time surveillance, video remote storage, and video playback. In addition, it is recommended to provide other value-added services when new AI technologies are used to realize object detection or identification. All devices accessing the HSS are recommended to follow the protocols and interface specifications defined in the following sections.

8 Reference architecture for HSS

In this clause, a reference architecture for HSS will be presented in detail.

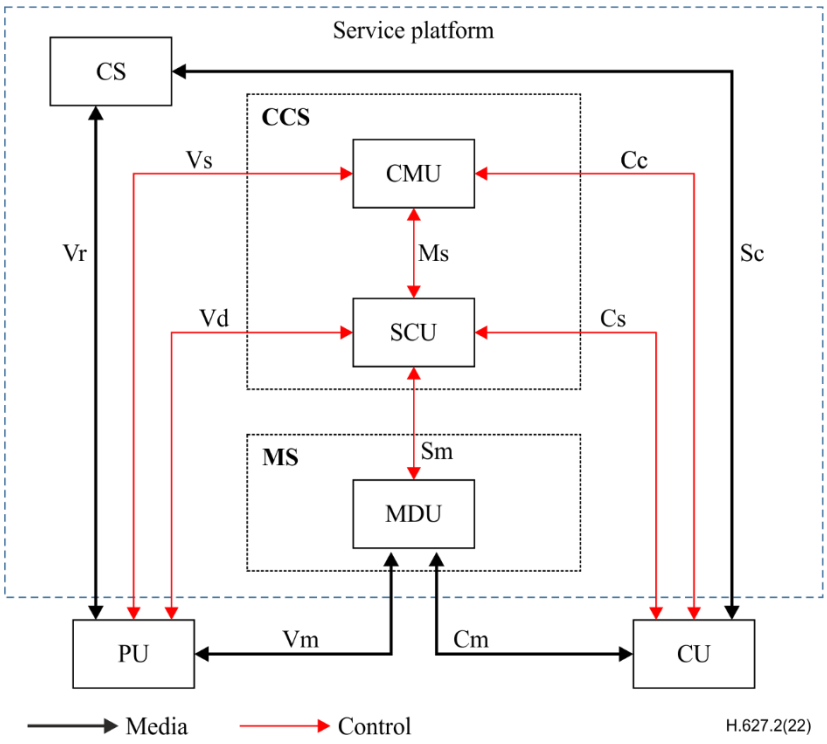

Figure 8-1 – Reference architecture of home surveillance system

8.1 Architectural framework

The architectural framework of home surveillance systems contains three major parts, including a premises unit (PU), customer unit (CU) and a service platform.

PU is a functional unit to capture multimedia information from a surveilled object. CU is a functional unit to present multimedia information towards the end user. There are four additional functional units in the service platform, namely, core storage (CS), service control unit (SCU), media distribution unit (MDU), and centre management unit (CMU). CS is used to retrieve and store media and as well as provides a streaming media service. SCU is used for access service control and signal call control between the PU and the CU. MDU is used to transport media from the PU to the CU. CMU is used for service and operational management functionality.

In order to provide the following functions via the service platform, all PUs are required to access the service platform with authentication and authorization according to the requirements and protocols for home surveillance systems in clause 9.

8.2 Functional entities

There are three main functional entities in the service platform, they are signalling control service, media broadcasting service and video service.

a) Signalling control service is responsible for control signalling between PUs and the service platform. PUs are required to access the service platform through interface Vs and interface Vd shown in Figure 8-1.

b) Media broadcasting service is responsible for push flow, transcoding, distribution and other functions of PUs. PUs are required to access the service platform through the interface Vm.

c) Video service is responsible for record slice uploading, storage, playback and other functions of PUs. PUs are required to access service platform through the interface Vr.

d) Scheduling service is responsible for sending instructions to PUs to connect to the designated signalling server and the peer-to-peer server.

8.3 Protocols and interfaces

8.3.1 Interface stack

This specification mainly defines the protocols of interfaces (Vs/Vd/Vm/Vr) between PUs and the service platform, including signalling and data transmission protocols. The entire protocol stack is presented in Figure 8-2 which consists of three functions, they are the transmission function, access function and service function.

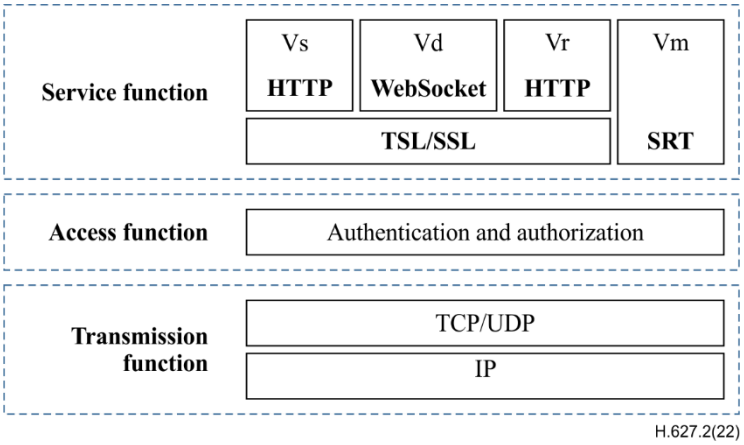

Figure 8-2 – Interface stack

The transmission function uses transmission control protocol (TCP) / user datagram protocol (UDP) protocol for data transmission. Signalling control service is achieved by interfaces Vs and Vd, respectively. Media broadcasting service is achieved by interface Vm, video service is achieved by interface Vr, and device management is achieved by interface Vd.

In the access function, it is required to provide the PU with an authentication service during the access process. The service platform is required to set several security mechanisms, including the identity verification of PUs, the prevention of brute force attacks and replay attacks, etc. It is required that only the PU with successful authentication can access the service platform. It is required that the CU can only obtain the access right of its binding PU.

In the service function, the detailed protocols of interfaces (Vs/Vd/Vm/Vr) between PUs and the service platform are specified as below.

8.3.2 Interface definition

a) Interface Ms: CMU-SCU

It is used by the SCU to report the status, capability of the PU and the alarm information to the CMU. It is also used by the CMU to manage service resources, session status and transferring the session ID for authentication.

b) Interface Sm: SCU-MDU

It is used by the SCU to control the MDU for media distribution, media selection and media location.

c) Interface Cs: SCU-CU

It is used by the CU to send certain video requests to the SCU and to receive the responses.

d) Interface Cm: MDU-CU

It is used by the CU to send a request to the MDU and get the media to stream from the MDU.

e) Interface Cc: CMU-CU

It is used to send a request and return the authentication result, which includes configuration and status query, alarm information, video list and information of cloud storage and front-end storage from the CMU to the CU.

f) Interface Sc: CS-CU

It is used by the CU to send a query, certain video requests, on command and download command to the CS and get live media stream and to record files from the CS when using the on command and download function.

g) Scheduling service interface Vs

Interface Vs complies with HTTPS and JSON. Service platform uses Vs to realize PUs dispatching and signalling access among server rooms.

h) Device management interface Vd

Interface Vd complies with WebSockets and JSON. It defines the control signalling of security devices, including device management, status query, device control, status subscription, live broadcast, network configuration, cloud platform control, alarm reporting, talkback and other functions.

i) Media transmission interface Vm

Interface Vm complies with a secure reliable transport (SRT) protocol based on the user datagram protocol (UDP). It is used for pushing real-time audio and video media stream.

j) Video recording interface Vr

Interface Vr complies with HTTPS and JSON. It defines video recording services, including video segment storage allocation, video content upload, video meta-data upload, playback and other functions.

8.3.3 Interface message definition

The standard interface message using HTTPS and JSON is defined in Table 8-1. Here, the property of direction sets the direction for the message which is sent from the service platform to the PU or from the PU to the service platform. There are several properties contained in the "Request message body" which are request method, request URL, request message header and request content. For response message body, there are mainly three categories, namely, return status code, response message header and response message payload.

Table 8-1 – Interface message for HTTPS+JSON

Protocol type		HTTPS + JSON
Direction		
Request message body	Request method	
	Request URL	
	Request message header	
	Request content	
Response message body	Return status code	
	Response message header	
	Response content	

The standard message for WebSocket protocol and JSON format is defined in Table 8-2. Here, Request name refers to the main purpose of this message. The message type can be either remote procedure call (RPC) or Event type (seen in clause 9.3.2). The property of Direction defines the direction for the message which is sent from the service platform to the PU or from the PU to the service platform. There are several parameters that can be defined in the Request message body according to requirements. For response message body, there are mainly two types, namely, return status code and response content.

Table 8-2 – Interface message for WebSocket and JSON

Request name		
Message type		
Direction		
Request message body (Request content)		
Response message body	Return status code	
	Response content	

9 Protocols for home surveillance system (HSS)

9.1 Requirements of the network transmission function

The protocols defined in this Recommendation are required to support the secure reliable transport (SRT) / user datagram protocol (UDP), WebSocket / TCP and HTTP over TLS [IETF RFC 2818] transmission protocols requirements. The connection between interfaces is required to support HTTP short connection and long connection mechanisms which are in accord with the regulations specified as hypertext transfer protocol – HTTP/1.1 [IETF RFC 7230]. The data transmitted through networks is required to be encrypted using the secure sockets layer (SSL) / transport layer security (TLS) to ensure data security.

9.2 Requirements of access function

The interface information delivered via access function protocols is recommended to use the resource in representation state transfer (REST) architecture, using URI as a unique identification. The major purpose of the access function protocol is to provide authentication and access control for accessing devices. In general, HTTP protocols are recommended for access control. Specifically, a new parameter <Mac-Identity> is recommended to be used to extend the common HTTP header field for representing the requester. Here, the Mac address or similar unique information is required to identify the device when considering the public access networks environment.

9.3 Requirements of service function

This clause will provide a detailed description about the requirements of a service function.

9.3.1 Interface of scheduling service – Vs

A scheduling server sends instructions to PUs through Vs and instructs PUs to connect to the designated signalling server and the peer-to-peer server. The response information is required to contain a timestamp to synchronize the system time of the dispatching server to PUs. Vs uses HTTPS and JSON.

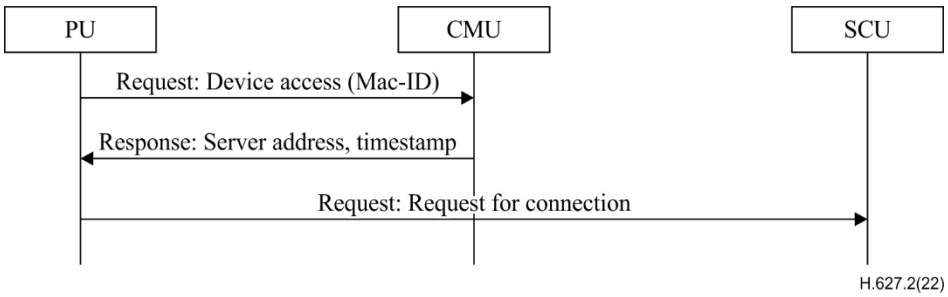

Figure 9-1 – Flow of scheduling service

9.3.1.1 Scheduling service procedure

The procedure of scheduling service shown in Figure 9-1 is as follows:

1 When a PU is successfully connected to the Internet, it is required to send a scheduling Request to a preset address which is configured in SDK in advance.

2 The scheduling service module in CMU is required to check the Mac-ID and relevant information to determine the current signalling server address, P2P server, and current timestamp in the Response message.

3 When receiving the Response, the PU compares the timestamp with the current system time. If the difference is larger than a threshold (i.e., 10 s), the system time is reset with the received timestamp to ensure synchronization.

4 The device sends a connection Request according to the address information contained in the Response to the SCU.

9.3.1.2 Re-scheduling service procedure

The device needs to send a new scheduling request if it is required to restart services. The situations that may cause re-scheduling service is as follows:

1 If the PU is running normally but some network signalling or transmission request fails, it may cause several re-connection requests between the device and the signalling server. When there are multiple re-connection failures (i.e., more than 15 times), the PU is required to send a re-scheduling request.

2 If the PU works normally, the signalling server sends a rescheduling request to the device when assuming asynchronized procedure is agreed upon. The device will initiate a rescheduling procedure if all other sessions are proceeded in normal.

9.3.1.3 Service re-connection procedure

There are some unusual reasons such as abnormal network conditions that can cause connection failures between the devices and the scheduling service module. However, it is still required to avoid re-connection requests congestion in some single moment when many devices send requests for service re-connection. It is recommended to follow the rules as given below:

1 Assuming N represents re-connection tries, the device needs to hibernate some time period [0, 5*N] seconds if $N \leq 12$;

2 The device is required to hibernate for 60 seconds and then sends a re-connection request if $N > 12$.

9.3.2 Interface of device management – Vd

Vd defines the control signalling of PUs, including device management, status query, device control, status subscription, live broadcast, network configuration, cloud platform control, alarm reporting, talkback and other functions. Vd uses WebSocket and JSON protocols.

9.3.2.1 Interface protocols

There are two types of interface messages for device management. One is remote procedure call (RPC) which is called a request-response mode, and the other is Event for event notification without responses.

9.3.2.1.1 RPC message

RPC is used for a request-response mode. Both PU and the service platform are required to initiate RPC requests. Regarding a single direction, the responder is required to process and reply to the request according to the timing order of arrivals. The RPC message template is shown in Table 9-1.

Table 9-1 – RPC message template

Request content	{
	"req": < request name, string, required>;
	"params": < request content, JSON object, optional, values of RPC method>, if there is no value for the method this field does not exist;
	"seq": < sequence number of RPC request message, int, required >, the calculation value adds one for each request;
	}

Return status code	200: normal 400~499: user input error 500~599: server error

Response content	{ //successful calling examples
	"seq": < sequence number of RPC request message, int, required; to be consistent with the sequence number of the corresponding RPC requests>;
	"resp": < response content, JSON object, required>;
	}

	{ //failed call example
	"seq": < sequence number of RPC request message, int, required> to be consistent with the sequence number of the corresponding RPC request
	>,
	"err": {
	"code": < error code, int, required>,
	"msg": < error message, string, required>
	}
	}

The communication procedure of the RPC message is shown in Figure 9-2.

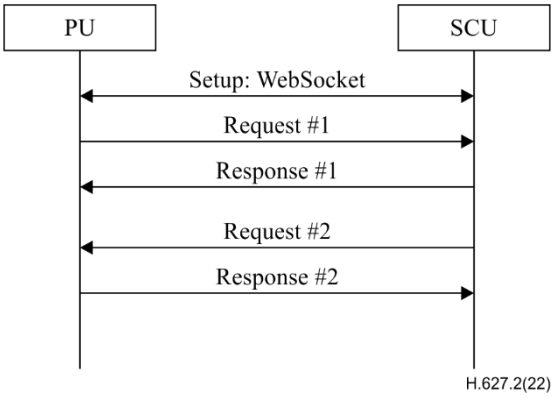

Figure 9-2 – Communication procedure of RPC message

9.3.2.1.2 Event message

Event message works in a no response mode. Both PU and service platform are required to send an Event message to each other. The Event message template is shown in Table 9-2.

Table 9-2 – Event message template

Request content	{
	"event": < request name, string, required>,
	"params": < request content, JSON object, optional, parameter >; this field does not exist if there is no value for the parameter
	}

The communication procedure of the Event message is shown in Figure 9-3.

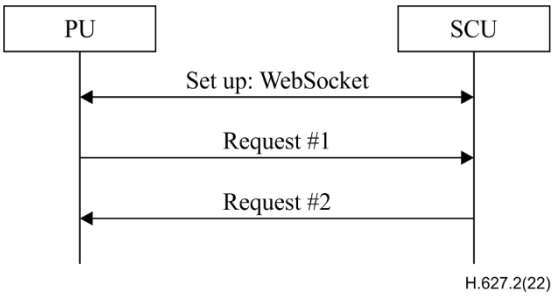

Figure 9-3 – Communication procedure of Event message

9.3.2.2 Messages for Vd

When the connection is establishing, the format of WebSocket URL is shown as below:

wss://<domain:port>/Vd?login_code=<device ID>&login_passwd=<login password>&hardware_model=<hardware model>&firmware_model=<firmware version>&sdk=<SDK version>&reconnect=<reconnect or not>(optional)

Field information is shown in Table 9-3.

Table 9-3 – Detailed information for the above fields

Name	Type	Information
login_code	string	Required: device ID
login_passwd	string	Required: Device login password. If the device ID and login password in the URL do not match the pre-configured information stored in the service platform, the login fails.
hardware_model	string	Required: Hardware model which is customized by the manufacturer is used for problem tracking and remote firmware upgrades.
firmware_model	string	Required: Firmware version which is customized by the manufacturer is used for problem tracking and remote firmware upgrades.
sdk	string	Optional: The current version of the SDK is filled within SDK. If the SDK is not used, this parameter can be omitted. The default value is null.
reconnect	int	Optional: To confirm whether this connection is a reconnect or not. "0" means this connection is the first connection after start up. "1" means this connection is a reconnect after a disconnection. The default value is zero.

Table 9-4 describes detailed messages used for Vd, which contains three parts including message name, message type and detailed description about this message.

Table 9-4 – Detailed information for Vd messages

Message name	Message type	Description
Heartbeat report	RPC	A PU is required to use this message to periodically report its working status to the service platform.
Get server information	RPC	A PU is required to use this message to obtain the IP address from the service platform.
Binding report	EVENT	A PU is required to use this message to send the binding information to the service platform.
Reboot channel	EVENT	The service platform is required to use this message to ask a PU to reboot a specified channel.
Reboot device	EVENT	The service platform is required to use this message to reboot a PU.
Reschedule device	EVENT	The service platform is required to use this message to ask a PU to conduct a re-schedule service procedure.
Upgrade firmware	EVENT	The service platform is required to use this message to launch a firmware upgrade procedure for a PU.
Upgrade status query	RPC	The service platform is required to use this message to inquire and present the upgrade status of a PU, such as the process bar.
Set device time	RPC	The service platform is required to use this message to set the date and time of a PU.
Query device time	RPC	The service platform is required to use this message to inquire about the date and time of a PU.
Query device status	RPC	The service platform is required to use this message to inquire about the running status of a PU.
Push live video	RPC	The service platform is required to use this message to ask a PU to push a live video stream to an assigned URL.
Stop live video	RPC	The service platform is required to use this message to ask a PU to stop pushing live video.
Start recording	RPC	The service platform is required to use this message to ask a PU to make records and upload them to the cloud storage.
Stop recording	RPC	The service platform is required to use this message to ask a PU to stop recording.
Alarm notification	RPC	A PU is required to use this message to report alarm events.
PTZ control	EVENT	The service platform is required to use this message to control Pan/Tilt/Zoom (PTZ).
Get list	RPC	The service platform is required to use this message to obtain the PTZ preset list of a PU.
Get configuration	RPC	The service platform is required to use this message to obtain the configuration of a PU.
Set configuration	RPC	The service platform is required to use this message to set the configuration of a PU.

Table 9-4 – Detailed information for Vd messages

Message name	Message type	Description
Reset configuration	RPC	The service platform is required to use this message to reset the configuration of a PU.
Snapshot	RPC	The service platform is required to use this message to take a snapshot of the camera screen and upload it to the service platform.
Start playing	EVENT	The service platform is required to use this message to ask a PU to play a video.
Control playing	EVENT	The service platform is required to use this message to control video playing.
Query playing status	RPC	The service platform is required to use this message to inquire about the playing state of a PU.
Keep awake	RPC	The service platform is required to use this message to prevent a PU and its specified channel from hibernating within a specified time or wake up a PU and its specified channel.
Battery reporting	EVENT	A PU is required to use this message to report the current power to the service platform.
Get capability list	RPC	The service platform is required to use this message to obtain the capability list of a PU.
Upload log	RPC	The service platform is required to use this message to control a PU to upload its logs for debugging and analysis.
Upload log done	EVENT	A PU is required to use this message to inform the service platform that the log file is uploaded.
Get disk information	RPC	The service platform is required to use this message to get the SD card status of a PU.
Format disk	RPC	The service platform is required to use this message to format the SD card of a PU.

9.3.3 Interface of media broadcasting service – Vm

When the SCU receives a command for publishing the video, a PU uploads the video to the service platform. If SCU receives a command for stopping, the PU stops uploading. The media broadcasting service interface Vm uses the SRT protocol for providing media broadcasting services.

9.3.3.1 Specification of interface Vm

Vm is responsible for media data transmission while data transmission control is realized by the device management interface (Vd).

9.3.3.1.1 Media broadcasting service procedure

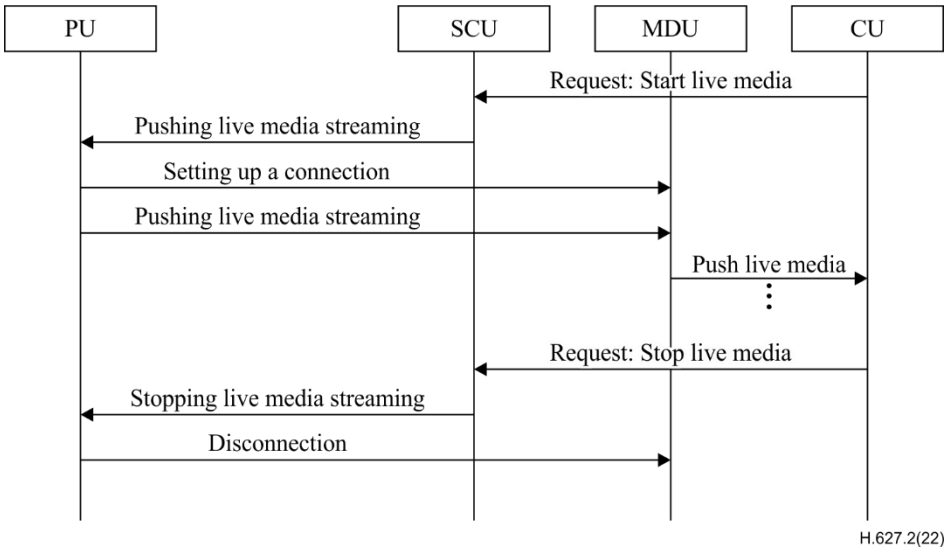

Figure 9-4 – Media broadcasting service procedure

The media broadcasting service procedure is shown in Figure 9-4. The steps are as follows:

1 Users send the "start live media" request on the CU if they want to watch the live media streaming;

2 After receiving the "start live media streaming" request from the CU, the service platform (SCU) sends a "push live media" command to the PU;

3 The PU sets up a connection with the MDU, after receiving the address;

4 After the connection is established, the PU pushes a live media streaming to the MDU by using the SRT protocol. Then the media server (MS) pushes the live media streaming to the CU;

5 Users send "stop live media" requests on the CU to the SCU;

6 After receiving the "stop live media" request from the CU, the service platform (SCU) sends a "stop live media" command to the PU;

7 The PU stops pushing the live media streaming and disconnects with the media server after it receives the command.

9.3.3.2 Messages of Vm

9.3.3.2.1 Starting media broadcasting service

The service platform sends the "pushing live media streaming" command to the PU via Vd. After receiving the command, the PU pushes a live media streaming to the dedicated URL address, which is embedded in the command. The messages used here are required to support the SRT protocol. Please refer to clause 9.3.3.1.1 for more information.

9.3.3.2.2 Stopping media service

The service platform sends the "stopping live media streaming" command to the PU via Vd. After the PU receives the command, it stops pushing the live media streaming. The messages used here are required to support the SRT protocol. Please refer to clause 9.3.3.1.1 for more information.

9.3.4 Interface of video service – Vr

The video service uses the interface Vr to realize the functions of video storage and distribution, video content upload, video metadata upload, and look back. When Vr receives a command of starting a record, a PU uploads the video to the storage device. After receiving a command of stopping the service, the PU stops uploading. Vr uses HTTPS and JSON protocols.

9.3.4.1 Specification of interface Vr

9.3.4.1.1 Video storage procedure

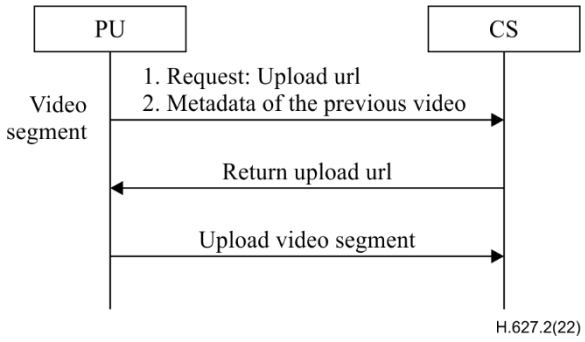

Figure 9-5 – Video storage procedure

The entire procedure of video storage is shown in Figure 9-5. The steps are as follows:

1) For a video segment, the PU requests an upload URL from the service platform and reports the metadata of the latest video segment if there was already a previous video segment.
2) The service platform returns an upload URL and records the metadata of the latest video segment.
3) The PU uploads the video segment to the CS.
4) Repeat steps from 1 to 3.

9.3.4.1.2 Exceptional handling

Note that some network abnormal issues may lead to video segment uploading failures. The PU is required to re-upload several times as possible. If the final uploading fails, the PU is required to send a message to the CS to notify that the task of video segment uploading finally fails. Then, the service platform will clean up all the residual information about this video segment.

9.3.4.1.3 Uploading confirmation

In general, there is a two-stage strategy defined for video segment uploading. For each segment uploading task, the interface Vr has to be called twice. One is for the previous segment confirmation and another is for the next segment creation. To improve efficiency, it is recommended to confirm the two-stage requests at the same time since the video segments are consecutively uploaded.

9.3.4.2 Message of Vr

The PU relies on the interface Vr to call the storage service using HTTPS and JSON. Each PU is required to establish a corresponding channel. To identify each video storage task, a specific HTTPS URI for each video channel must be assigned.

For example,

https://<domain:port>/live/cseg/<channel ID>.

Here, the channel ID is the channel identification assigned by the service platform for each PU. This ID is used to distinguish the video requests of different PUs. The PU uses the POST method to initiate a request and the response is returned in the JSON format.

Table 9-5 – Detailed information for Vr messages

Message name	Message type	Description
Create video segment	HTTPS+JSON	A PU is required to use this message to request the service platform to allocate resources for storing a video segment.
Confirm video segment	HTTPS+JSON	A PU is required to use this message to confirm with the service platform the information of a video segment.
Cancel video segment	HTTPS+JSON	A PU is required to use this message to notify the service platform of which video segment is to be cancelled.
Get server information	HTTPS+JSON	A PU is required to use this message to obtain the date and time of the service platform.
Report statistical data	HTTPS+JSON	A PU is required to use this message to report the statistical data of the video segment within a time interval.

Appendix I

Interface message examples

(This appendix does not form an integral part of this Recommendation.)

I.1 Messages of the scheduling service interface

Table I.1 – Specification of scheduling service interface

Protocol	HTTPS + JSON
Direction	From PU to platform
Request method	GET
Request URL	https://<domain:port>/Vs/getServerAddr?macid=XXXXX
Request message header	None

Request content	{
	"mac_id": <required, string>
	}

Return status code	200: normal 400~499: user input error 500~599: service error

Response content	{//successful response
	"PassDomain":"www.test01.com ", //signalling server address
	"P2p_passDomain":"www.test02.com ", // P2P server address
	"TurnDomain": "www.test03.com ", //turn server address
	"HibernationDomain": "www.test04.com", //hibernation server address
	"PassPort": xxxx, //signalling server ws interface(non-encryption interface)
	"Secure_PassPort": xxxx, //signalling server wss interface(send non-zero value if the PU supports wss protocol)
	"P2p_passPort": xxxx, //P2P interface
	"TurnPort": xxxx, //turn server port
	"SyncTime": xxxxx //current time of the service platform
	}

	{//failed response
	"info"： <string, required: error description>
	}

I.2 Messages of device management interface

I.2.1 An example of keeping alive report

Table I.2-1 – Keep alive report message

Request name	Keepalive
Message type	RPC
Direction	From PU to platform

Table I.2-1 – Keep alive report message

<table>
<tr><td>Request content</td><td>{
"mac_id": <required, string>
"state": <required, int; device status,1:online, 2: upgrading, 3: error>
"channels": [{
"channel": <required, int>,
"state": <required, int; channel status, 0: offline, 1: online, 3: error, 4: disable>,
"stream_id": <optional, string; channel stream_id represents the channel which is currently used for streaming. Note if stream_id is null or does not exist, the corresponding channel cannot be used for streaming>
"record_session": <optional, string; record_session represents the index of session being currently used for recording in the service platform. Note if the record_session is null or does not exist, the PU is not recording>
"alarm": <optional, int; flags, each bit position of current alarm status shown in flags represents one alarm type,
Flag = 1, the corresponding type of alarm is triggered
More information about specific values is defined in the table of alarm status flags>
}]
}</td></tr>
<tr><td>Return status code</td><td>200: normal
400~499: user input error
500~599: server error</td></tr>
<tr><td>Response content</td><td>flags
<table>
<tr><td>alarm type</td><td>position of bit</td></tr>
<tr><td>external alarm</td><td>the second bit</td></tr>
<tr><td>motion detection</td><td>the third bit</td></tr>
<tr><td>intrusion detection</td><td>the fourth bit</td></tr>
<tr><td>cry detection</td><td>the fifth bit</td></tr>
<tr><td>face recognition</td><td>the sixth bit</td></tr>
<tr><td>sound detection</td><td>the seventh bit</td></tr>
<tr><td>low battery</td><td>the eighth bit</td></tr>
<tr><td>lock breaking</td><td>the ninth bit</td></tr>
<tr><td>ring the bell</td><td>the tenth bit</td></tr>
<tr><td>human detection</td><td>the eleventh bit</td></tr>
</table></td></tr>
</table>

I.2.2 An example of getting server information

Table I.2-2 – Get server information

<table>
<tr><td>Request name</td><td>GetServerInfo</td></tr>
<tr><td>Message type</td><td>RPC</td></tr>
<tr><td>Direction</td><td>From PU to platform</td></tr>
<tr><td>Request content</td><td>{
"mac_id": <required, string>
"state": <required, int; device status, 1: online, 2: upgrading, 3: error>
}</td></tr>
</table>

Table I.2-2 – Get server information

Return status code	200: normal 400~499: user input error 500~599: server error

Response content	{
	"hibernation": <optional, string; format of hibernation server address, IP: port, Hibernation is not available if this field is null or non-exist>,
	"hb_interval": <optional, int: device heartbeat interval(sec), the default is 10 seconds if this field does not exist>,
	"hb_token": <optional, string: hibernation service token, length is no more than 16 characters. Token is not required for hibernation service if this field does not exist>
	}

I.2.3 An example of getting binding report

Table I.2-3 – Binding report

Request name	Bind
Message type	Event
Direction	From PU to platform

Request content	{
	"mac_id": <required, string>
	"bind_id": <required, string>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.4 An example of rebooting channel

Table I.2-4 – Reboot channel

Request name	RebootChannel
Message type	Event
Direction	From platform to PU

Request content	{
	"channel": <required, int>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.5 An example of rebooting device

Table I.2-5 – Reboot device

Request name	RebootDevice
Message type	Event
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.6 An example of rescheduling device

Table I.2-6 – Reschedule device

Request name	Reschedule
Message type	Event
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.7 An example of upgrading firmware

Table I.2-7 – Upgrade firmware

Request name	UpgradeFirmware
Message type	Event
Direction	From platform to PU

Request content	{
	"firmware_model": <required, string>
	"url": <required, string>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.8 An example of querying upgrading status

Table I.2-8 – Query upgrade status

Request name	QueryUpgrade
Message type	RPC
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error

Response content	{
	"upgrade_status": <required, string; upgrading status: notstart/downloading/installing/done/error>
	"progress": <required, int; current progress, 0-100>
	"firmware_model": <required, string>
	}

I.2.9 An example of setting device time

Table I.2-9 – Set device time

Request name	Settime
Message type	RPC
Direction	From platform to PU

Request content	{
	"datetime": <required, string; format: yyyy-MM-ddTHH:mm:ss>
	"tz": <optional, int: time zone code>
	"offset": <required, int>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.10 An example of querying device time

Table I.2-10 – Query device time

Request name	QueryTime
Message type	RPC
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error

Table I.2-10 – Query device time

Response content	{
	"datetime": <required, string; format: yyyy-MM-ddTHH:mm:ss>
	"tz": <optional, int; time zone code>
	}

I.2.11 An example of querying device status

Table I.2-11 – Query device status

Request name	DevStatus
Message type	RPC
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error

Response content	{
	"dev_id": <required, string: device ID>
	"hardware_model": <required, string: device model>
	"firmware_model": <required, string: device firmware version number>
	"last_upgrade_datetime": <required, string; last successfully upgrading time, format: YY-MM-DDTHH:MM:SS, e.g., 2016-12-05T02:15:32>
	"wifi_ssid": <optional, string: the ssid of the wifi currently connected, the absence of this field or an empty string means that the device is not connected to wifi>
	"wifi_signal": <optional, int: the current wifi signal strength of the device >
	"up_bandwidth": <optional, int: maximum uplink bandwidth detected by the device, unit bps, if it does not exist, it means the uplink bandwidth is unknown>
	"down_bandwidth": <optional, int: minimum downlink bandwidth detected by the device, unit bps, if it does not exist, it means the downlink bandwidth is unknown>
	"ip_addr": <optional, string: the current internal network IP address of the device, if it does not exist, it means the IP address is unknown>
	"mac_addr": <optional, string: the mac address of the device network card, if it does not exist, it means the mac address is unknown>
	"battery": <optional, int: current device battery level, 0-100 >
	}

I.2.12 An example of pushing live video

Table I.2-12 – Push live video

Request name	PushLive
Message type	RPC
Direction	From platform to PU

Table I.2-12 – Push live video

Request content	{
	"channel": <required, int>
	"url": <required, string; preferred destination URL1 for streaming>
	"stream_id": <required, string; the ID used by the platform to identify this stream>
	"max_bitrate": <required, int >
	}

Return status code	200: normal 400~499: user input error 500~599: server error 11: the code stream is too large 12: parameter error 101: unsupported RPC method
Response content	Pushing stream is successful, or the stream already exists.

I.2.13 An example of stopping live video

Table I.2-13 – Stop live video

Request name	StopLive
Message type	RPC
Direction	From platform to PU

Request content	{
	"stream_id": <required, string; the stream_id when push live video>
	"channel": <required, int>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	Stop pushing is completed, or the stream does not exist.

I.2.14 An example of starting recording

Table I.2-14 – Start recording

Request name	StartRecord
Message type	RPC
Direction	From platform to PU
Request content	{ "session_id": <required, string; the corresponding record session ID of the platform> "channel": <required, int> "max_bitrate": <required, int > "seg_duration": <required, int; video segment duration> "seg_max_size": <required, int; maximum size of each video segment> "seg_max_count": <required, int; maximum number of video segments cached in memory>

Table I.2-14 – Start recording

	"prerecord_seconds": <optional, int; pre-recorded seconds> "start_ts": <required, float; timestamp of the first video segment> "cbk_url": <required, callback URL, maximum is 256 bytes; the video module obtains the upload address by requesting the url> }
Return status code	200: normal 400~499: user input error 500~599: server error 11: the code stream is too large 12: parameter error 101: unsupported RPC method
Response content	None

I.2.15 An example of stopping record

Table I.2-15 – Stop recording

Request name	StopRecord
Message type	RPC
Direction	From platform to PU

Request content	{
	"session_id": <required, string; the corresponding recording session ID of the platform>
	"channel": <required, int>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	Stop recording is completed, or the record session does not exist.

I.2.16 An example of alarming notification

Table I.2-16 – Alarm notification

Request name	AlarmNotify
Message type	RPC
Direction	From PU to platform

Table I.2-16 – Alarm notification

<table>
<tr><td>Request content</td><td>{
"channel": <required, int>
"timestamp": <required, int>
"type": <required, int; alarm type: see the table below>
"state": <required, string enumeration, start/end/pulse>
"desc": <required, string; description>
}
Alarm type value:
<table>
<tr><td>Type</td><td>Type value</td><td>State/Single Trigger</td></tr>
<tr><td>External alarm</td><td>2</td><td>state</td></tr>
<tr><td>Motion detection</td><td>3</td><td>state</td></tr>
<tr><td>Mixed net</td><td>4</td><td>single</td></tr>
<tr><td>Cry detection</td><td>5</td><td>state</td></tr>
<tr><td>Face recognition</td><td>6</td><td>state</td></tr>
<tr><td>Sound detection</td><td>7</td><td>state</td></tr>
<tr><td>Low power</td><td>8</td><td>single</td></tr>
<tr><td>Lock picking</td><td>9</td><td>single</td></tr>
<tr><td>Bell ring</td><td>10</td><td>single</td></tr>
<tr><td>Human detection</td><td>11</td><td>state</td></tr>
<tr><td>PIR motion detection</td><td>12</td><td>state</td></tr>
</table></td></tr>
<tr><td>Return status code</td><td>200: normal
400~499: user input error
500~599: server error</td></tr>
<tr><td>Response content</td><td>{
"pic_upload_url": <optional, string>
}</td></tr>
</table>

I.2.17 An example of PTZ control

Table I.2-17 – PTZ control

<table>
<tr><td>Request name</td><td>CtrlPTZ</td></tr>
<tr><td>Message type</td><td>EVENT</td></tr>
<tr><td>Direction</td><td>From platform to PU</td></tr>
<tr><td>Request content</td><td>{
"channel": <required, int>
"op": <required, string; PTZ operation code, see the table below for details>
"value": <optional, int; the table is shown in the table below>
}
<table>
<tr><td>operation</td><td>value</td></tr>
<tr><td>up</td><td>Optional int speed, 0-100, 0 slowest, 100 fast, default 50</td></tr>
<tr><td>down</td><td>Same as above</td></tr>
</table></td></tr>
</table>

Table I.2-17 – PTZ control

	left	Same as above
	right	Same as above
	upleft	Same as above
	upright	Same as above
	downleft	Same as above
	downright	Same as above
	zoomin	Same as above
	zoomout	Same as above
	stop	optional, int, but the value is meaningless
	goto_preset	Preset ID, 0-255
	set_preset	Same as above
	clear_preset	Same as above
	up_step	optional, int, single step, 0-100, min 0, max 100, default 0
	down_step	Same as above
	left_step	Same as above
	right_step	Same as above
	upleft_step	Same as above
	upright_step	Same as above
	downleft_step	Same as above
	downright_step	Same as above
	zoomin_step	Same as above
	zoomout_step	Same as above
Return status code	200: normal 400~499: user input error 500~599: server error	
Response content	None	

I.2.18 An example of getting PTZ preset list

Table I.2-18 – Get PTZ preset list

Request name	GetPTZPresetList
Message type	RPC
Direction	From platform to PU

Request content	{
	"channel": <required, int>
	}

Return status code	200: normal 400~499: user input error 500~599: server error

Table I.2-18 – Get PTZ preset list

Response content	"presetList":
	[
	{
	"ptzId": <required, int; preset ID, 0-255>
	}
	...
	]

I.2.19 An example of getting device configuration

Table I.2-19 – Get configuration

Request name	GetConfig
Message type	RPC
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error
Response content	Configuration set

I.2.20 An example of setting device configuration

Table I.2-20 – Set configuration

Request name	SetConfig
Message type	RPC
Direction	From platform to PU
Request content	User defined
Return status code	200: normal 400~499: user input error 500~599: server error 12: parameter error 13: unsupported configuration 102: channel does not exist
Response content	Configuration set

I.2.21 An example of resetting device configuration

Table I.2-21 – Reset configuration

Request name	ResetConfig
Message type	RPC
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.22 An example of snapshot

Table I.2-22 – Snapshot

Request name	Snapshot
Message type	RPC
Direction	From platform to PU

Request content	{
	"channel": <required, int>
	"url": <required, string; the url of uploaded picture>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.23 An example of starting playing a video

Table I.2-23 – Start playing

Request name	PlayVideo
Message type	EVENT
Direction	From platform to PU

Request content	{
	"channel": <required, int>
	"url": <required, string; the url of video file>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.24 An example of controlling playing

Table I.2-24 – Control playing

Request name	CtrlVideo
Message type	EVENT
Direction	From platform to PU

Request content	{
	"channel": <required, int>
	"op": <required, string; play control instructions, stop/pause/resume supported currently>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.25 An example of querying playing status

Table I.2-25 – Query playing status

Request name	QueryVideoStatus
Message type	RPC
Direction	From platform to PU

Request content	{
	"channel": <required, int>
	}

Return status code	200: normal 400~499: user input error 500~599: server error

Response content	{
	"status": <required, string; play status, stopped, pausing, playing>
	"url": <optional, string; URL of the video file current playing, if this field is empty or does not exist, it means that it is not currently playing>
	}

I.2.26 An example of keeping awaken

Table I.2-26 – Keep awake

Request name	KeepAwaken
Message type	RPC
Direction	From platform to PU

Request content	{
	"channel": <required, int>
	"expired": <required, int: minimum time to keep power on, unit is second>
	}

Table I.2-26 – Keep awake

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.27 An example of battery change

Table I.2-27 – Battery reporting

Request name	BatteryChange
Message type	EVENT
Direction	From platform to PU

Request content	{
	"battery": <required, int: current device battery power percentage, 0-100>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.28 An example of getting device capability list

Table I.2-28 – Get capability list

Request name	GetCap
Message type	RPC
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.29 An example of uploading device log

Table I.2-29 – Upload log

Request name	LogUploadAsync
Message type	RPC
Direction	From platform to PU

Table I.2-29 – Upload log

Request content	{
	"trans_id": <required, int: mark the task ID of this log upload>
	"start": <required, string; logging start time, format yyyy-MM-ddTHH:mm:ss>
	"end": <required, string; logging end time, format yyyy-MM-ddTHH:mm:ss>
	"url": <required, string, URL of log upload,the device upload the corresponding log file through the PUT method>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.30 An example of uploading device log done

Table I.2-30 – Upload log done

Request name	LogDone
Message type	EVENT
Direction	From PU to platform

Request content	{
	"mac_id": <required, string, the mac address of this device, test uniqueness>
	"trans_id": <required, string; mark the task ID of this log upload, consistent with LogUploadAsync>
	"status": <required, string enumeration; success/fail>
	"start": <required, string; logging end time, format yyyy-MM-ddTHH:mm:ss>
	"end": <required, string; logging end time, format yyyy-MM-ddTHH:mm:ss>
	"url": <required, string, url of log uploaded>
	"size": < required, int; the number of bytes of the uploaded log file>
	}

Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.2.31 An example of getting disk information

Table I.2-31 – Get disk information

Request name	GetDiskInfo
Message type	RPC
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error

Table I.2-31 – Get disk information

Response content	{
	"state": <required, int; the state of SD card, 0: normal, 1: no card inserted, 2:card exception, 3: formatting>
	"total": <required, int; total capability of the SD card, unit is MB>
	"free": <required, int; left capability of the SD card, unit is MB>
	}

I.2.32 An example of formatting disk

Table I.2-32 – Format disk

Request name	FormatDisk
Message type	RPC
Direction	From platform to PU
Request content	None
Return status code	200: normal 400~499: user input error 500~599: server error
Response content	None

I.3 Messages of video service interface

I.3.1 An example of creating video segment

Table I.3-1 – Create video segment

Protocol type	HTTPS + JSON
Direction	From PU to platform
Request method	POST
Request URL	"upload address" returned by the platform, e.g., https://<domain:port>/live/cseg/<channelID>
Request message head	Content-Type: application/x-www-form-urlencoded

Request content	{
	"op": <string enumeration, required: create video segment are fixed as create>
	"content_type": <string, required>
	"size": <int,required: the size of the video segment, unit is byte>
	"start": <float,required: the start time of the segment recording, recording time from epoch to the start of the slice, unit is second>
	"duration": <float,required: the duration time of this segmentation, unit is second>
	"discontinue": <int, required: (0, 1)>
	"https": <int, required>
	}
	The example string of the body:
	op=create&content_type=video%2Fmp2t&size=32768&start=1574322174&duration=600&discontinue=0&https=0

Table I.3-1 – Create video segment

Return status code	200: normal 400~499: user input error 500~599: server error
Response header	Content-Type: application/json; charset=utf-8 Access-Control-Allow-Origin: * Access-Control-Allow-Methods: POST

Response message content	Normal response:
	{
	"name": <string, required>
	"url": <string, required>
	}
	Abnormal response:
	{
	"info": <string, required: wrong description>
	}

I.3.2 An example of confirm video segment

Table I.3-2 – Confirm video segment

Protocol type	HTTPS + JSON
Direction	From PU to platform
Request method	POST
Request URL	"upload address" returned by the platform, e.g., https://<domain:port>/live/cseg/<channelID>
Request message head	Content-Type: application/x-www-form-urlencoded

Request content	{
	"op": <string enumeration, required>
	"name": <string, required>
	}
	The example of the body string: op=save&name=0003.ts

Return status code	200: normal 400~499: user input error 500~599: server error
Response header	Content-Type: application/json; charset=utf-8 Access-Control-Allow-Origin: * Access-Control-Allow-Methods: POST

Response message content	Normal response:
	None
	Abnormal response:
	{
	"info": <string, required: wrong description>
	}

I.3.3 An example of cancelling a video segment

Table I.3-3 – Cancel video segment

Protocol type	HTTPS + JSON
Direction	From PU to platform
Request method	POST
Request URL	"upload address" returned by the platform, e.g., https://<domain:port>/live/cseg/<channelID>
Request message head	Content-Type: application/x-www-form-urlencoded

Request content	{
	"op": <string enumeration, required: cancel video segment are fixed as fail>
	"name": <string, required>
	}
	The example of the body string: op=fail&name=0003.ts

Return status code	200: normal 400~499: user input error 500~599: server error
Response header	Content-Type: application/json; charset=utf-8 Access-Control-Allow-Origin: * Access-Control-Allow-Methods: POST

Response message content	Normal response:
	None
	Abnormal response:
	{
	"info": <string, required: wrong description>
	}

I.3.4 An example of getting server information

Table I.3-4 – Get server information

Protocol type	HTTPS + JSON
Direction	From PU to platform
Request method	POST
Request URL	"upload address" returned by the platform, e.g., https://<domain:port>/live/cseg/<channelID>
Request message head	Content-Type: application/x-www-form-urlencoded

Request content	{
	"op": <string enumeration, required; operation type, getting the current calendar time is fixed as get current time>
	}
	The example of the body string: op=getcurrenttime

Return status code	200: normal 400~499: user input error 500~599: server error

Table I.3-4 – Get server information

Response header	Content-Type: application/json; charset=utf-8 Access-Control-Allow-Origin: * Access-Control-Allow-Methods: POST

Response message content	Normal response:
	{
	"currenttime":<float, required: the number of seconds from epoch to current, unit is second>
	}
	Abnormal response:
	{
	"info": <string, required: wrong description>
	}

I.3.5 An example of report statistical data

Table I.3-5 – Report statistical data

Protocol type	HTTPS + JSON
Direction	From PU to platform
Request method	POST
Request URL	"upload address" returned by the platform, e.g., https://<domain:port>/live/cseg/<channelID>
Request message head	Content-Type: application/x-www-form-urlencoded

Request content	{
	"op": <string enumeration, required; operation type>
	"Dropped_nb": <int, optional>
	"Dropped_time": <float, optional>
	}
	The example of the body string: op=stat&dropped_nb=3&dropped_time=30

Return status code	200: normal 400~499: user input error 500~599: server error
Response header	Content-Type: application/json; charset=utf-8 Access-Control-Allow-Origin: * Access-Control-Allow-Methods: POST

Response message content	Normal response:
	None
	Abnormal response:
	{
	"info": <string, required: wrong description>
	}

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2022