ITUPublications
Recommendations

International Telecommunication Union
Standardization Sector

Recommendation

ITU-T J.1206 (01/2024)

SERIES J: Cable networks and transmission of television, sound programme and other multimedia signals

Smart TV operating system

Smart television operating system – Application programming interface

ITU-T J-SERIES RECOMMENDATIONS

Cable networks and transmission of television, sound programme and other multimedia signals

GENERAL RECOMMENDATIONS	J.1-J.9
GENERAL SPECIFICATIONS FOR ANALOGUE SOUND-PROGRAMME TRANSMISSION	J.10-J.19
PERFORMANCE CHARACTERISTICS OF ANALOGUE SOUND-PROGRAMME CIRCUITS	J.20-J.29
EQUIPMENT AND LINES USED FOR ANALOGUE SOUND-PROGRAMME CIRCUITS	J.30-J.39
DIGITAL ENCODERS FOR ANALOGUE SOUND-PROGRAMME SIGNALS – PART 1	J.40-J.49
DIGITAL TRANSMISSION OF SOUND-PROGRAMME SIGNALS	J.50-J.59
CIRCUITS FOR ANALOGUE TELEVISION TRANSMISSION	J.60-J.69
ANALOGUE TELEVISION TRANSMISSION OVER METALLIC LINES AND INTERCONNECTION WITH RADIO-RELAY LINKS	J.70-J.79
DIGITAL TRANSMISSION OF TELEVISION SIGNALS	J.80-J.89
ANCILLARY DIGITAL SERVICES FOR TELEVISION TRANSMISSION	J.90-J.99
OPERATIONAL REQUIREMENTS AND METHODS FOR TELEVISION TRANSMISSION	J.100-J.109
INTERACTIVE SYSTEMS FOR DIGITAL TELEVISION DISTRIBUTION (DOCSIS FIRST AND SECOND GENERATIONS)	J.110-J.129
TRANSPORT OF MPEG-2 SIGNALS ON PACKETIZED NETWORKS	J.130-J.139
MEASUREMENT OF THE QUALITY OF SERVICE – PART 1	J.140-J.149
DIGITAL TELEVISION DISTRIBUTION THROUGH LOCAL SUBSCRIBER NETWORKS	J.150-J.159
IPCABLECOM (MGCP-BASED) – PART 1	J.160-J.179
DIGITAL TRANSMISSION OF TELEVISION SIGNALS – PART 1	J.180-J.189
CABLE MODEMS AND HOME NETWORKING	J.190-J.199
APPLICATION FOR INTERACTIVE DIGITAL TELEVISION – PART 1	J.200-J.209
INTERACTIVE SYSTEMS FOR DIGITAL TELEVISION DISTRIBUTION (DOCSIS THIRD TO FIFTH GENERATIONS)	J.210-J.229
MULTI-DEVICE SYSTEMS FOR CABLE TELEVISION	J.230-J.239
MEASUREMENT OF THE QUALITY OF SERVICE – PART 2	J.240-J.249
DIGITAL TELEVISION DISTRIBUTION THROUGH LOCAL SUBSCRIBER NETWORKS	J.250-J.259
IPCABLECOM (MGCP-BASED) – PART 2	J.260-J.279
DIGITAL TRANSMISSION OF TELEVISION SIGNALS – PART 2	J.280-J.289
CABLE SET-TOP BOX	J.290-J.299
APPLICATION FOR INTERACTIVE DIGITAL TELEVISION – PART 2	J.300-J.309
MEASUREMENT OF THE QUALITY OF SERVICE – PART 3	J.340-J.349
IPCABLECOM2 (SIP-BASED) – PART 1	J.360-J.379
DIGITAL TRANSMISSION OF TELEVISION SIGNALS – PART 3	J.380-J.389
MEASUREMENT OF THE QUALITY OF SERVICE – PART 4	J.440-J.449
IPCABLECOM2 (SIP-BASED) – PART 2	J.460-J.479
DIGITAL TRANSMISSION OF TELEVISION SIGNALS – PART 4	J.480-J.489
TRANSPORT OF LARGE SCREEN DIGITAL IMAGERY	J.600-J.699
SECONDARY DISTRIBUTION OF IPTV SERVICES	J.700-J.799
MULTIMEDIA OVER IP IN CABLE	J.800-J.899
TRANSMISSION OF 3-D TV SERVICES	J.900-J.999
CONDITIONAL ACCESS AND PROTECTION	J.1000-J.1099
SWITCHED DIGITAL VIDEO OVER CABLE NETWORKS	J.1100-J.1119
SMART TV OPERATING SYSTEM	**J.1200-J.1209**
IP VIDEO BROADCAST	J.1210-J.1219
CLOUD-BASED CONVERGED MEDIA SERVICES FOR IP AND BROADCAST CABLE TELEVISION	J.1300-J.1309
TELEVISION TRANSPORT NETWORK AND SYSTEM DEPLOYMENT IN DEVELOPING COUNTRIES	J.1400-J.1409
ARTIFICIAL INTELLIGENCE (AI) ASSISTED CABLE NETWORKS	J.1600-J.1649

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

Recommendation ITU-T J.1206

Smart television operating system – Application programming interface

Summary

Recommendation ITU-T J.1206 specifies the application programming interface of a smart television (TV) operating system over integrated broadcast and broadband cable networks. A smart TV operating system is intended to be installed in an integrated broadcast and broadband (IBB)-capable cable set-top box (STB) and TV and to enable broadcasting and IP-based interactive services provided by cable television operators and third-party providers. By running a smart TV operating system, the IBB-capable cable STB and TV will be able to intelligently provide subscribers with advanced and personalized services by downloading and installing advanced and personalized applications from cable operators' platforms and third-party platforms, which are interconnected with the related cable operators' platforms.

Recommendation ITU-T J.1206 specifies the application programming interface of a smart TV operating system over integrated broadcast and broadband cable networks, including Java application programming interface and web application programming interface, and conforms to the requirements of Recommendations ITU-T J.1201 and Recommendation ITU-T J.1202. More information can be found in the Recommendations about the specification (Recommendation ITU-T J.1203), security framework (Recommendation ITU-T J.1204) and hardware abstract layer (HAL) interface (Recommendation ITU-T J.1205) of smart TV operating system.

History *

Edition	Recommendation	Approval	Study Group	Unique ID
1.0	ITU-T J.1206	2024-01-13	9	11.1002/1000/15802

Keywords

Application programming interface, API, broadband cable, smart TV.

* To access the Recommendation, type the URL https://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID.

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 J.1206

Smart television operating system – Application programming interface

1 Scope

This Recommendation specifies the application programming interface of a smart TV operating system over integrated broadcast and broadband cable networks. The smart TV operating system is intended to be installed in an integrated broadcast and broadband (IBB)-capable cable set-top box (STB) and TV and to enable broadcasting and IP-based interactive services provided by cable television operators and third-party providers. By running the smart TV operating system, the IBB-capable cable STB and TV will be able to intelligently provide subscribers with advanced and personalized services by downloading and installing advanced and personalized applications (apps) from cable operators' platforms and third-party platforms which are interconnected with the related cable operators' platforms.

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 J.1201] Recommendation ITU-T J.1201 (2022), *Smart television operating system – Functional requirements*.

[ITU-T J.1202] Recommendation ITU-T J.1202 (2022), *Smart television operating system – Architecture*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 integrated broadcast and broadband (IBB) DTV service [b-ITU-T J.205]: A service that simultaneously provides an integrated experience of broadcasting and interactivity relating to media content, data and applications from multiple sources, where the interactivity is sometimes associated with broadcasting programmes.

3.1.2 smart television operating system (TVOS) [ITU-T J.1201]: A system software running on an integrated broadcast and broadband-capable (IBB-capable) cable set top box (STB) and television (TV) that is capable of managing hardware, software and data resources of the IBB-capable cable STB and TV, supporting and controlling the application software execution.

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

API	Application Programming Interface
App	Application
AV	Audio Video
CA	Certification Authority
CAS	Conditional Access System
CSS	Cascading Style Sheets
DCAS	Downloadable Condition Access System
DHCP	Dynamic Host Configuration Protocol
DRM	Digital Rights Management
DTMB	Digital Television Terrestrial Multimedia Broadcast
DTV	Digital Television
DVB	Digital Video Broadcasting
ECM	Entitlement Control Message
EMM	Entitlement Management Message
EPG	Electronic Program Guide
ES	Elementary Stream
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IPTV	IP Television
JS	JavaScript
LAN	Local Area Network
MPEG	Moving Picture Experts Group
NTP	Network Time Protocol
NVOD	Near Video On Demand
NVM	Non-Volatile Memory
OSD	On-Screen Display
OTA	Over The Air
PID	Packet Identifier
TA	Trusted Application
Tapp	Trusted Application
TEE	Trusted Execution Environment
TS	Transport Stream

5 Conventions

In this Recommendation:

The phrase "is required to" indicates a requirement which must be strictly followed and from which no deviation is permitted if conformity with this Recommendation is to be claimed.

The phrase "is recommended" indicates a requirement which is recommended but which is not absolutely required. Thus, this requirement needs not be present to claim conformity.

The phrase "is prohibited from" indicates a requirement which must be strictly followed and from which no deviation is permitted if conformity with this Recommendation is to be claimed.

The phrase "can optionally" indicates an optional requirement which is permissible, without implying any sense of being recommended. This term is 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 conformity with this Recommendation.

In the body of this Recommendation and its annexes, the words *shall*, *shall not*, *should*, and *may* sometimes appear, in which case they are to be interpreted, respectively, as *is required to*, *is prohibited from*, *is recommended*, and *can optionally*. The appearance of such phrases or keywords in an appendix or in material explicitly marked as *informative* are to be interpreted as having no normative intent.

6 Interface overview

6.1 Introduction

The applications supported by TVOS application framework layer include two categories, namely, JAVA applications and WEB applications:

– JAVA application refers to a general term of applications developed with Java language;

– WEB application refers to a general term of applications developed with web technologies such as hyper text markup language (HTML), JavaScript and cascading style sheets (CSS).

The TVOS application framework layer implements packaging and adaptation of JAVA application and WEB application with the function component module. The application programming interface (API) defined in this Recommendation provides a uniform application programming interface for JAVA application and WEB application at the application layer, and also offers the reference for TVOS application developers to develop JAVA applications and WEB applications.

The interfaces defined in this part conform to the relevant requirements of [ITU-T J.1201] and [ITU-T J.1202].

6.2 TVOS JAVA application programming interface

6.2.1 Overview of the TVOS JAVA application programming interface

TVOS JAVA application programming interface provides the invocation interface in the form of Java object, and supports the application to implement relevant digital television service functions such as electronic program guide, channel list and TV program playing. JAVA application programming interface consists of 11 function units, including one-way broadcast network access unit, broadcast protocol processing unit, two-way broadband network access unit, human-machine interaction unit, audio video (AV) setting unit, media processing unit, system management unit, application engine unit, multi-screen interaction unit, digital rights management (DRM) unit and downloadable condition access system (DCAS) management unit.

6.2.2 One-way broadcast network access unit

6.2.2.1 Overview of the one-way broadcast network access unit

The one-way broadcast network access unit is used to implement one-way broadcast network access function, including parameter control (tuning frequency, modulation mode and symbol rate), and

acquisition of signal intensity, quality and other information. The unIt defines the tuning and demodulation module. Refer to Annex A for the detailed definition of the Java interface.

6.2.2.2 Tuning and demodulation module

The tuning and demodulation module defines the interface used for turning and demodulation, class and exception, including definition of tuning parameters, definition of Tuner object, and tuning and demodulation management class, and implements the tuning and demodulation function through these definitions.

See Table 1 for an overview of the tuning and demodulation module.

Table 1 – Overview of the tuning and demodulation module

Object name	Type	Description	Remarks
DeliverySystemType	Interface	Constant definition of delivery system type under digital video broadcasting (DVB) technical system.	See clause A.2.1
TuningParameters	Interface	Interface of tuning and demodulation parameters.	See clause A.2.2
TuningListener	Interface	Interface of network interface event listener, providing tuning methods to handle network interface events.	See clause A.2.3
DvbcTuningParameters	Class	Applicable to tuning and demodulation parameter class of DVB-C delivery system.	See clause A.2.4
AbsssTuningParameters	Class	Applicable to tuning and demodulation parameter class of ABS-SS delivery system.	See clause A.2.5
DtmbTuningParameters	Class	Applicable to tuning and demodulation parameter class of digital television terrestrial multimedia broadcast (DTMB) delivery system.	See clause A.2.6
TunerEvent	Class	Tuner and demodulator event class.	See clause A.2.7
TunerTuningEvent	Class	Network interface starts to tune events and inherits TunerEvent class.	See clause A.2.8
TunerTuningOverEvent	Class	End event tuning and inherit TunerEvent class.	See clause A.2.9
Tuner	Class	Tuning and demodulation control interface.	See clause A.2.10
TunerManager	Class	Tuning and demodulation manager class, used to track broadcast network interface connected to the receiving equipment; it is an entry class of tuning and demodulation unit.	See clause A.2.11
TunerException	Exception	Network interface exception	See clause A.2.12
IncorrectLocatorException	Exception	Incorrect locator format exception; inherit TunerException class.	See clause A.2.13
StreamNotFoundException	Exception	No exception is found for stream; inherit TunerException class. When the quotation of the transport stream cannot be analyzed because the transport stream	See clause A.2.14

Table 1 – Overview of the tuning and demodulation module

Object name	Type	Description	Remarks
		is not in StreamTable, such exception is thrown.	
TuningParameterNotFound Exception	Exception	The exception is thrown when getting current tuning and demodulation parameters fails or DeliverySystemType is wrong.	See clause A.2.15

6.2.3 Broadcast protocol processing unit

6.2.3.1 Overview of the broadcast protocol processing unit

The broadcast protocol processing unit is used to implement broadcast protocol processing, including Moving Picture Experts Group (MPEG) object definition module, DVB object definition module, SECTION filter module, URL packaging module, DVB locator module and broadcast protocol processing module. Refer to Annex B for the detailed definition of the Java interface.

6.2.3.2 MPEG object definition module

The MPEG object definition module defines the fundamental objects and possible system exceptions under MPEG-2 system.

The fundamental MPEG-2 objects defined in this module include:

– TransportStream class;

– ElementaryStream class;

– Service class.

MPEG-2 exceptions defined include:

– NotAuthorizedException;

– ResourceException;

– TuningException.

See Table 2 for an overview of the MPEG object definition module.

Table 2 – Overview of the MPEG objectdefinition module

Object name	Type	Description	Remarks
NotAuthorizedInterface	Interface	The broadcasting content does not authorize the report interface. It defines the failure cause constant and provides the method to seek failure causes.	See clause B.2.1
TransportStream	Class	MPEG-2 TransportStream class, it represents a MPEG-2 transport stream (TS) and provides the method to get transport stream (TS) message.	See clause B.2.2
ElementaryStream	Class	MPEG-2 ElementaryStream class, it represents an elementary stream (ES) in TS and provides the method to get elementary stream (ES) message.	See clause B.2.3

Table 2 – Overview of the MPEG objectdefinition module

Object name	Type	Description	Remarks
Service	Class	MPEG-2 Service class, it represents a MPEG-2 service in TS and provides the method to get service message.	See clause B.2.4
NotAuthorizedException	Exception	Broadcasting content unauthorized exception, it implements NotAuthorizedInterface interface, and is thrown when accessing unauthorized scrambling data.	See clause B.2.5
TuningException	Exception	Tuning and demodulation exception, it is thrown when tuning and demodulation fail.	See clause B.2.6
ResourceException	Exception	Resource exception, it is thrown when the operation cannot be implemented due to the lack of resources.	See clause B.2.7

6.2.3.3 DVB object definition module

The DVB object definition module defines elementary objects of MPEG-2 under DVB system:

– DvbTransportStream class;

– DvbElementaryStream class;

– DvbService class.

See Table 3 for an overview of the DVB object definition module.

Table 3 – Overview of the DVB object definition module

Object name	Type	Description	Remarks
DvbElementaryStream	Class	DVB elementary stream class, it represents a MPEG-2 elementary stream (ES) which conforms to DVB semantic constraints in transport stream (TS), and provides the method to get DVB ES message.	See clause B.3.1
DvbService	Class	DVB service class, it represents a MPEG-2 service which conforms to DVB semantic constraints in transport stream, and provides the method to get DVB service message.	See clause B.3.2
DvbTransportStream	Class	DVB transport stream class, it represents a MPEG-2 transport stream which conforms to DVB semantic constraints and provides the method to get DVB transport stream message.	See clause B.3.3

6.2.3.4 SECTION filter module

The SECTION filter module provides the classes and methods related to MPEG-2 section filtering.

See Table 4 for an overview of the SECTION filter module.

Table 4 – Overview of the SECTION filter module

Object name	Type	Description	Remarks
SectionFilterListener	Interface	Interface of section filter event listener, it provides section filter event handling and callback method, and is implemented by the application layer.	See clause B.4.1
Section	Class	MPEG-2 section class, it describes a section filtered from transport steam.	See clause B.4.2
SectionFilterGroup	Class	Section filter group class, it represents a MPEG-2 filter group, and can be enabled and released as a basic operation unit.	See clause B.4.3
SectionFilter	Class	Section filter class, this class is a base class of a group of section filter classes characterized by different life cycle and buffer length, and provides basic operation methods for filters.	See clause B.4.4
SimpleSectionFilter	Class	Simple section filter, it inherits SectionFilter class.	See clause B.4.5
TableSectionFilter	Class	Table section filter class, it inherits SectionFilter class.	See clause B.4.6
RingSectionFilter	Class	Ring section filter, it inherits SectionFilter class.	See clause B.4.7
SectionFilterEvent	Event	Section filter event class, base class of a group of section filter event classes.	See clause B.4.8
SectionAvailableEvent	Event	Event of section data available, it inherits SectionFilterEvent class and reports a complete section is filtered.	See clause B.4.9
VersionChangeDetectedEvent	Event	Event of section filter version change, it inherits SectionFilterEvent class.	See clause B.4.10
EndOfFilteringEvent	Event	Event of end of section filtering, it inherits SectionFilterEvent class and reports section filtering ends.	See clause B.4.11
IncompleteFilteringEvent	Event	Incomplete section filtering event, it inherits SectionFilterEvent class.	See clause B.4.12
TimeOutEvent	Event	Event of section filtering time-out, it inherits SectionFilterEvent class.	See clause B.4.13
FilterResourcesAvailableEvent	Event	Filter resource available event, it inherits SectionFilterEvent class.	See clause B.4.14
ForcedDisconnectedEvent	Event	Event of forced disconnection between section filter group and transport stream, it inherits ResourceStatusEvent class.	See clause B.4.15
SectionFilterException	Exception	Base class of section filter exception.	See clause B.4.16

Table 4 – Overview of the SECTION filter module

Object name	Type	Description	Remarks
ConnectionLostException	Exception	Connection lost exception, it inherits SectionFilterException class.	See clause B.4.17
FilteringInterruptedException	Exception	Filtering interrutted exception, it inherits SectionFilterException class.	See clause B.4.18
FilterResourceException	Exception	Filter resource exception, it inherits SectionFilterException class.	See clause B.4.19
IllegalFilterDefinitionException	Exception	Illegal filter definition exception, it inherits SectionFilterException class.	See clause B.4.20
InvalidSourceException	Exception	Invalid section data source exception, it inherits SectionFilterException class.	See clause B.4.21
NoDataAvailableException	Exception	Exception of no data available for section object, it inherits SectionFilterException class.	See clause B.4.22

6.2.3.5 URL packaging module

The URL packaging module provides the method to quote URL packaging.

See Table 5 for an overview of the URL packaging module.

Table 5 – Overview of the URL packaging module

Object name	Type	Description	Remarks
Locator	Class	Resource locator, it packages URL as a locator object.	See clause B.5.1
InvalidLocatorException	Exception	Invalid locator exception.	See clause B.5.2

6.2.3.6 DVB locator module

The DVB locator module provides the method to access DVB broadcasting services and contents.

See Table 6 for an overview of the DVB locator module.

Table 6 – Overview of the DVB locator module

Object name	Type	Description	Remarks
DvbLocator	Class	DVB locator class, it packages URL of DVB format as a locator object.	See clause B.6.1
DvbNetworkBoundLocator	Class	DVB locator class bound to network, the objects of such class uniquely identify a given entity and delivery system.	See clause B.6.2

6.2.3.7 Broadcast protocol processing module

The broadcast protocol processing module defines the classes and methods related to DVB broadcast protocol processing.

See Table 7 for an overview of the broadcast protocol processing module.

Table 7 – Overview of the broadcast protocol processing module

Object name	Type	Description	Remarks
SICommonInformation	Interface	PSI/SI common information interface, it provides the method to get PSI/SI common properties.	See clause B.7.1
SINetwork	Interface	Network information interface, it provides the method get network information. Every SINetwork object is uniquely identified by network_id.	See clause B.7.2
SIBouquet	Interface	Bouquet information interface, it provides the method to get bouquet information. Every SIBouquet object is uniquely identified by network_id and bouquet_id.	See clause B.7.3
SIService	Interface	Service information interface, it provides the method to get service information. Every SIService object is uniquely identified by network_id, original_network_id, transport_stream_id and service_id jointly.	See clause B.7.4
SITransportStream	Interface	Transport stream information interface, it provides the method to get transport stream (transport_stream) information. Every SITransportStream object is uniquely identified by network_id, original_network_id and transport_stream_id jointly.	See clause B.7.5
SIElementaryStream	Interface	Elementary stream information interface, it provides the method to get elementary stream (elementary_stream) information. Every SIElementaryStream object is uniquely identified by network_id, original_network_id, transport_stream_id, service_id and component_tag (or elementary_PID) jointly.	See clause B.7.6
SIEvent	Interface	Program event information interface, it provides the method to get event information. Every SIEvent object is uniquely identified by network_id, original_network_id, transport_stream_id, service_id and event_id jointly.	See clause B.7.7
SITime	Interface	Time information interface, it provides the method to get time information. Time information is acquired from TDT or TOD, and every SITime object is uniquely identified by network_id.	See clause B.7.8
SIDescriptor	Interface	Descriptor information interface, it provides the methods about descriptor access.	See clause B.7.9
SIRequest	Interface	PSI/SI information request interface, it describes one PSI/SI information retrieval request from the application and the application can cancel this request through this object.	See clause B.7.10

Table 7 – Overview of the broadcast protocol processing module

Object name	Type	Description	Remarks
SIRetrieveListener	Interface	SI information retrieve listener, it is implemented by the application.	See clause B.7.11
SIUpdateListener	Interface	PSI/SI table update listener, it is implemented by the application.	See clause B.7.12
SIDescriptorTag	Interface	Interface of descriptor tag constant definition.	See clause B.7.13
SIRunningStatus	Interface	Running status constant definition interface of service or program.	See clause B.7.14
SIServiceType	Interface	Interface of service type constant definition.	See clause B.7.15
SIStreamType	Interface	Interface of stream type constant definition.	See clause B.7.16
SIDatabase	Class	PSI/SI information database, it provides management and operation methods for PSI/SI information database in DVB mode, and it is an entry class for the application to get PSI/SI information.	See clause B.7.17
SIRequestFailureType	Class	Cause for PSI/SI information request failure.	See clause B.7.18
SIRetrieveEvent	Event	PSI/SI information request event, it is a base class of a group of events related to PSI/SI information request defined in this packet. One PSI/SI information request will only generate one event like this.	See clause B.7.19
SISuccessRetrieveEvent	Event	PSI/SI information request success event, it inherits SIRetrieveEvent class.	See clause B.7.20
SIFailureRetrieveEvent	Event	PSI/SI information request failure event, it inherits SIRetrieveEvent class.	See clause B.7.21
SIUpdateEvent	Event	PSI/SI table update event.	See clause B.7.22
InvalidPeriodException	Exception	When the specified period is invalid, this exception will be thrown.	See clause B.7.23

6.2.4 Two-way broadband network access unit

6.2.4.1 Overview of the two-way broadband network access unit

The two-way broadband network access unit is used to implement the functions related to two-way broadband network access control, including two-way network access management and data operation, etc., and It defines the Ethernet management module and WiFi management module. Refer to Annex C for detailed definition of the Java interface.

6.2.4.2 Ethernet management module

The Ethernet management module provides dynamic host configuration protocol (DHCP) configuration as well as the class and method for Ethernet management and control.

See Table 8 for an overview of the Ethernet management module.

Table 8 – Overview of the Ethernet management module

Object name	Type	Description	Remarks
Listener	Interface	Ethernet status change listener.	See clause C.2.1
DhcpInfo	Class	Provide the method to get DHCP configuration information.	See clause C.2.2
EthernetManager	Class	Provide Ethernet management method.	See clause C.2.3

6.2.4.3 WiFi management module

The WiFi management module provides the class and method about WiFi network interface control.

See Table 9 for an overview of the WiFi management module.

Table 9 – Overview of the WiFi management module

Object name	Type	Description	Remarks
ActionListener	Interface	WiFi status change action listener, it is implemented by the application layer.	See clause C.3.1
WifiInfo	Class	WiFi connection information class, it provides the method to get connection information.	See clause C.3.2
WifiManager	Class	WiFi manager, it provides the management function of WiFi wireless network. a) Find, scan and manage current wireless network access points (AP) available; b) Manage current network access linkage, such as connection establishment, disconnection, connection forbidden and connection deleting.	See clause C.3.3
ScanResult	Class	WiFi scanning result, it describes information of a connection access point found by WiFi scanning.	See clause C.3.4

6.2.5 Human-machine interaction unit

6.2.5.1 Overview of the human-machine interaction unit

The human-machine interaction unit is used to implement the functions of device input control and front panel display control. The input packages user command sent by input devices such as remote controller, mouse, keyboard and front panel key as key message, and the output feeds information back through front panel or display screen. Meanwhile, it also supports speech input control and implementation. This unIt defines the human-machine interaction module. Refer to Annex D for the detailed definition of the Java interface.

6.2.5.2 Human-machine interaction module

The human-machine interaction module provides the class and method related to human-machine interaction, including user input and front panel output.

See Table 10 for an overview of the human-machine interaction module.

Table 10 – Overview of the human-machine interaction module

Object name	Type	Description	Remarks
UserInput	Interface	Definition of user input message.	See clause D.2.1

Table 10 – Overview of the human-machine interaction module

Object name	Type	Description	Remarks
NgbKeyListener	Interface	Key event listener interface, it is implemented by the application which needs to monitor KeyEvent.	See clause D.2.2
NgbMouseListener	Interface	Mouse event listener interface, it is implemented by the application which needs to monitor MouseEvent.	See clause D.2.3
NgbVoiceListener	Interface	Voice event listener interface, it is implemented by the application which needs to monitor voice recognition.	See clause D.2.4
FrontPanel	Class	Display output control for front panel information, including LED indicator light and LED digital tube display control.	See clause D.2.5
NgbInputManager	Class	Input control manager, used to receive the listener of remote controller, key and mouse and control input.	See clause D.2.6
NgbVoiceManager	Class	Voice-related control manager, used to control and implement voice recognition function.	See clause D.2.7
NgbInputEvent	Event	Input event class, it is a base class of input events.	See clause D.2.8
KeyEvent	Event	Key event class, it inherits NgbInputEvent class.	See clause D.2.9
MouseEvent	Event	Mouse event class, it inherits NgbInputEvent class.	See clause D.2.10

6.2.6 AV setting unit

6.2.6.1 Overview of the AV setting unit

The AV setting unit is used to implement the functions of getting and setting audio video parameters, including audio output port status, track type, global volume and volume status, etc., as well as video output port status, window matching module, brightness, contrast ratio, saturation, system and transparency, etc. It defines the AV setting module. Refer to Annex E for the detailed definition of the Java interface.

6.2.6.2 AV setting module

The AV setting module provides the class and method related to the setting of audio video output parameters.

See Table 11 for an overview of the AV setting module.

Table 11 – Overview of the AV setting module

Object name	Type	Description	Remarks
AudioSetting	Class	Set various parameters of audio input/output unit.	See clause E.2.1
VideoSetting	Class	Set various parameters of video input/output unit.	See clause E.2.2

6.2.7 Media processing unit

6.2.7.1 Overview of the media processing unit

The media processing unit is used to implement media player function, including playing, control and language selection, etc. It defines the media processing module. Refer to Annex F for the detailed definition of the Java interface.

6.2.7.2 Media processing module

The media processing module provides the interface class of all media functions as well as relevant audio and video information classes. The media processing module is used to support DVB, VOD, IP broadcasting, IP broadcasting on demand and local broadcasting functions.

The fundamental classes defined in the media processing module include:

– MediaPlayer class;

– TrackInfo class for audio, video and subtitle;

– MediaFormat class.

See Table 12 for an overview of the media processing module.

Table 12 – Overview of the media processing module

Object name	Type	Description	Remarks
MediaPlayer	Class	Provide all media function interfaces. Support DVB, VOD, IP broadcasting, IP broadcasting on demand and local broadcasting.	See clause F.2.1
TrackInfo	Class	Describe track information such as audio, video and subtitle.	See clause F.2.2
MediaFormat	Class	Use HashMap to store audio video stream and other information.	See clause F.2.3

6.2.8 System management unit

6.2.8.1 Overview of the system management unit

The system management unit is used to implement peripheral device management, over the air (OTA) upgrade management, storage management and other system-related functions. It defines the system management module, OTA upgrade module and storage management module. Refer to Annex G for the detailed definition of the Java interface.

6.2.8.2 System management module

The system management module provides the class and method for configuration parameter access, software and hardware configuration information retrieval, peripheral management and system operation, etc.

See Table 13 for an overview of the system management module.

Table 13 – Overview of the system management module

Object name	Type	Description	Remarks
PeripheralType	Interface	Constant definition interface of peripheral type supported by TVOS.	See clause G.2.1

Table 13 – Overview of the system management module

Object name	Type	Description	Remarks
Peripheral	Interface	Peripheral description interface, it provides the methods to get peripheral name, status, type and ID, etc.	See clause G.2.2
PeripheralListener	Interface	Peripheral listener, it is implemented by the application.	See clause G.2.3
PeripheralManager	Class	Peripheral manager, it is an entry class of peripheral management module.	See clause G.2.4
DataConfig	Class	Configuration data access class, it provides the method to access configuration data saved in the receiving terminal non-volatile memory (NVM).	See clause G.2.5
HardwareInfo	Class	Hardware information description class, it provides the method to get hardware parameter information at the receiving terminal.	See clause G.2.6
SoftwareInfo	Class	Software information description class, it provides the method to get software parameter information at the receiving terminal.	See clause G.2.7
SysTool	Class	System tool class, it provides the operation method for system standby, hibernation and restart, etc.	See clause G.2.8
PeripheralEvent	Event	Peripheral message event.	See clause G.2.9

6.2.8.3 OTA upgrade module

The OTA upgrade module provides the class and method for OTA software upgrade detection and handling.

See Table 14 for an overview of the OTA upgrade module.

Table 14 – Overview of the OTA upgrade module

Object name	Type	Description	Remarks
OTAEventListener	Interface	OTA event listener, it is implemented by the application.	See clause G.3.1
OTAManager	Class	OTA manager, it is an entry class of OTA function module.	See clause G.3.2
OTAEvent	Event	OTA event.	See clause G.3.3

6.2.8.4 Storage management module

The storage management module provides the class and method for storage device management and partitioned access of storage device.

See Table 15 for an overview of the storage management module.

Table 15 – Overview of the storage management module

Object name	Type	Description	Remarks
Storage	Interface	It describes storage device information, such as name, size, idle status and partition.	See clause G.4.1
StorageEventListener	Interface	Storage event listener, it is implemented by the application.	See clause G.4.2
StoragePartition	Interface	It describes partition information of storage device, such as name, size, idle status, access path and partition type.	See clause G.4.3
StorageManager	Class	It provides the method to manage the storage device and partition the storage device.	See clause G.4.4
StorageEvent	Event	Storage events related to the storage device.	See clause G.4.5

6.2.9 Application engine unit

6.2.9.1 Overview of the application engine unit

The application engine unit is used to implement the functions of channel search, electronic program guide acquisition and information search. It defines the channel search module, electronic program guide module and information search module. Refer to Annex H for the detailed definition of the Java interface.

6.2.9.2 Channel search module

The channel search module provides the class and method related to channel search.

See Table 16 for an overview of the channel search module.

Table 16 – Overview of the channel search module

Object name	Type	Description	Remarks
ChannelScanListener	Interface	Channel scanning listener, it is implemented by the application.	See clause H.2.1
ChannelScanEngine	Class	Search engine. It is an entry class of channel scanning function unit.	See clause H.2.2
ChannelScanEvent	Event	Channel scanning event, base class	See clause H.2.3
ChannelScanFailureEvent	Event	Channel scanning failure event, it inherits ChannelScanEvent class.	See clause H.2.4
ChannelScanFinishEvent	Event	Channel scanning finish event, it inherits ChannelScanEvent class.	See clause H.2.5
ChannelScanNITSuccessEvent	Event	Channel scanning success and NIT analysis event, it inherits ChannelScanEvent class.	See clause H.2.6
ChannelScanSuccessEvent	Event	Channel scanning success event, it inherits ChannelScanEvent class.	See clause H.2.7

6.2.9.3 Electronic program guide module

Electronic program guide (EPG) provides the method for terminal users to browse broadcasting service information, such as service name, program starting and ending time, and content outline, so that terminal users can fast retrieve and access the services. This part follows the analysis and presentation of EPG program information.

The electronic program guide module provides the class and method for EPG information acquisition. Cache mechanism can be adopted for EPG information acquisition, or EPG information can be temporarily downloaded, if necessary. If the cache mechanism is applied, EPG information should be monitored in real time to ensure the application can extract the latest EPG information.

See Table 17 for an overview of the electronic program guide module.

Table 17 – Overview of the electronic program guide module

Object name	Type	Description	Remarks
ProgramEvent	Interface	It describes the information of a program event.	See clause H.3.1
ProgramEventFilter	Interface	It defines the filter interface used by the application to query program event information from EPG module, and it is implemented by the application layer.	See clause H.3.2
ProgramService	Interface	It describes service information of a program, and it is the packaging of a group of program event information belonging to the same service.	See clause H.3.3
ProgramServiceFilter	Interface	It defines the filter interface used by the application to query program service information from EPG module, and it is implemented by the application layer.	See clause H.3.4
EPGUpdateListener	Interface	PEG information update listener, it is implemented by the application.	See clause H.3.5
EPGManager	Class	EPG manager, it is an entry class of EPG information retrieval.	See clause H.3.6
EPGUpdateEvent	Event	EPG information update event.	See clause H.3.7

6.2.9.4 Information search module

The information search module provides the class and method related to global search and auto complete search. The definitions of "global search" and "auto complete search" are as below:

"Global search" – seek SI and PVR contents according to the search conditions set by users, and return meaningful results to improve user experience.

"Auto complete search" – give matched character strings in the current data source according to user's input to shorten user's time to input query keywords and lower user's difficulty in inputting query keywords.

The information search module includes the following components:

– SearchManager – entry class of information search module;

– GlobalSearchSession – session associated with global search process;

– AutoCompleteSearchSession – session associated with auto complete search process.

See Table 18 for an overview of the information search module.

Table 18 – Overview of the information search module

Object name	Type	Description	Remarks
AutoCompleteSearchListener	Interface	Auto complete search listener, it is implemented by the application.	See clause H.4.1
AutoCompleteSearchResultItem	Interface	It describes the results of auto complete search and provides the	See clause H.4.2

Table 18 – Overview of the information search module

Object name	Type	Description	Remarks
		method to get the results of auto complete search.	
AutoCompleteSearchResultList	Interface	It describes the list object of auto complete search results and provides the method to access the result items of auto complete search.	See clause H.4.3
AutoCompleteSearchSession	Interface	It describes a session of auto complete search and provides the method to control the session of auto complete search.	See clause H.4.4
GlobalSearchListener	Interface	Global search listener, it is implemented by the application.	See clause H.4.5
GlobalSearchResultItem	Interface	It describes an object of global search result and provides the method to access the search results.	See clause H.4.6
GlobalSearchResultList	Interface	It describes the list object of global search results and provides the method to access the result items of global search.	See clause H.4.7
GlobalSearchSession	Interface	It describes a global search session and provides the method to control the global search session.	See clause H.4.8
RetrieveDirection	Interface	It defines the constant of search result seeking direction.	See clause H.4.9
SearchContentType	Interface	Content type constant definition.	See clause H.4.10
SearchCriteriaFlags	Interface	Filtering criteria flag definition.	See clause H.4.11
SearchFields	Interface	It provides the interface for setting search fields.	See clause H.4.12
SearchHistoryItem	Interface	It describes a search history and provides the method to get all kinds of information of search history.	See clause H.4.13
SearchHistoryList	Interface	It describes the search history list and provides the function of traversing the search history list.	See clause H.4.14
SearchStatus	Interface	It defines the search status constant.	See clause H.4.15
SourceType	Interface	Definition of search data source constant.	See clause H.4.16
AutoCompleteSearchFilter	Class	It describes a filter of auto complete search and provides the method to set and get auto complete search filtering conditions.	See clause H.4.17
GlobalSearchFilter	Class	It describes a filter of global search and provides the method to set and get global search filtering conditions.	See clause H.4.18

Table 18 – Overview of the information search module

Object name	Type	Description	Remarks
SearchManager	Class	Search manager of global search and auto complete search, it is an entry class of search function module.	See clause H.4.19
SortCriteria	Class	It defines sorting constant and sorting method.	See clause H.4.20

6.2.10 Multi-screen interaction unit

6.2.10.1 Overview of the multi-screen interaction unit

The multi-screen interaction unit is used to implement multi-screen interaction local area network (LAN) interface, and it defines the multi-screen interaction module. Refer to Annex I for the detailed definition of the Java interface.

6.2.10.2 Multi-screen interaction module

The multi-screen interaction module can implement the functions of discovering, connecting and controlling server equipme6nt in LAN by the application client.

See Table 19 for an overview of the multi-screen interaction module.

Table 19 – Overview of the multi-screen interaction module

Object name	Type	Description	Remarks
IMultiScreenService	Interface	Function interface used to support multi-screen-interactive component in LAN environment.	See clause I.2.1
IMultiScreenCallBack	Interface	Remote access interface provided by multi-screen-interactive component.	See clause I.2.2

6.2.11 DRM management unit

6.2.11.1 Overview of the DRM management unit

The DRM management unit is used to implement DRM management function, and It defines the DRM management module. Refer to Annex J for the detailed definition of the Java interface.

6.2.11.2 DRM management module

See Table 20 for an overview of the DRM management module.

Table 20 – Overview of the DRM management module

Object name	Type	Description	Remarks
CommonDrmManager	Class	DRM module class.	See clause J.2.1
CommonDrmTeeRetVal	Class	Return type class of CommonDrm_SendCommandToTEE method of CommonDrmManager class.	See clause J.2.2

6.2.12 DCAS management unit

6.2.12.1 Overview of the DCAS management unit

The DCAS management unit is used to implement registration, filter setting and trusted application (Tapp) communication functions related to DCAS application, and It defines the conditional access system (CAS) descrambler module, CAS control module, CAS message module, and CAS listener module. Refer to Annex K for the detailed definition of the Java interface based on [b-ITU-T J.1033].

6.2.12.2 CAS descrambler module

The CAS descrambler module provides application programming interface for DCAS terminal software platform to descramble.

See Table 21 for an overview of the CAS descrambler module.

Table 21 – Overview of the CAS descrambler module

Object name	Type	Description	Remarks
CASModule	Interface	CASModule object used to express the request of descrambling a group of elementary streams.	See clause K.2.1
CASDataUtils	Interface	Used to get and set Certification Authority (CA) information, read and write DCAS data.	See clause K.2.2
CADescriptor	Interface	It provides CA descriptor information, and may provide CA descriptor in the PMT with given service. Besides, CA descriptor may appear in CAT.	See clause K.2.3
CAServiceComponentInfo	Interface	It is used to extract the information of specific CA service component, such as entitlement control message (ECM) packet identifier (PID) and DescramblerContext used to load control word.	See clause K.2.4
CASPacketListener	Interface	DCAS application receives out-of-band CAS Packets through this interface (such as EMMs).	See clause K.2.5
CASSession	Interface	Provide the information about CAS session.	See clause K.2.6
CAStatus	Interface	DCAS application sends CAStatus whenever descrambler status in DescramblerContext changes, and this status is used to prompt whether descrambling succeeds. If any descrambling component fails, this status must report descrambling request failure of the whole service. When the terminal software platform receives a new CATStatus, it should inform other applications through CAS event described here.	See clause K.2.7
CATListener	Interface	DCAS application needs to implement the interface and uses CA descriptor in CAT to filter in-band entitlement management message (EMM).	See clause K.2.8
CATNotifier	Interface	DCAS application uses this method to register a listener used to get CAT update notice.	See clause K.2.9
CASModuleManager	Class	Used to register all CASModule implemented by DCAS application.	See clause K.2.10

Table 21 – Overview of the CAS descrambler module

Object name	Type	Description	Remarks
CASPermission	Class	Any DCAS application cannot access CASModuleManager until it gets CASPersmission. This mechanism is used to ensure only DCAS application authorized by the network operator can use DCAS API.	See clause K.2.11

6.2.12.3 CAS control module

The CAS control module provides application programming interface for DCAS terminal software platform to control CAS.

See Table 22 for an overview of the CAS control module.

Table 22 – Overview of the CAS control module

Object name	Type	Description	Remarks
DescramblerContext	Interface	It is a component used to control the descrambling function of terminal security chip. It can instantiate multiple DescramblerContext to descramble multiple code streams with different keys.	See clause K.3.1
ChipController	Interface	It is a component used to control the execution of terminal security chip.	See clause K.3.2
Key	Class	An elementary cipher key.	See clause K.3.3
CWKey	Class	Descramble cipher or control word.	See clause K.3.4
CASTEEManager	Class	Function interface with trusted application (TA) communication in trusted execution environment (TEE).	See clause K.3.5

6.2.12.4 CAS message module

The CAS message module provides the function interface for DCAS message monitoring.

See Table 23 for an overview of the CAS message module.

Table 23 – Overview of the CAS message module

Object name	Type	Description	Remarks
CASEventListener	Interface	It is used to implement the application of CAS event receiving.	See clause K.4.1
CASAppInfo	Interface	Provide DCAS application information.	See clause K.4.2
CASEventInfo	Interface	Provide CASEvent information.	See clause K.4.3
CASEventManager	Class	The application uses CASEventManager to register listener to get CAS event.	See clause K.4.4

6.2.12.5 CAS listener module

The CAS listener module provides application programming interface for DCAS detachable security device to monitor CAS.

See Table 24 for an overview of the CAS listener module.

Table 24 – Overview of the CAS listener module

Object name	Type	Description	Remarks
DetachableSecurityDevice	Interface	It is used by the application to register the listener of detachable security device so as to get device plug/unplug status.	See clause K.5.1
DetachableSecurityDevice Listener	Interface	Detachable security device listener, it is used by the application to monitor device plug/unplug status.	See clause K.5.2

6.3 TVOS WEB application programming interface

6.3.1 Overview of the TVOS WEB application programming interface

TVOS WEB application programming interface implements JavaScript (JS) packaging for each function component module interface in the component layer, provides invocation interface for WEB application in the form of JS object and supports the application to implement relevant digital television service functions such as electronic program guide, channel list and TV program playing. WEB application programming interface consists of 14 function units, including one-way broadcast network access unit, broadcast protocol processing unit, two-way broadband network access unit, human-machine interaction unit, AV setting unit, media processing unit, application management unit, system management unit, message management unit, application engine unit, broadcast message service management unit, multi-screen interaction unit, DRM management unit and DCAS management unit.

The definition of TVOS WEB application programming interface conforms to the requirements of ECMA-262.

6.3.2 One-way broadcast network access unit

6.3.2.1 Overview of the one-way broadcast network access unit

The one-way broadcast network access unit is used to implement one-way broadcast network access function under digital television, including parameter control (tuning frequency, modulation mode and symbol rate), and acquisition of signal intensity, quality and other information. It defines the tuning and demodulation module. Refer to Annex L for the detailed definition of the JS interface.

6.3.2.2 Tuning and demodulation module

See Table 25 for an overview of the tuning and demodulation module.

Table 25 – Overview of the tuning and demodulation module

Object	Description	Remarks
DvbcTuningParameters	DVB-C tuning and demodulation parameter object.	See clause L.2.3
AbsssTuningParameters	ABS-SS tuning and demodulation parameter object.	See clause L.2.4
DtmbTuningParameters	DTMB tuning and demodulation parameter object.	See clause L.2.5
DvbTune	Channel tuning and demodulation object.	See clause L.2.6
DvbTunerInfo	Tunerid and Tuner type matching object.	See clause L.2.7
DvbScan	Channel scanning object.	See clause L.2.8

6.3.3 Broadcast protocol processing unit

6.3.3.1 Overview of the broadcast protocol processing unit

The broadcast protocol processing unit is used to implement broadcast protocol processing, and It defines the DVB protocol processing module. Refer to Annex M for the detailed definition of the JS interface.

6.3.3.2 DVB protocol processing module

See Table 26 for an overview of the DVB protocol processing module.

Table 26 – Overview of the DVB protocol processing module

Object	Description	Remarks
DvbBroadcast	DVB broadcast channel object.	See clause M.2.3
DvbNetwork	DVB network object.	See clause M.2.4
DvbBouquet	DVB bouquet object.	See clause M.2.5
DvbTS	DVB transport stream object.	See clause M.2.6
DvbService	DVB service object.	See clause M.2.7
DvbVideoES	DVB video ES object.	See clause M.2.8
DvbAudioES	DVB audio ES object.	See clause M.2.9
DvbOtherES	Other ES objects except audio and video.	See clause M.2.10
DvbEvent	DVB program event object.	See clause M.2.11

6.3.4 Two-way broadband network access unit

6.3.4.1 Overview of the two-way broadband network access unit

The two-way broadband network access unit is used to implement the functions related to two-way broadband network access control, including two-way network access management and data operation, etc., and it defines the two-way broadband network setting module. Refer to Annex N for the detailed definition of the JS interface.

6.3.4.2 Two-way broadband network setting module

See Table 27 for an overview of the two-way broadband network setting module.

Table 27 – Overview of the two-way broadband network setting module

Object	Description	Remarks
Broadband	Two-way broadband network object.	See clause N.2.2
Ethernet	Ethernet card object.	See clause N.2.3
AP	Wireless access point object.	See clause N.2.4
IP	IP object.	See clause N.2.5
Proxy	Network proxy object.	See clause N.2.6

6.3.5 Human-machine interaction unit

6.3.5.1 Overview of the human-machine interaction unit

The human-machine interaction unit is used to implement the functions of device input control and front panel display control. The input packages user command sent by input devices such as remote controller, mouse, keyboard and front panel key as key message, and the output feeds information

back through front panel or display screen. It defines the user input unit module and front panel output module. Refer to Annex O for the detailed definition of the JS interface.

6.3.5.2 User input unit module

See Table 28 for an overview of the user input unit module.

Table 28 – Overview of the user input unit module

Object	Description	Remarks
event	User input message management object.	See clause O.2.2

6.3.5.3 Front panel output module

See Table 29 for an overview of the front panel output module.

Table 29 – Overview of the front panel output module

Object	Description	Remarks
FrontPanel	Front panel object.	See clause O.3.1

6.3.6 AV setting unit

6.3.6.1 Overview of the AV setting unit

The AV setting unit is used to implement the functions of getting and setting audio video parameters, including audio output port status, track type, global volume and volume status, etc., as well as video output port status, window matching module, brightness, contrast ratio, saturation, system and transparency, etc. It defines the audio video parameter setting module. Refer to Annex P for the detailed definition of the JS interface.

6.3.6.2 Audio video parameter setting module

See Table 30 for an overview of the audio video parameter setting module.

Table 30 – Overview of the audio video parameter setting module

Object	Description	Remarks
AudioSetting	Audio parameter setting object.	See clause P.2.1
VideoSetting	Video parameter setting object.	See clause P.2.2

6.3.7 Media processing unit

6.3.7.1 Overview of the media processing unit

The media processing unit is used to implement media player function, including playing, control, event handling and exception handling, etc. It defines the media processing module. Refer to Annex Q for the detailed definition of the JS interface.

6.3.7.2 Media processing module

See Table 31 for an overview of the media processing module.

Table 31 – Overview of the media processing module

Object	Description	Remarks
MediaPlayer	Media player object.	See clause Q.2.2

6.3.8 Application management unit

6.3.8.1 Overview of the application management unit

The application management unit is used to implement WEB application management function, including application query, application installation, application unloading, and application update, etc., and it defines the application management module. Refer to Annex R for the detailed definition of the JS interface.

6.3.8.2 Application management module

See Table 32 for an overview of the application management module.

Table 32 – Overview of the application management module

Object	Description	Remarks
widget	Application management object.	See clause R.2.2

6.3.9 System management unit

6.3.9.1 Overview of the system management unit

The system management unit is used to implement peripheral device management, OTA upgrade management, storage management and other system-related functions. It defines the data management module, external storage device management module, file management module, multimedia file management module, OTA software upgrade module, system tool module and software & hardware information module. Refer to Annex S for the detailed definition of the JS interface.

6.3.9.2 Data management module

See Table 33 for an overview of the data management module.

Table 33 – Overview of the data management module

Object	Description	Remarks
DataConfig	Configuration data management object.	See clause S.2.2

6.3.9.3 External storage device management module

See Table 34 for an overview of the external storage device management module.

Table 34 – Overview of the external storage device management module

Object	Description	Remarks
StorageDeviceManager	External storage device manager object.	See clause S.3.2
StorageDevice	External storage device object.	See clause S.3.3
StoragePartition	External storage partition object.	See clause S.3.4

6.3.9.4 File management module

See Table 35 for an overview of the file management module.

Table 35 – Overview of the file management module

Object	Description	Remarks
FileManager	Directory and file management object.	See clause S.4.2
Directory	Directory object.	See clause S.4.3
File	File object	See clause S.4.4

6.3.9.5 Multimedia file module

See Table 36 for an overview of the multimedia file module.

Table 36 – Overview of the multimedia file module

Object	Description	Remarks
AudioFile	Audio file object.	See clause S.5.1
VideoFile	Video file object.	See clause S.5.2
ImageFile	Image file object.	See clause S.5.3

6.3.9.6 OTA software upgrade module

See Table 37 for an overview of the OTA software upgrade module.

Table 37 – Overview of the OTA software upgrade module

Object	Description	Remarks
Upgrade	OTA upgrade control object.	See clause S.6.2

6.3.9.7 System tool module

See Table 38 for an overview of the system tool module.

Table 38 – Overview of the system tool module

Object	Description	Remarks
Utility	Tool object used to complete event message acquisition and character string printing, etc.	See clause S.7.1
GlobalVarManager	Global variable manager object.	See clause S.7.2
Rectangle	Rectangular window object.	See clause S.7.3
SysTool	System tool object.	See clause S.7.4

6.3.9.8 Software and hardware information module

See Table 39 for an overview of the software and hardware information module.

Table 39 – Overview of the software and hardware information module

Object	Description	Remarks
HardwareInfo	Hardware parameter information object of receiving terminal.	See clause S.8.1
SoftwareInfo	Software parameter information object of receiving terminal.	See clause S.8.2

6.3.10 Message management unit

6.3.10.1 Overview of the message management unit

The message management unit is used to implement the functions of receiving and managing messages, including key message and system message. This unIt defines the message management module. Refer to Annex T for the detailed definition of the JS interface.

6.3.10.2 Message management module

See Table 40 for an overview of the message management module.

Table 40 – Overview of the message management module

Object	Description	Remarks
event	Message object.	See clause T.2.1

6.3.11 Application engine unit

6.3.11.1 Overview of the application engine unit

The application engine unit is used to implement the functions of channel management, electronic program guide acquisition, order reminder management and information search. It defines the electronic program guide module, channel management module, order reminder module and information search module. Refer to Annex U for the detailed definition of the JS interface.

6.3.11.2 Channel management module

See Table 41 for an overview of the channel management module.

Table 41 – Overview of the channel management module

Object	Description	Remarks
ChannelManager	Channel manager object.	See clause U.2.2
Channel	Channel object.	See clause U.2.3

6.3.11.3 Electronic program guide module

See Table 42 for an overview of the electronic program guide module.

Table 42 – Overview of the electronic program guide module

Object	Description	Remarks
EPGManager	EPG manager object.	See clause U.3.2
ProgramEvent	Program event object.	See clause U.3.3
ReferenceEvent	Reference program event object.	See clause U.3.4
TimeShiftEvent	Time shift program event object.	See clause U.3.5

6.3.11.4 Order reminder module

See Table 43 for an overview of the order reminder module.

Table 43 – Overview of the order reminder module

Object	Description	Remarks
OrderManager	Order manager object.	See clause U.4.2
Order	Order object.	See clause U.4.3

6.3.11.5 Information search module

See Table 44 for an overview of the information search module.

Table 44 – Overview of the information search module

Object	Description	Remarks
SearchManager	Search manager object.	See clause U.5.2
GlobalSearchSession	Global search session object.	See clause U.5.3
AutoCompleteSearchSession	Auto complete search session object.	See clause U.5.4
GlobalSearchFilter	Global search filter object.	See clause U.5.5
AutoCompleteSearchFilter	Auto complete search filter object.	See clause U.5.6
SortCritiria	Sorting mechanism object.	See clause U.5.7
GlobalSearchResultItem	Search result item object.	See clause U.5.8
SearchHistoryItem	Search history item object.	See clause U.5.9

6.3.12 Broadcast message service management unit

6.3.12.1 Overview of the broadcast message service management unit

The broadcast message service management unit is used to detect, receive and process broadcast message services, and supports the services of emergency broadcasting, message service, advertisement and on-screen display (OSD) text update, etc. Refer to Annex V for the detailed definition of the JS interface.

6.3.12.2 Broadcast message service management unit

The broadcast message service management module implements the functions of monitoring BAT table and NIT table, and filtering Section data of broadcast message service. Meanwhile, it gets CA user ID corresponding to the terminal through DCAS management unit interface. On this basis, it implements accurate reception of terminal emergency broadcasting and advertisement, etc.

See Table 45 for an overview of the broadcast message service management module.

Table 45 – Overview of the broadcast message service management module

Object	Description	Remarks
DthManager	Broadcast message service manager object.	See clause V.2.2
Ad	Advertisement object.	See clause V.2.3
AdService	Advertisement service object.	See clause V.2.4

6.3.13 Multi-screen interaction unit

6.3.13.1 Overview of the multi-screen interaction unit

The multi-screen interaction unit is used to implement multi-screen interaction LAN interface, and it defines the multi-screen interaction module. Refer to Annex W for the detailed definition of the JS interface.

6.3.13.2 Multi-screen interaction module

See Table 46 for an overview of the multi-screen interaction module.

Table 46 – Overview of the multi-screen interaction module

Object	Description	Remarks
MultiScreen	Multi-screen interaction control object.	See clause W.2.1

6.3.14 DRM management unit

6.3.14.1 Overview of the DRM management unit

The DRM management unit is used to implement DRM management function, and it defines the DRM management module. Refer to Annex X for the detailed definition of the JS interface.

6.3.14.2 DRM management module

See Table 47 for an overview of the DRM management module.

Table 47 – Overview of the DRM management module

Object	Description	Remarks
CommonDrmManager	DRM manager object.	See clause X.2.1

6.3.15 DCAS management unit

6.3.15.1 Overview of the DCAS management unit

The DCAS management unit is used to implement registration, filter setting and TApp communication functions related to DCAS application, and it defines the EPG_DCAS module and DCAS_APP module. Refer to Annex Y for the detailed definition of the JS interface based on [b-ITU-T J.1033].

6.3.15.2 EPG_DCAS module

See Table 48 for an overview of the EPG_DCAS module.

Table 48 – Overview of the EPG_DCAS module

Object	Description	Remarks
EPG_DCAS	DCAS message return object.	See clause Y.2.1

6.3.15.3 DCAS_APP module

See Table 49 for an overview of the DCAS_APP module.

Table 49 – Overview of the DCAS_APP module

Object	Description	Remarks
JSDCAS.CASDescriptor	CAS descriptor object.	See clause Y.3.2
JSDCAS.CASEcmEvent	Ecm message object.	See clause Y.3.3
JSDCAS.CASEmmEvent	Emm message object.	See clause Y.3.4
JSDCAS.CASFilter	CAS filter object.	See clause Y.3.5
JSDCAS.CASM	CAS global object.	See clause Y.3.6

Table 49 – Overview of the DCAS_APP module

Object	Description	Remarks
JSDCAS.CASModule	CAS module object.	See clause Y.3.7
JSDCAS.CASModuleManager	CAS module manager object.	See clause Y.3.8
JSDCAS.CASPacketEvent	CAS data packet object.	See clause Y.3.9
JSDCAS.CASSession	CAS descrambler request object.	See clause Y.3.10
JSDCAS.CASStatus	CAS status message object.	See clause Y.3.11
JSDCAS.TeeController	CAS and TEE communication controller object.	See clause Y.3.12
JSDCAS.TeeRetVal	CAS and TEE communication return value object.	See clause Y.3.13

7 Invocation mechanism

The TVOS application framework layer consists of a JAVA application framework and a WEB application framework which respectively form application programming interfaces for JAVA application and WEB application at the application layer to be invoked so that these applications can complete their respective functions.

The invocation model of the TVOS application programming interface is shown in Figure 1.

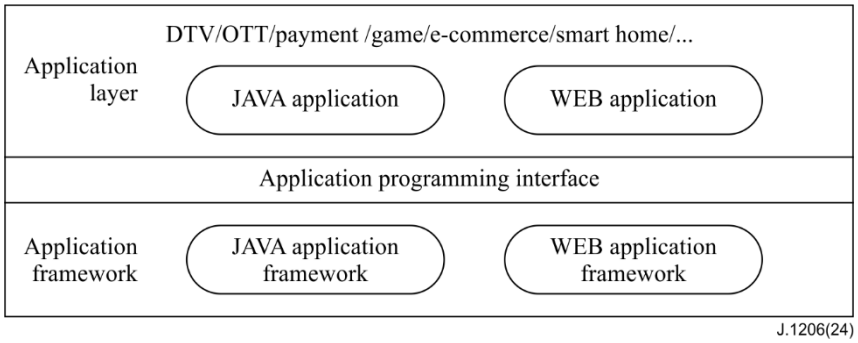


Figure 1 – Invocation model of TVOS application programming interface

JAVA application uses TVOS JAVA application programming interface to implement corresponding functions through introducing Jar packet containing TVOS JAVA application programming interface in the development process.

WEB application implements corresponding functions through introducing all kinds of built-in JavaScript objects provided by TVOS WEB application programming interface in the development process.

Annex A

JAVA-unidirectional broadcast network access unit

(This annex forms an integral part of this Recommendation.)

A.1 Overview

This annex defines the JAVA interface of the unidirectional broadcast network access unit, which is mainly the tuning and demodulation module, realizing the function of tuning and demodulation.

A.2 Tuning and demodulation module

The tuning and demodulation module defines interfaces, classes and exceptions related to tuning and demodulation.

See Table A.1 for the summary of tuning and demodulation module.

Table A.1 – Summary of the tuning and demodulation module

Interface	
DeliverySystemType	Definitions of constants of the delivery system type under the DVB technology system.
TuningParameters	Tuning and demodulation parameter interface.
TuningListener	The network interface event listener interface, providing a callback method for handling network interface related events, which is implemented by the application layer.
Classes	
DvbcTuningParameters	A tuning and demodulation parameter class applicable to the DVB-C delivery system.
AbsssTuningParameters	A tuning and demodulation parameter class applicable to the ABS-SS delivery system.
DtmbTuningParameters	A tuning and demodulation parameter class applicable to the DTMB delivery system.
TunerEvent	A class of tuner and demodulator events, which is a base class of network interface related events.
TunerTuningEvent	An event that the network interface starts tuning, inheriting the TunerEvent class.
TunerTuningOverEvent	An event that the tuning ends, inheriting the TunerEvent class.
Tuner	Tuning and demodulation control interface.
TunerManager	The tuning and demodulation manager class, which is used to track the broadcast network interface connected to the receiving device. It is the entry class of the tuning and demodulation unit.
Exceptions	
TunerException	Network interface exception, a base class of other network exceptions.
IncorrectLocatorException	An exception where the format of the locator is incorrect. It inherits the TunerException class.
StreamNotFoundException	No exception is found in the stream. It inherits the TunerException class. This exception is thrown when reference to the transport stream cannot be resolved.

Table A.1 – Summary of the tuning and demodulation module

TuningParameterNotFoundException	This exception is thrown when failed to obtain the current tuning and demodulation parameters or when the DeliverySystemType is wrong.

A.2.1 Interface org.ngb.broadcast.dvb.tuner.DeliverySystemType

Prototype: public interface org.ngb.broadcast.dvb.tuner.DeliverySystemType

Description: Definition of constants of the delivery system type under the DVB technology system, including the definitions of constants of the following three types of delivery systems: DVB-C, DTMB and ABS-SS.

A.2.1.1 Constant field – delivery system type

A.2.1.1.1 SATELLITE_DELIVERY_SYSTEM

Prototype: public final static int SATELLITE_DELIVERY_SYSTEM = 0

Description: delivery system type – ABS-SS.

A.2.1.1.2 CABLE_DELIVERY_SYSTEM

Prototype: public final static int CABLE_DELIVERY_SYSTEM = 1

Description: delivery system type – DVB-C.

A.2.1.1.3 TERRESTRIAL_DELIVERY_SYSTEM

Prototype: public final static int TERRESTRIAL_DELIVERY_SYSTEM = 2

Description: delivery system type – DTMB (National Standard Terrestrial Digital TV).

A.2.1.1.4 UNKNOWN_DELIVERY_SYSTEM

Prototype: public final static int UNKNOWN_DELIVERY_SYSTEM = 0xFF

Description: Unknown error type.

A.2.2 Interface org.ngb.broadcast.dvb.tuner.TuningParameters

Prototype: public interface org.ngb.broadcast.dvb.tuner.TuningParameters

Description: Tuning and demodulation parameter interface.

A.2.2.1 Method

A.2.2.1.1 getDeliverySystemType

Prototype: public int getDeliverySystemType()

Description: Getting the applicable delivery system type suitable for tuning and demodulation parameters.

Parameter: None.

Return: Int type, indicating the delivery system type. For the value, please refer to the org.ngb.broadcast.dvb.tuner.DeliverySystemType interface and the "delivery system type" constant field definition.

A.2.3 Interface org.ngb.broadcast.dvb.tuner.TuningListener

Prototype: public interface org.ngb.broadcast.dvb.tuner.TuningListener

Description: Network interface event listener interface, providing a callback method for handling network interface related events, which is implemented by the application layer.

A.2.3.1 Method

A.2.3.1.1 receiveNIEvent

Prototype: public abstract void receiveNIEvent (org.ngb.broadcast.dvb.tuner.TunerEvent anEvent)

Description: Network interface event callback method.

Parameter: anEvent – An org.ngb.broadcast.dvb.tuner. A TunerEvent object, indicating a network interface event.

Return: None.

A.2.4 Class org.ngb.broadcast.dvb.tuner.DvbcTuningParameters

Prototype: public class org.ngb.broadcast.dvb.tuner.DvbcTuningParameters implements org.ngb.broadcast.dvb.tuner.TuningParameters

Description: Tuning and demodulation parameters type, applicable to the DVB-C delivery system. For getDeliverySystemType(), the return value of the method is fixed to DeliverySystemType.CABLE_DELIVERY_SYSTEM.

A.2.4.1 Constant field – modulation method

A.2.4.1.1 DVB_C_MOD_UNDEFINED

Prototype: public static final int DVB_C_MOD_UNDEFINED = 0

Description: DVB-C modulation method – undefined.

A.2.4.1.2 DVB_C_MOD_QAM16

Prototype: public static final int DVB_C_MOD_QAM16 = 1

Description: DVB-C modulation method – 16-QAM.

A.2.4.1.3 DVB_C_MOD_QAM32

Prototype: public static final int DVB_C_MOD_QAM32 = 2

Description: DVB-C modulation method – 32-QAM.

A.2.4.1.4 DVB_C_MOD_QAM64

Prototype: public static final int DVB_C_MOD_QAM64 = 3

Description: DVB-C modulation method – 64-QAM.

A.2.4.1.5 DVB_C_MOD_QAM128

Prototype: public static final int DVB_C_MOD_QAM128 = 4

Description: DVB-C modulation method – 128-QAM.

A.2.4.1.6 DVB_C_MOD_QAM256

Prototype: public static final int DVB_C_MOD_QAM256 = 5

Description: DVB-C modulation method – 256-QAM.

A.2.4.2 Method

A.2.4.2.1 DvbcTuningParameters

Prototype: public DvbcTuningParameters()

Description: A construction method, creating a default DVB-C tuning and demodulation parameter object.

Parameter: None.

A.2.4.2.2 DvbcTuningParameters

Prototype: public DvbcTuningParameters(int frequency, int modulation, int symbolRate)

Description: A construction method, creating a DVB-C tuning and demodulation parameter object according to the input parameters.

Parameter: frequency – Int type, indicating the center frequency of DVB-C signal, in kHz;

modulation – Int type, indicating the DVB-C signal modulation method. For the value, please refer to the "modulation method" constant field definition of the org.ngb.broadcast.dvb.tuner.DvbcTuningParameters class;

symbolRate – Int type, indicating the symbol rate of the DVB-C signal, in ksymbol/s.

A.2.4.2.3 setFrequency

Prototype: public void setFrequency(int frequency)

Description: Setting the DVB-C signal tuning frequency.

Parameter: frequency – Int type, indicating the center frequency of the DVB-C signal, in kHz.

Return: None.

A.2.4.2.4 getFrequency

Prototype: public int getFrequency()

Description: Getting DVB-C signal tuning frequency.

Parameter: None.

Return: Int type, indicating the tuning frequency of the DVB-C signal, in kHz.

A.2.4.2.5 setModulation

Prototype: public void setModulation(int modulation)

Description: Setting the modulation method.

Parameter: Int type, indicating DVB-C signal modulation method. For the value, please refer to the "modulation method" constant field definition of the org.ngb.broadcast.dvb.tuner.DvbcTuningParameters class.

Return: None.

A.2.4.2.6 getModulation

Prototype: public int getModulation()

Description: Getting the modulation method.

Parameter: None.

Return: Int type, indicating DVB-C signal modulation method. For the value, please refer to the "modulation method" constant field definition of the org.ngb.broadcast.dvb.tuner.DvbcTuningParameters class.

A.2.4.2.7 setSymbolRate

Prototype: public void setSymbolRate(int symbolRate)

Description: Setting the symbol rate of the DVB-C signal.

Parameter: symbolRate – Int type, indicating the symbol rate of the DVB-C signal, in ksymbol/s.

Return: None.

A.2.4.2.8 getSymbolRate

Prototype: public int getSymbolRate()

Description: Getting DVB-C signal symbol rate.

Parameter: None.

Return: Int type, indicating the symbol rate of the DVB-C signal, in ksymbol/s.

A.2.5 Class org.ngb.broadcast.dvb.tuner.AbsssTuningParameters

Prototype: public class org.ngb.broadcast.dvb.tuner.AbsssTuningParameters implements org.ngb.broadcast.dvb.tuner.TuningParameters

Description: Tuning parameter class, applicable to the ABS-SS delivery system.

For getDeliverySystemType(), the method return value is fixed to SATELLITE_DELIVERY_SYSTEM.

A.2.5.1 Constant field – polarization method

A.2.5.1.1 ABS_SS_POLAR_LINEAR_H

Prototype: public static final int ABS_SS_POLAR_LINEAR_H = 0

Description: ABS-SS polarization method – horizontal line polarization.

A.2.5.1.2 ABS_SS_POLAR_LINEAR_V

Prototype: public static final int ABS_SS_POLAR_LINEAR_V = 1

Description: ABS-SS polarization method – vertical line polarization.

A.2.5.1.3 ABS_SS_POLAR_CIRCULAR_L

Prototype: public static final int ABS_SS_POLAR_CIRCULAR_L = 2

Description: ABS-SS polarization method – Left-hand circular polarization.

A.2.5.1.4 ABS_SS_POLAR_CIRCULAR_R

Prototype: public static final int ABS_SS_POLAR_CIRCULAR_R = 3

Description: ABS-SS polarization method – Right-hand circular polarization.

A.2.5.2 Method

A.2.5.2.1 AbsssTuningParameters

Prototype: public AbsssTuningParameters()

Description: A construction method, creating a default ABS-SS tuning and demodulation parameter object.

Parameter: None.

A.2.5.2.2 AbsssTuningParameters

Prototype: public AbsssTuningParameters(int frequency, int symbolRate, int polarization)

Description: A construction method, creating an ABS-SS tuning and demodulation parameter object according to the input parameters.

Parameter: frequency – Int type, indicating the center frequency of the ABS-SS signal, in MHz;

symbolRate – Int type, indicating the symbol rate of ABS-SS signal, in ksymbol/s;

polarization – Int type, indicating the polarization method of the ABS-SS signal.

A.2.5.2.3 setFrequency

Prototype: public void setFrequency(int freq)

Description: Setting the center frequency of the ABS-SS signal.

Parameter: freq – Int type, indicating the center frequency of the ABS-SS signal, in MHz.

Return: None.

A.2.5.2.4 getFrequency

Prototype: public int getFrequency()

Description: Getting the center frequency of the ABS-SS signal.

Parameter: None.

Return: Int type, indicating the center frequency of the ABS-SS signal, in MHz.

A.2.5.2.5 setPolarization

Prototype: public void setPolarization (int polarization)

Description: Setting the polarization method.

Parameter: polarization – Int type, indicating the polarization method of the ABS-SS signal.

Return: None.

A.2.5.2.6 getPolarization

Prototype: public int getPolarization()

Description: Getting ABS-SS signal polarization method.

Parameter: None.

Return: Int type, indicating the polarization method of the ABS-SS signal.

A.2.5.2.7 setSymbolRate

Prototype: public void setSymbolRate(int symbolRate)

Description: Setting the symbol rate of the ABS-SS signal.

Parameter: Int type, indicating the symbol rate of the ABS-SS signal, in ksymbol/s.

Return: None.

A.2.5.2.8 getSymbolRate

Prototype: public int getSymbolRate()

Description: Getting the symbol rate of the ABS-SS signal.

Parameter: None.

Return: Int type, indicating the symbol rate of the ABS-SS signal, in ksymbol/s.

A.2.6 Class org.ngb.broadcast.dvb.tuner.DtmbTuningParameters

Prototype: public class org.ngb.broadcast.dvb.tuner.DtmbTuningParameters implements org.ngb.broadcast.dvb.tuner.TuningParameters

Description: Tuning and demodulation parameter class, applicable to the DTMB delivery system. For getDeliverySystemType(), the method return value is fixed to TERRESTRIAL_DELIVERY_SYSTEM.

A.2.6.1 Constant field – demodulation method

A.2.6.1.1 DTMB_MOD_UNDEFINED

Prototype: public static final int DTMB_MOD_UNDEFINED = 0

Description: DTMB demodulation method – undefined.

A.2.6.1.2 DTMB_MOD_QAM4

Prototype: public static final int DTMB_MOD_QAM4 = 1

Description: DTMB demodulation method – 4-QAM.

A.2.6.1.3 DTMB_MOD_QAM4_NR

Prototype: public static final int DTMB_MOD_QAM4_NR = 2

Description: DTMB demodulation method – 4-QAM-NR.

A.2.6.1.4 DTMB_MOD_QAM16

Prototype: public static final int DTMB_MOD_QAM16 = 3

Description: DTMB demodulation method – 16-QAM.

A.2.6.1.5 DTMB_MOD_QAM32

Prototype: public static final int DTMB_MOD_QAM32 = 4

Description: DTMB demodulation method – 32-QAM.

A.2.6.1.6 DTMB_MOD_QAM64

Prototype: public static final int DTMB_MOD_QAM64 = 5

Description: DTMB demodulation method – 64-QAM.

A.2.6.1.7 DTMB_MOD_QAM128

Prototype: public static final int DTMB_MOD_QAM128 = 6

Description: DTMB demodulation method – 128-QAM.

A.2.6.1.8 DTMB_MOD_QAM256

Prototype: public static final int DTMB_MOD_QAM256 = 7

Description: DTMB demodulation method – 256-QAM.

A.2.6.1.9 DTMB_MOD_QAM512

Prototype: public static final int DTMB_MOD_QAM512 = 8

Description: DTMB demodulation method – 512-QAM.

A.2.6.1.10 DTMB_BANDWIDTH_6M

Prototype: public static final int DTMB_BANDWIDTH_6M = 6000

Description: DTMB frequency bandwidth 6M.

A.2.6.1.11 DTMB_BANDWIDTH_7M

Prototype: public static final int DTMB_BANDWIDTH_7M = 7000

Description: DTMB frequency bandwidth 7M.

A.2.6.1.12 DTMB_BANDWIDTH_8M

Prototype: public static final int DTMB_BANDWIDTH_8M = 8000

Description: DTMB frequency bandwidth 8M.

A.2.6.2 Method

A.2.6.2.1 DtmbTuningParameters

Prototype: public DtmbTuningParameters()

Description: A construction method, creating a default DTMB tuning and demodulation parameter object.

Parameter: None.

A.2.6.2.2 DtmbTuningParameters

Prototype: public DtmbTuningParameters(int frequency, int modulation)

Description: A construction method, creating a DTMB tunning and demodulation parameter object based on the input parameters.

Parameter: frequency – Int type, indicating the center frequency of the DTMB signal, in kHz.

modulation – Int type, indicating the modulation method of the DTMB signal. For the value, please refer to the constant field definition of the "modulation method" of the org.ngb.broadcast.dvb.tuner.DtmbTuningParameters class.

A.2.6.2.3 setFrequency

Prototype: public void setFrequency(int frequency)

Description: Setting the center frequency of the DTMB signal.

Parameter: frequency – Int type, indicating the center frequency of the DTMB signal, in kHz.

Return: None.

A.2.6.2.4 setModulation

Prototype: public void setModulation (int modulation)

Description: Setting the DTMB signal modulation method.

Parameter: modulation – Int type, indicating the modulation method of the DTMB signal. For the value, please refer to the constant field definition of the "modulation method" of the org.ngb.broadcast.dvb.tuner.DtmbTuningParameters class.

Return: None.

A.2.6.2.5 setBandWidth

Prototype: public void setBandWidth(int bandWidth)

Description: Setting the DTMB frequency bandwidth.

Parameter: bandWidth – Int type, indicating the DTMB frequency bandwidth.

Return: None.

A.2.6.2.6 getFrequency

Prototype: public int getFrequency()

Description: Getting the center frequency of the DTMB signal.

Parameter: None.

Return: Int type, indicating the center frequency of the DTMB signal, in kHz.

A.2.6.2.7 getModulation

Prototype: public int getModulation()

Description: Getting the DTMB signal modulation method. After successful tuning and demodulation, call this method to return the modulation mode of the DTMB signal, otherwise the return value is meaningless.

Parameter: None.

Return: Int type, indicating the modulation method of the DTMB signal. For the value, please refer to the constant field definition of the "modulation method" of the org.ngb.broadcast.dvb.tuner.DtmbTuningParameters class.

A.2.6.2.8 getBandWidth

Prototype: public int getBandWidth()

Description: Getting the DTMB frequency bandwidth. After successful tuning and demodulation, call this method to return the frequency bandwidth of the DTMB signal, otherwise the return value is meaningless.

Parameter: None.

Return: Int type, indicating the DTMB frequency bandwidth.

A.2.6.2.9 getCodingRatio

Prototype: public java.lang.String getCodingRatio()

Description: Getting DTMB signal coding efficiency. After successful tuning and demodulation, call this method to return the coding efficiency of the DTMB signal, otherwise the return value is meaningless.

Parameter: None.

Return: A java.lang.String object, indicating the string description of the DTMB signal encoding efficiency, which can be "0.4", "0.6" or "0.8".

A.2.6.2.10 getPNMode

Prototype: public java.lang.String getPNMode()

Description: Getting the DTMB signal frame header mode. After successful tuning and demodulation, call this method to return the frame header mode of the DTMB signal, otherwise the return value is meaningless.

Parameter: None.

Return: A java.lang.String object, indicating the DTMB signal frame header mode, such as "PN945".

A.2.7 Class org.ngb.broadcast.dvb.tuner.TunerEvent

Prototype: public abstract class org.ngb.broadcast.dvb.tuner.TunerEvent extends java.lang.Object

Description: A tuner and demodulator event class, the base class of a group of network interface related events defined by this package.

A.2.7.1 Method

A.2.7.1.1 TunerEvent

Prototype: public TunerEvent(java.lang.Object tuner)

Description: Construction method.

Parameter: tuner – A java.lang.Object object, indicating the tuner that issued this event.

A.2.7.1.2 getSource

Prototype: public java.lang.Object getSource()

Description: Getting the object that generated the event.

Parameter: None.

Return: A java.lang.Object object, returning the object that generated the event.

A.2.8 Class org.ngb.broadcast.dvb.tuner.TunerTuningEvent

Prototype: public class org.ngb.broadcast.dvb.tuner.TunerTuningEvent extends org.ngb.broadcast.dvb.tuner.TunerEvent

Description: An event that the network interface starts tuning, inheriting the org.ngb.broadcast.dvb.tuner.TunerEvent class.

A.2.8.1 Method

A.2.8.1.1 TunerTuningEvent

Prototype: public TunerTuningEvent (Object tuner)

Description: Construction method.

Parameter: tuner – A java.lang.Object object, indicating the tuner object that starts tuning.

A.2.9 Class org.ngb.broadcast.dvb.tuner.TunerTuningOverEvent

Prototype: public class org.ngb.broadcast.dvb.tuner.TunerTuningOverEvent extends org.ngb.broadcast.dvb.tuner.TunerEvent

Description: An event that the tuning ends, inheriting the org.ngb.broadcast.dvb.tuner.TunerEvent class.

A.2.9.1 Constant field – tuning result

A.2.9.1.1 SUCCEEDED

Prototype: public final static int SUCCEEDED = 0

Description: The frequency lock is successful.

A.2.9.1.2 FAILED

Prototype: public final static int FAILED = 1

Description: The frequency lock fails.

A.2.9.2 Method

A.2.9.2.1 TunerTuningOverEvent

Prototype: public TunerTuningOverEvent(java.lang.Object tuner, int status)

Description: Construction method.

Parameter: tuner – A java.lang.Object object, indicating the network interface that completes the tuning;

status – Int type, indicating the status code that indicates whether the tuning action is successful.

Return: None.

A.2.9.2.2 getStatus

Prototype: public int getStatus()

Description: Getting the status code of the end of the tuning operation.

Parameter: None.

Return: Int type, indicating the status code of the end of the tuning. The value can be SUCCEEDED or FAILED.

A.2.10 Class org.ngb.broadcast.dvb.tuner.Tuner

Prototype: public class org.ngb.broadcast.dvb.tuner.Tuner

Description: A tuning and demodulation controller class.

A.2.10.1 Method

A.2.10.1.1 tune

Prototype: public void tune(org.ngb.broadcast.dvb.tuner.TuningParameters tuningParameters) throws java.lang.IllegalArgumentException, org.ngb.broadcast.dvb.tuner.TunerException

Description: An asynchronous method, performing tuning and demodulation according to the specified tuning and demodulation parameters.

– If an exception occurs, the state of the network interface will remain unchanged and no event will be generated;
– If no exception occurs, this method will send org.ngb.broadcast.dvb.tuner.TunerTuningEvent and org.ngb.broadcast.dvb.tuner.TunerTuningOverEvent events to the network interface event listener.

Parameter: tuningParameters – An org.ngb.broadcast.dvb.tuner.TuningParameters object, indicating the tuning parameters. The method will further determine the type of the input parameter through the instanceof method.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input tuning and demodulation parameters are invalid, this exception is thrown.

org.ngb.broadcast.dvb.tuner.TunerException – If the tuning fails, this exception is thrown.

A.2.10.1.2 getSignalIntensity

Prototype: public int getSignalIntensity()

Description: Getting the signal strength (signal level) of the current tuning frequency point.

Parameter: None.

Return: Int type, indicating the signal strength expressed as a percentage, the value range being 0-100, 0 indicating the signal strength is the weakest, and 100 indicating the signal strength is the strongest.

A.2.10.1.3 getSignalQuality

Prototype: public int getSignalQuality()

Description: Getting the signal quality (signal-to-noise ratio) of the current tuning frequency point.

Parameter: None.

Return: Int type, indicating the signal quality expressed as a percentage, the value range being 0 to 100, 0 indicating the signal quality is the worst, and 100 indicating the signal quality is the best.

A.2.10.1.4 getCurrentTunningParam

Prototype: public org.ngb.broadcast.dvb.tuner.TuningParameters getCurrentTunningParam()

throws org.ngb.broadcast.dvb.tuner.TuningParameterNotFoundException

Description: Getting the tuning parameters of the current tuning frequency point.

Parameter: None.

Return: Getting the tuning parameters of the current tuning frequency point. When it fails or org.ngb.broadcast.dvb.tuner.DeliverySystemType is wrong, an exception is thrown. After successful acquisition, you need to determine the specific type of the org.ngb.broadcast.dvb.tuner.TuningParameters instance.

A.2.10.1.5 addNetworkInterfaceListener

Prototype: public void addNetworkInterfaceListener(org.ngb.broadcast.dvb.tuner.TuningListener listener)

Description: Register the network interface event listener.

Parameter: listener – An org.ngb.broadcast.dvb.tuner.TuningListener object, indicating the network interface event listener to be registered.

Return: None.

A.2.10.1.6 removeNetworkInterfaceListener

Prototype: public void removeNetworkInterfaceListener(org.ngb.broadcast.dvb.tuner. TuningListener listener)

Description: Log off the network interface event listener.

Parameter: listener – An org.ngb.broadcast.dvb.tuner.TuningListener object, indicating the network interface event listener to be registered.

Return: None.

A.2.10.1.7 getCurrentTransportStream

Prototype: public org.davic.mpeg.TransportStream getCurrentTransportStream()

Description: Getting the transport stream to which the current broadcast network interface is tuned.

Parameter: None.

Return: An org.davic.mpeg.TransportStream object, indicating the transport stream to which the current network interface is tuned. If the current network interface is not tuned to any transport stream, null is returned.

A.2.10.1.8 getDeliverySystemType

Prototype: public int getDeliverySystemType()

Description: Getting the current broadcast network interface delivery type.

Parameter: None.

Return: Int type, indicating the delivery type of the current network interface.

A.2.11 Class org.ngb.broadcast.dvb.tuner.TunerManager

Prototype: public class org.ngb.broadcast.dvb.tuner.TunerManager

Description: The modulation demodulation manager class is used to track the broadcast network interface connected to the receiving device, and is the entry class of the tuning and demodulation unit.

A.2.11.1 Method

A.2.11.1.1 getInstance

Prototype: public static org.ngb.broadcast.dvb.tuner.TunerManager getInstance()

Description: Getting the only instance of the TunerManager class implemented by the system.

Parameter: None.

Return: org.ngb.broadcast.dvb.tuner, a TunerManager class singleton.

A.2.11.1.2 getNetworkInterfaces

Prototype: public org.ngb.broadcast.dvb.tuner.Tuner[] getNetworkInterfaces()

Description: Getting an array of all network interface objects connected to the receiving device.

Parameter: None.

Return: An array of Tuner objects. If not, it returns null.

A.2.12 Exception org.ngb.broadcast.dvb.tuner.TunerException

Prototype: public class org.ngb.broadcast.dvb.tuner.TunerException extends java.lang.Exception

Description: Network interface exception, a group of base classes for other network exceptions defined by this package.

A.2.13 Exception org.ngb.broadcast.dvb.tuner.IncorrectLocatorException

Prototype: public class org.ngb.broadcast.dvb.tuner.IncorrectLocatorException extends org.ngb.broadcast.dvb.tuner.TunerException

Description: An exception that the format of the locator is incorrect. It inherits the org.ngb.broadcast.dvb.tuner.TunerException class.

A.2.14 Exception org.ngb.broadcast.dvb.tuner.StreamNotFoundException

Prototype: public class org.ngb.broadcast.dvb.tuner.StreamNotFoundException extends org.ngb.broadcast.dvb.tuner.TunerException

Description: No exception is found in the stream. It inherits the org.ngb.broadcast.dvb.tuner.TunerException class. This exception is thrown when the reference to the transport stream cannot be resolved because the transport stream is not in the StreamTable.

A.2.15 Exception org.ngb.broadcast.dvb.tuner.TuningParameterNotFoundException

Prototype: public class org.ngb.broadcast.dvb.tuner.TuningParameterNotFoundException extends org.ngb.broadcast.dvb.tuner.TunerException

Description: It inherits org.ngb.broadcast.dvb.tuner.TunerException class. When getting the current tuning and demodulation parameters fails or org.ngb.broadcast.dvb.tuner.DeliverySystemType is wrong, this exception is thrown.

Annex B

JAVA-Broadcast protocol processing unit

(This annex forms an integral part of this Recommendation.)

B.1 Overview

This annex defines JAVA interfaces related to broadcast protocol processing, including MPEG object definition module, DVB object definition module, SECTION segment filtering module, URL encapsulation module, DVB locator module and broadcast protocol processing module.

B.2 MPEG object definition module

The MPEG object definition module defines the most basic objects under the MPEG-2 system and the exceptions in the system.

Defined the most basic MPEG-2 objects are:

– Transport Stream;

– Elementary Stream;

– Service.

Defined MPEG-2 exceptions are:

– Not Authorized Exception;

– Resource Exception;

– Tuning Exception.

The summary of MPEG object definition module is shown in Table B.1.

Table B.1 – Summary of MPEG object definition module

Interface	
NotAuthorizedInterface	An unauthorized broadcast content report interface, defining the failure reason constant and providing a method to find the failure reason.
Class	
TransportStream	An MPEG-2 transport stream class, indicating an MPEG-2 transport stream (TS) and providing a method for obtaining transport stream information.
ElementaryStream	An MPEG-2 elementary stream class, indicating an elementary stream (ES) carried in a transport stream (TS) and providinga method for obtaining elementary stream information.
Service	An MPEG-2 service class, indicating an MPEG-2 service carried in the transport stream, and providing a method for obtaining service information.
Exception	
NotAuthorizedException	An unauthorized broadcast content exception, implements NotAuthorizedInterface interface, thrown when accessing unauthorized scrambled data.
TuningException	Tuning and demodulation exception, thrown when tuning and demodulation fails.
ResourceException	Resource exception, thrown when operation cannot be performed due to lack of resources.

B.2.1 Interface org.davic.mpeg.NotAuthorizedInterface

Prototype: public interface org.davic.mpeg.NotAuthorizedInterface

Description: An unauthorized broadcast content report interface, defines the failure reason constant, and providing a method to find the failure reason.

Example:

(1) Call the getType() method to determine whether the descrambling failure occurred in the service or the basic stream:

– If the getType() method returns SERVICE, call the getService() method to obtain the org.davic.mpeg.Service object that cannot be descrambled;

– If the getType() method returns ELEMENTARY_STREAM, call the getElementaryStreams() method to obtain an array of org.davic.mpeg.ElementaryStream objects that cannot be descrambled;

(2) Call the getReason() method to get the specific reason why it cannot be descrambled:

– POSSIBLE_UNDER_CONDITIONS, including the following secondary reasons:

1.1 COMMERCIAL_DIALOG

1.2 MATURITY_RATING_DIALOG

1.3 TECHNICAL_DIALOG

1.4 FREE_PREVIEW_DIALOG

1.5 OTHER

– NOT_POSSIBLE, including the following secondary reasons:

1.6 NO_ENTITLEMENT

1.7 MATURITY_RATING

1.8 TECHNICAL

1.9 GEOGRAPHICAL_BLACKOUT

1.10 OTHER

B.2.1.1 Constant field – main reason for failure

B.2.1.1.1 POSSIBLE_UNDER_CONDITIONS

Prototype: public static final int POSSIBLE_UNDER_CONDITIONS = 0

Description: Main reason for failure – Accessible under certain conditions.

B.2.1.1.2 NOT_POSSIBLE

Prototype: public static final int NOT_POSSIBLE = 1

Description: Main reason for failure – Impossible to access.

B.2.1.2 Constant field – secondary reason for failure

B.2.1.2.1 COMMERCIAL_DIALOG

Prototype: public static final int COMMERCIAL_DIALOG = 1

Description: The secondary reason for POSSIBLE_UNDER_CONDITIONS – A conversation about payment is required.

B.2.1.2.2 MATURITY_RATING_DIALOG

Prototype: public static final int MATURITY_RATING_DIALOG = 2

Description: The secondary reason for POSSIBLE_UNDER_CONDITIONS – A conversation about age restrictions is required.

B.2.1.2.3 TECHNICAL_DIALOG

Prototype: public static final int TECHNICAL_DIALOG = 3

Description: The secondary reason for POSSIBLE_UNDER_CONDITIONS – A dialogue about technical support is required.

B.2.1.2.4 FREE_PREVIEW_DIALOG

Prototype: public static final int FREE_PREVIEW_DIALOG = 4

Description: The secondary reason for POSSIBLE_UNDER_CONDITIONS – A dialogue explaining the free preview is required.

B.2.1.2.5 NO_ENTITLEMENT

Prototype: public static final int NO_ENTITLEMENT = 1

Description: The secondary reason for NOT_POSSIBLE – The user is not authorized.

B.2.1.2.6 MATURITY_RATING

Prototype: public static final int MATURITY_RATING = 2

Description: The secondary reason for NOT_POSSIBLE – Operation is not allowed due to age reasons.

B.2.1.2.7 TECHNICAL

Prototype: public static final int TECHNICAL = 3

Description: The secondary reason for NOT_POSSIBLE – The operation is not allowed due to some technical reasons.

B.2.1.2.8 GEOGRAPHICAL_BLACKOUT

Prototype: public static final int GEOGRAPHICAL_BLACKOUT = 4

Description: The secondary reason for NOT_POSSIBLE – Operation is not allowed due to geographic reasons.

B.2.1.2.9 OTHER

Prototype: public static final int OTHER = 5

Description: The secondary reason for POSSIBLE_UNDER_CONDITIONS and NOT_POSSIBLE – Other reasons.

B.2.1.3 Constant field – MPEG-2 object

B.2.1.3.1 SERVICE

Prototype: public static final int SERVICE = 0

Description: The MPEG-2 object where the error occurred – Access to the service object is denied.

B.2.1.3.2 ELEMENTARY_STREAM

Prototype: public static final int ELEMENTARY_STREAM = 1

Description: The MPEG-2 object where the error occurred – Access to the elementary stream object is denied.

B.2.1.4 Method

B.2.1.4.1 getType

Prototype: public int getType()

Description: Getting the MPEG-2 object type where the unauthorized error occurred.

Parameter: None.

Return: Int type, indicating the type of MPEG-2 object that cannot be descrambled. The value can be SERVICE or ELEMENTARY_STREAM. For details, see the "MPEG-2 Object" constant field definition of the org.davic.mpeg.NotAuthorizedInterface interface.

B.2.1.4.2 getService

Prototype: public org.davic.mpeg.Service getService()

Description: Getting MPEG-2 service objects that cannot be descrambled.

Parameter: None.

Return: A service object, indicating a service that cannot be descrambled.

– If the type returned by the getType() method is SERVICE, this method returns a service object that cannot be descrambled;

– If the type returned by the getType() method is ELEMENTARY_STREAM, this method returns null.

B.2.1.4.3 getElementaryStreams

Prototype: public org.davic.mpeg.ElementaryStream[] getElementaryStreams()

Description: Getting the elementary stream object that cannot be descrambled.

Parameter: None.

Return: An array of ElementaryStream objects, indicating an elementary stream that cannot be descrambled. Otherwise, it returns null.

– If the type returned by the getType() method is ELEMENTARY_STREAM, this method returns an array of elementary stream objects that cannot be descrambled;

– If the type returned by the getType() method is SERVICE, this method returns null.

B.2.1.4.4 getReason

Prototype: public int[] getReason(int index) throws java.lang.IndexOutOfBoundsException

Description: Getting the reason why it cannot be descrambled.

Parameter: index – Int type, indicating the serial number of the component.

– If the descrambling of the MPEG-2 service object fails, this parameter is set to 0;

– If the descrambling of the MPEG-2 elementary stream object fails, the index parameter indicates the sequence number of the org.davic.mpeg.ElementaryStream object array returned by the getElementaryStreams() method.

Return: Int type array, the length of the array is 2, int[0] indicating the main reason, int[1] indicating the secondary reason. For the value, please refer to the constant field definition "main reason for failure" and "secondary reason for failure" of the org.davic.mpeg.NotAuthorizedInterface interface.

Exception: java.lang.IndexOutOfBoundsException – If the MPEG-2 service object descrambling fails and the index parameter takes a non-zero value, this exception is thrown; if the MPEG-2 elementary stream object descrambling fails, and the index exceeds the length of the ElementaryStream array returned by the getElementaryStreams() method, this exception is thrown.

B.2.2 Class org.davic.mpeg.TransportStream

Prototype: public class org.davic.mpeg.TransportStream

Description: An MPEG-2 transport stream class, indicating an MPEG-2 transport stream (TS), and providing a method for obtaining transport stream information. A transport stream object, indicating a transport stream that can be accessed through a specific network interface, which indicates that there is an implicit relationship between the transport stream object and the network interface. Therefore, if multiple network interfaces send the same transport stream at the same time, a separate transport stream object should also be constructed for the transport stream sent by each interface.

B.2.2.1 Method

B.2.2.1.1 getTransportStreamId

Prototype: public int getTransportStreamId()

Description: Obtain the transport stream identifier (that is, obtain the transport_stream_id field information in the PAT).

Parameter: None.

Return: Int type, indicating the identifier of the transport stream.

B.2.2.1.2 retrieveService

Prototype: public org.davic.mpeg.Service retrieveService(int serviceId)

Description: Obtain the service object carried in the transport stream according to the specified service identifier.

Parameter: serviceId – Int type, indicating the service identifier of the requested service.

Return: An org.davic.mpeg.Service object, indicating the requested service. If the specified service does not exist, null is returned.

B.2.2.1.3 retrieveServices

Prototype: public org.davic.mpeg.Service[] retrieveServices()

Description: Getting all service objects carried in the transport stream.

Parameter: None.

Return: An array of service objects, indicating all the services carried in the transport stream. If there is no service object, null is returned.

B.2.3 Class org.davic.mpeg.ElementaryStream

Prototype: public class org.davic.mpeg.ElementaryStream

Description: An MPEG-2 elementary stream class, indicating an elementary stream (ES) carried in a transport stream (TS), and providing a method for obtaining elementary stream information. If an elementary stream is used for multiple services, multiple instances of this class should be implemented, that is, one instance is used for one service object.

Description: The elementary stream information defined in this class comes from PMT.

B.2.3.1 Method

B.2.3.1.1 getPID

Prototype: public int getPID()

Description: Obtain the PID value of the transport packet carrying the elementary stream.

Parameter: None.

Return: Int type, indicating the PID value of the transport packet carrying the elementary stream.

B.2.3.1.2 getAssociationTag

Prototype: public java.lang.Integer getAssociationTag()

Description: Obtain the DSM-CC association identifier of the elementary stream.

Parameter: None.

Return: A java.lang.Integer object, indicating the DSM-CC associated identifier of the elementary stream, or null is returned if there is no associated identifier.

B.2.3.1.3 getService

Prototype: public org.davic.mpeg.Service getService()

Description: Getting the service to which the elementary stream belongs.

Parameter: None.

Return: An org.davic.mpeg.Service object, indicating the service that contains the elementary stream.

B.2.4 Class org.davic.mpeg.Service

Prototype: public class org.davic.mpeg.Service

Description: An MPEG-2 service class, indicating an MPEG-2 service carried in the transport stream, and providing a method for obtaining service information.

B.2.4.1 Method

B.2.4.1.1 getServiceId

Prototype: public int getServiceId()

Description: Getting the service identifier (service_id).

Parameter: None.

Return: Int type, indicating the service identifier of the service.

B.2.4.1.2 getTransportStream

Prototype: org.davic.mpeg.TransportStream getTransportStream()

Description: Obtain the transport stream object that carries the service.

Parameter: None.

Return: An org.davic.mpeg.TransportStream object, indicating the transport stream object that carries the service.

B.2.4.1.3 retrieveElementaryStream

Prototype: public org.davic.mpeg.ElementaryStream retrieveElementaryStream(int pid)

Description: According to the specified transport stream PID, obtain the elementary stream objects contained in the service.

Parameter: pid – Int type, indicating the identifier of the transport packet that carries the elementary stream.

Return: An ElementaryStream object indicating the elementary stream contained in the service, returning null if no specified elementary stream object is found.

B.2.4.1.4 retrieveElementaryStreams

Prototype: public org.davic.mpeg.ElementaryStream[] retrieveElementaryStreams()

Description: Getting all the elementary stream objects in the service.

Parameter: None.

Return: An org.davic.mpeg.ElementaryStream object array indicating all the elementary streams in the service, and returning null if none.

B.2.5 Exception org.davic.mpeg.NotAuthorizedException

Prototype: public class org.davic.mpeg.NotAuthorizedException extends java.langException implements NotAuthorizedInterface

Description: Unauthorized broadcast content exception, thrown when accessing unauthorized scrambled data.

B.2.6 Exception org.davic.mpeg.TuningException

Prototype: public class org.davic.mpeg.TuningException extends java.lang.Exception

Description: Tuning and demodulation exception, thrown when tuning and demodulation fails.

B.2.7 Exception org.davic.mpeg.ResourceException

Prototype: public class org.davic.mpeg.ResourceException extends java.lang.Exception

Description: Resource lack exception, thrown when operations cannot be performed due to resource lack.

B.3 DVB object definition module

The DVB object definition module defines the basic objects of MPEG-2 under the DVB system:

– DVB transport stream class (org.davic.mpeg.dvb.DvbTransportStream);

– DVB elementary stream class (org.davic.mpeg.dvb.DvbElementaryStream);

– DVB broadcasting service class (org.davic.mpeg.dvb.DvbService).

The summary of DVB object definition module is shown in Table B.2.

Table B.2 – Summary of DVB object definition module

Class	
DvbElementaryStream	A DVB elementary stream class, indicating an MPEG-2 elementary stream (ES) that conforms to the semantic constraints of DVB carried in a transport stream (TS), and providing a method for obtaining DVB elementary stream information.
DvbService	A DVB service class, indicating an MPEG-2 service that conforms to the semantic constraints of DVB carried in the transport stream, and providing a method for obtaining DVB service information.
DvbTransportStream	A DVB transport stream class, indicating an MPEG-2 transport stream that conforms to the semantic constraints of DVB, and providing a method for obtaining DVB transport stream information.

B.3.1 Class org.davic.mpeg.dvb.DvbElementaryStream

Prototype: public class org.davic.mpeg.dvb.DvbElementaryStream extends org.davic.mpeg.ElementaryStream

Description: A DVB elementary stream class, indicating an MPEG-2 elementary stream (ES) that conforms to the semantic constraints of DVB carried in a transport stream (TS), and providing a method for obtaining DVB elementary stream information.

B.3.1.1 Method

B.3.1.1.1 getComponentTag

Prototype: public java.lang.Integer getComponentTag()

Description: Getting the component tag of the DVB elementary stream (that is, get the value of the component_tag field in the descriptor of the stream_identifier_descriptor).

Parameter: None.

Return: A java.lang.Integer object, indicating the component tag of the DVB elementary stream. If there is no component tag (that is, the ES stream ring does not carry the descriptor of the stream_identifier_descriptor), it returns null.

B.3.2 Class org.davic.mpeg.dvb.DvbService

Prototype: public class org.davic.mpeg.dvb.DvbService extends org.davic.mpeg.Service

Description: A DVB service class, indicating an MPEG-2 service that conforms to the semantic constraints of DVB carried in the transport stream, and providing a method for obtaining DVB service information.

B.3.2.1 Method

B.3.2.1.1 retrieveDvbElementaryStream

Prototype: public org.davic.mpeg.dvb.DvbElementaryStream retrieveDvbElementaryStream(int componentTag)

Description: Obtain the DVB elementary stream contained in the DVB service according to the specified component tag.

Parameter: componentTag – Int type, indicating the component tag value of the DVB elementary stream.

Return: An org.davic.mpeg.dvb.DvbElementaryStream object, indicating the DVB elementary stream object. If the DVB elementary stream specified by the parameter componentTag does not exist, null is returned.

B.3.3 Class org.davic.mpeg.dvb.DvbTransportStream

Prototype: public class org.davic.mpeg.dvb.DvbTransportStream extends org.davic.mpeg.TransportStream

Description: A DVB transport stream class, indicating an MPEG-2 transport stream that conforms to the semantic constraints of DVB, and providing a method for obtaining DVB transport stream information.

B.3.3.1 Method

B.3.3.1.1 getNetworkId

Prototype: public int getNetworkId()

Description: Getting the network ID of the network where the DVB transport stream is located.

Parameter: None.

Return: Int type, indicating the network identifier of the network where the transport stream is located.

B.3.3.1.2 getOriginalNetworkId

Prototype: public int getOriginalNetworkId()

Description: Getting the original network ID of the DVB transport stream.

Parameter: None.

Return: Int type, indicating the original network identifier of the transport stream.

B.4 SECTION filter module

The SECTION filter module provides classes and methods related to MPEG-2 section filtering.

The summary of the SECTION filter module is shown in Table B.3.

Table B.3 – Summary of the SECTION filter module

Interface	
SectionFilterListener	The section filter event listener interface, provides a callback method for section filter event handling, which is implemented by the application layer.
Class	
Section	An MPEG-2 section class, describing a section filtered from the transport stream.
SectionFilterGroup	A section filter group class, indicating an MPEG-2 filter group, can be activated and released as a basic operation unit.
SectionFilter	The section filter class, which is a base class for a group of section filter classes with different life cycles and cache length characteristics, provides basic operation methods of filters.
SimpleSectionFilter	The simple section filter class inherits the SectionFilter class.
TableSectionFilter	The table section filter class inherits the SectionFilter class.
RingSectionFilter	The ring section filter class inherits the SectionFilter class.
Event	
SectionFilterEvent	Section filter event class, a base class of a group of section filter event classes.
SectionAvailableEvent	Section data available events, inherit the SectionFilterEvent class, report filtering to a complete section.
VersionChangeDetectedEvent	The section filter version change event, inherits the SectionFilterEvent class.
EndOfFilteringEvent	Section filtering end event, inherits the SectionFilterEvent class, and reports the end of section filtering.
Event	
IncompleteFilteringEvent	Section filtering incomplete event, inherits the EndOfFilteringEvent class.
TimeOutEvent	Section filtering timeout event, inherits the EndOfFilteringEvent class.
FilterResourcesAvailableEvent	Filter resource available events, inherit the ResourceStatusEvent class.
ForcedDisconnectedEvent	The forced disconnection event between the section filter group and the transport stream, inherits the ResourceStatusEvent class.
Exception	
SectionFilterException	The base class for section filter exceptions.
ConnectionLostException	Connection lost exception, inherits the SectionFilterException class.
FilteringInterruptedException	Filtering interrupted exception, inherits the SectionFilterException class.

Table B.3 – Summary of the SECTION filter module

FilterResourceException	Filter resource exception, inherits the SectionFilterException class.
IllegalFilterDefinitionException	Illegal filter definition exception, inherits the SectionFilterException class.
InvalidSourceException	Invalid section data source exception, inherits the SectionFilterException class.
NoDataAvailableException	No data available exception of the section objects inherits the SectionFilterException class.

The relationship between the objects of the SECTION filter module is shown in Figure B.1.

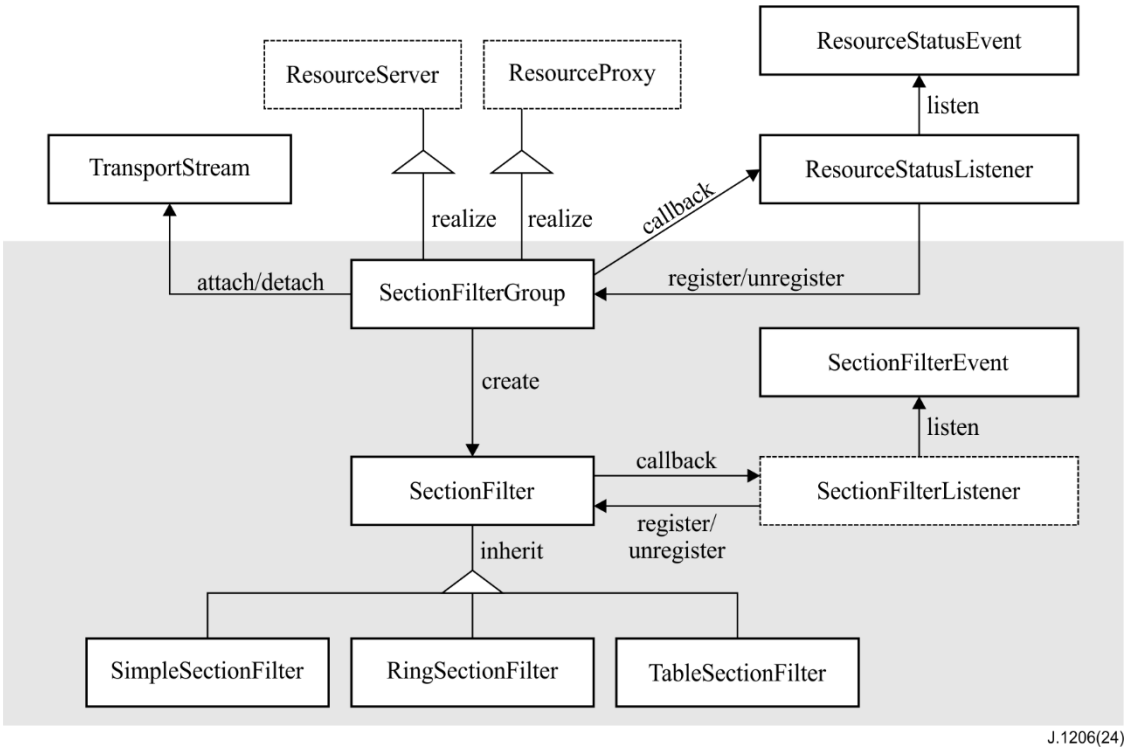


Figure B.1 – Relationship between the objects of the SECTION filter module

The state transition diagram of the connection relationship between the SectionFilterGroup object and the TransportStream object is shown in Figure B.2.

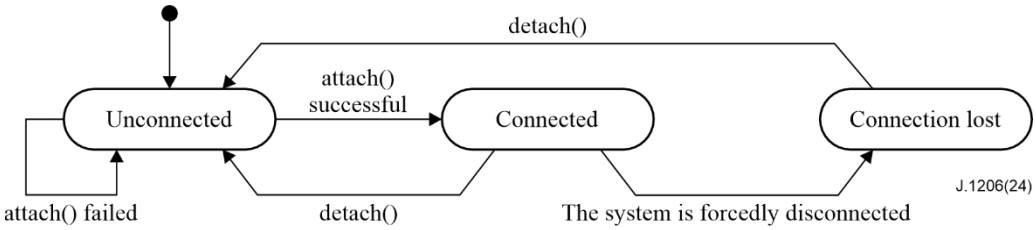


Figure B.2 – State transition diagram of the connection relationship between the SectionFilterGroup object and the TransportStream object

B.4.1 Interface org.davic.mpeg.sections.SectionFilterListener

Prototype: public interface org.davic.mpeg.sections.SectionFilterListener extends java.util.EventListener.

Description: The section filter event listener interface, provides a callback method for section filter event handling, which is implemented by the application layer.

B.4.1.1 Method

B.4.1.1.1 sectionFilterUpdate

Prototype: void sectionFilterUpdate(org.davic.mpeg.sections.SectionFilterEvent event)

Description: The callback method for section filtering event handling.

Parameter: event – An org.davic.mpeg.sections.SectionFilterEvent object, indicating the section filtering event. The method implementer should further determine the type of event parameter through the instanceof method.

Return: None.

B.4.2 Class org.davic.mpeg.sections.Section

Prototype: public class org.davic.mpeg.sections.Section implements Cloneable

Description: The MPEG-2 section class describes a section filtered from the transport stream. The section object copied by the clone() method is an independent new object. The state change of the original object will not affect the new object. Copying objects with the clone() method must be implemented without throwing an exception.

B.4.2.1 Method

B.4.2.1.1 clone

Prototype: public java.lang.Object clone()

Description: Clone the section object. The section object copied by the clone() method is an independent new object. The state change of the original object will not affect the new object. Copying objects with the clone() method must be implemented without throwing an exception.

Rewrite: The clone() method of the object class.

Parameter: None.

Return: Section object.

B.4.2.1.2 current_next_indicator

Prototype: public boolean current_next_indicator() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the current_next_indicator field in the section header.

Parameter: None.

Return: boolean type, true value indicating that the current_next_indicator field is 1, and false indicating that the current_next_indicator field is 0.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.3 getByteAt

Prototype: public byte getByteAt(int index) throws org.davic.mpeg.sections.NoDataAvailableException, java.lang.IndexOutOfBoundsException

Description: Getting the specified byte of the filtered section data in the section object.

Parameter: index-int type, indicating the position of the byte to be retrieved in the section. The index value of the first byte in the section (that is, the table_id field) is 1.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

java.lang.IndexOutOfBoundsException – This exception is thrown if the byte to be retrieved is out of the section data range.

B.4.2.1.4 getData

Prototype: public byte[] getData() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Obtain the filtered section data in the section object, including the section header. Each call to this method will return a new copy of the section data.

Parameter: None.

Return: None.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.5 getData

Prototype: public byte[] getData(int index, int length) throws org.davic.mpeg.sections.NoDataAvailableException, java.lang.IndexOutOfBoundsException

Description: Getting the specified part of the filtered section data in the section object. Each call to this method will return a new copy of the section data.

Parameter: index – Int type, indicating the position of the byte to be retrieved in the section. The index value of the first byte in the section (ie the table_id field) is 1.

length-int type, indicating the number of consecutive bytes of the data to be retrieved starting from the first byte.

Return: None.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

java.lang.IndexOutOfBoundsException – This exception is thrown if a certain part of the data to be retrieved is out of the range of the section data.

B.4.2.1.6 getFullStatus

Prototype: public boolean getFullStatus()

Description: Determine whether the section object contains valid data.

Parameter: None.

Return: boolean type, true value indicating that the section object contains valid data, and false indicating that there is no valid data.

B.4.2.1.7 last_section_number

Prototype: public int last_section_number() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the last_section_number field of the section header.

Parameter: None.

Return: Int type, indicating the value of the last_section_number field.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.8 private_indicator

Prototype: public boolean private_indicator() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the private_indicator field in the section header.

Parameter: None.

Return: None.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.9 section_length

Prototype: public int section_length() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the section_length field in the section header.

Parameter: None.

Return: Int type, indicating the value of the section_length field in the section header.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.10 section_number

Prototype: public int section_number() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the section_number field in the section header.

Parameter: None.

Back:Int type, indicating the value of the section_number field.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.11 section_syntax_indicator

Prototype: public boolean section_syntax_indicator() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the section_syntax_indicator field in the section header.

Parameter: None.

Return: boolean type, true value indicating section_syntax_indicator = 1, the false value indicating section_syntax_indicator=0.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.12 setEmpty

Prototype: public void setEmpty()

Description: The data in the set section object is no longer valid, and is generally used by org.davic.mpeg.sections.RingSectionFilter to indicate that the specific object can be used again.

Parameter: None.

Return: None.

B.4.2.1.13 table_id_extension

Prototype: public int table_id_extension() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the table_id_extension field in the section header.

Parameter: None.

Return: Int type, indicating the value of the table_id_extension field. For the application, it needs to determine the specific semantics of the table_id_extension field according to the type of table_id.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.14 table_id

Prototype: public int table_id() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the table_id field in the section header.

Parameter: None.

Return: Int type, indicating the table_id field of the section.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.2.1.15 version_number

Prototype: public short version_number() throws org.davic.mpeg.sections.NoDataAvailableException

Description: Getting the value of the version_number field in the section header.

Parameter: None.

Return: short type, indicating the value of the version_number field.

Exception: org.davic.mpeg.sections.NoDataAvailableException – If the section data is not filtered, this exception is thrown.

B.4.3 Class org.davic.mpeg.sections.SectionFilterGroup

Prototype: public class org.davic.mpeg.sections.SectionFilterGroup implements org.davic.resources.ResourceProxy, org.davic.resources.ResourceServer

Description: Section filter group class, indicating an MPEG-2 filter group and can be activated and released as a basic operation unit. The purpose of this class is to minimize resource deadlocks that may occur between independent applications.

B.4.3.1 Method

B.4.3.1.1 SectionFilterGroup

Prototype: public SectionFilterGroup(int numberOfFilters) throws java.lang.IllegalArgumentException

Description: Construction method creates a section filter group object.

Parameter: numberOfFilters – Int type, indicating the number of section filters that the section filter group needs to occupy.

java.lang.IllegalArgumentException – If the parameter numberOfFilters is less than 1, this exception is thrown.

B.4.3.1.2 SectionFilterGroup

Prototype: public SectionFilterGroup(int numberOfFilters, boolean resourcePriority) throws java.lang.IllegalArgumentException

Description: Construction method creates a section filter group object.

Parameter: numberOfFilters – Int type, indicating the number of section filters that the section filter group object needs to occupy.

resourcePriority – boolean type, indicating the relative priority of the resources possessed between the filter group object and other objects in the sections package when there are insufficient filter resources. True value indicates high priority, and false indicates low priority. The scope of the resourcePriority parameter is limited to one application.

Exception: java.lang.IllegalArgumentException – If the parameter numberOfFilters is less than 1, this exception is thrown.

B.4.3.1.3 addResourceStatusEventListener

Prototype: public synchronized void addResourceStatusEventListener

(org.davic.resources.ResourceStatusListener listener)

Description: Register the resource status change event listener of the filter group, and implement the addResourceStatusEventListener() method in the interface org.davic.resources.ResourceProxy. If this method is called multiple times, each listener object can be notified of each resource change.

Parameter: listener An org.davic.resources.ResourceStatusListener object, indicating the resource status change event listener to be registered.

Return: None.

B.4.3.1.4 removeResourceStatusEventListener

Prototype: public synchronized void removeResourceStatusEventListener

(org.davic.resources.ResourceStatusListener listener)

Description: Unregister the resource status change event listener of the filter group, and implement the removeResourceStatusEventListener() method in the ResourceProxy interface. If the listener object was not notified originally, this method has no other effect.

Parameter: listener An org.davic.resources.ResourceStatusListener object, indicating the resource status change event listener to be unregistered.

Return: None.

B.4.3.1.5 attach

Prototype: public boolean attach(org.davic.mpeg.TransportStream stream, org.davic.resources.ResourceClient client, java.lang.Object requestData) throws org.davic.mpeg.sections.FilterResourceException, org.davic.mpeg.sections.InvalidSourceException, org.davic.mpeg.TuningException

Description: Associate the filter group object with the transport stream. This filter group will try to obtain the number of section filters specified when it was created. Call the attach() method again on the connected filter group and return false directly. When using filters and filter groups, the order of function calling is: filter group attach, filter startFiltering, filter stopFiltering, filter group detach.

Parameter: stream An org.davic.mpeg.TransportStream object, indicating the transport stream object;

client – An org.davic.resources.esourceClient object, indicating the resource client, the object that needs to be notified when the filter resource is lost;

requestData – A java.lang.Object object, indicating the application-specific data passed to the resource client when the resource manager notifies that the resource is lost.

Return: boolean type, returning true upon success and false upon failure.

Exception: org.davic.mpeg.sections.FilterResourceException – If the reservation of a specific section filter fails, this exception is thrown;

org.davic.mpeg.sections.InvalidSourceException – If the specified transport stream is invalid, this exception is thrown;

org.davic.mpeg.TuningException – If it is not tuned to the specified transport stream, this exception is thrown.

B.4.3.1.6 detach

Prototype: public boolean detach()

Description: Disconnect the filter group object from the transport stream. The section filter occupied by the filter group will be released. The disconnected filter group returns false directly.

Parameter: None.

Return: boolean type, returning true upon success and false upon failure.

B.4.3.1.7 getClient

Prototype: public org.davic.resources.ResourceClient getClient()

Description: Getting the org.davic.resources.ResourceClient object specified in the attach() method so that you can be notified when the section filter is removed from the SectionFilterGroup. Implement the getClient() method in the interface org.davic.resources.ResourceProxy.

Parameter: None.

Return: An org.davic.resources.ResourceClient object, indicating a resource customer. If the filter group is not associated with the transport stream, null is returned.

B.4.3.1.8 getSource

Prototype: public org.davic.mpeg.TransportStream getSource()

Description: Getting the transport stream currently associated with the filter group object.

Parameter: None.

Return: An org.davic.mpeg.TransportStream object, indicating the transport stream associated with this filter group. If the filter group is not associated with the transport stream, null is returned.

B.4.3.1.9 newRingSectionFilter

Prototype: public RingSectionFilter newRingSectionFilter(int ringSize) throws java.lang.IllegalArgumentException

Description: Creating a ring section filter in the parent filter group. Once started, the ring section filter org.davic.mpeg.sections.RingSectionFilter object will use the section filter specified when the parent filter group was created.

Parameter: ringSize – Int type, indicating the number of Section objects occupied by the ring section filter.

Return: An org.davic.mpeg.sections.RingSectionFilter object, indicating a ring section filter.

Exception: java.lang.IllegalArgumentException – If the parameter ringSize is less than 1, this exception is thrown.

B.4.3.1.10 newRingSectionFilter

Prototype: public RingSectionFilter newRingSectionFilter(int ringSize, int sectionSize) throws java.lang.IllegalArgumentException

Description: Creating a ring section filter in the parent filter group. Once started, the ring section filter org.davic.mpeg.sections.RingSectionFilter object will use the section filter specified when the parent filter group was created.

Parameter: ringSize – Int type, indicating the number of Section objects occupied by the ring section filter;

sectionSize – Int type, indicating the length of the section data buffer, in bytes. If the length of the section to be filtered is greater than this value, the excess data will be cut off, and the filtering will continue without any notification to the application.

Return: An org.davic.mpeg.sections.RingSectionFilter object, indicating a ring section filter.

Exception: java.lang.IllegalArgumentException – If the ringSize parameter is less than 1 or the sectionSize parameter is less than 1, this exception is thrown.

B.4.3.1.11 newSimpleSectionFilter

Prototype: public SimpleSectionFilter newSimpleSectionFilter()

Description: Creating a simple section filter in the parent filter group. Once started, the simple section filter org.davic.mpeg.sections.SimpleSectionFilter object will use the section filter specified when the parent filter group was created. This type of section filter has a buffer that can hold the default long section (4096 bytes).

Parameter: None.

Return: An org.davic.mpeg.sections.SimpleSectionFilter object, indicating a simple section filter.

B.4.3.1.12 newSimpleSectionFilter

Prototype: public SimpleSectionFilter newSimpleSectionFilter(int sectionSize) throws java.lang.IllegalArgumentException

Description: Creating a simple section filter in the parent filter group. Once started, the simple section filter org.davic.mpeg.sections.SimpleSectionFilter object will use the section filter specified when the parent filter group was created.

Parameter: sectionSize – Int type, indicating the length of the section data buffer, in bytes. If the length of the section to be filtered is greater than this value, the excess data will be cut off, and the filtering will continue without any notification to the application.

Return: An org.davic.mpeg.sections.SimpleSectionFilter object, indicating a simple section filter.

Exception: java.lang.IllegalArgumentException – If the sectionSize parameter is less than 1, this exception is thrown.

B.4.3.1.13 newTableSectionFilter

Prototype: public TableSectionFilter newTableSectionFilter()

Description: Creating a table section filter in the parent filter group. Once started, the simple section filter org.davic.mpeg.sections.SimpleSectionFilter object will use the section filter specified when the parent filter group was created. Each section of the table section filter has a buffer that contains the default long section (4096 bytes).

Parameter: None.

Return: An org.davic.mpeg.sections.TableSectionFilter object, indicating a table section filter.

B.4.3.1.14 newTableSectionFilter

Prototype: public TableSectionFilter newTableSectionFilter(int sectionSize) throws java.lang.IllegalArgumentException

Description: Creating a table section filter in the parent filter group. Once started, the simple section filter org.davic.mpeg.sections.SimpleSectionFilter object will use the section filter specified when the parent filter group was created.

Parameter: sectionSize – Int type, indicating the length of the section data filter buffer, in byte. When the first section is captured and the total number of sections in the table is known, each section created will have a buffer of this length. If the length of the section to be filtered is greater than this value, the excess data will be truncated and the filtering will continue without any notification to the application.

Return: An org.davic.mpeg.sections.TableSectionFilter object, indicating a table filter. Exception: java.lang.IllegalArgumentException – If the sectionSize parameter is less than 1, this exception is thrown.

B.4.4 Class org.davic.mpeg.sections.SectionFilter

Prototype: public abstract class org.davic.mpeg.sections.SectionFilter

Description: Section filter class, which is a base class for a group of section filter classes with different life cycles and cache length characteristics, provides a basic filter operation method. If an org.davic.mpeg.sections.SectionFilterGroup object is disassociated from the transport stream due to user or resource recovery, etc., the section filter that has already started will continue to work. Therefore, when the org.davic.mpeg.sections.SectionFilterGroup object is again associated with the transport stream, these filters will be activated again.

B.4.4.1 Method

B.4.4.1.1 addSectionFilterListener

Prototype: public synchronized void addSectionFilterListener(org.davic.mpeg.sections. SectionFilterListener listener)

Description: Register the section filter event listener.

Parameter: listener – An org.davic.mpeg.sections.SectionFilterListener object, indicating the section filter event listener to be registered.

Return: None.

B.4.4.1.2 removeSectionFilterListener

Prototype: public synchronized void removeSectionFilterListener (org.davic.mpeg.sections.SectionFilterListener listener)

Description: Unregister the section filter event listener.

Parameter: listener – An org.davic.mpeg.sections.SectionFilterListener object, indicating the section filter event listener to be unregistered.

Return: None.

B.4.4.1.3 setTimeOut

Prototype: public void setTimeOut(long milliseconds) throws java.lang.IllegalArgumentException

Description: Setting the filter timeout period. When a timeout occurs, the generated timeout event org.davic.mpeg.sections.TimeOutEvent will be sent to the section filter event listener, and the filtering will stop.

– For org.davic.mpeg.sections.SimpleSectionFilter, if no section data arrives within a certain period of time, this timeout event will be generated;
– For org.davic.mpeg.sections.TableSectionFilter, if no complete table arrives within a certain period of time, this timeout event will be generated;
– For org.davic.mpeg.sections.RingSectionFilter, this timeout event will be generated if the filter is not successfully performed after a certain period of time since the last successful filtering.

Parameter: milliseconds – long type, indicating the timeout period, in milliseconds. If the parameter is set to 0, the timeout effect will be invalid. Setting the timeout is only valid for subsequent operations. The default value is 0.

Return: None.

Exception: java.lang.IllegalArgumentException – If the milliseconds value of the timeout parameter is negative, this exception is thrown.

B.4.4.1.4 startFiltering

Prototype: public boolean startFiltering(java.lang.Object appData, int pid) throws org.davic.mpeg.sections.FilterResourceException, org.davic.mpeg.NotAuthorizedException, org.davic.mpeg.sections.IllegalFilterDefinitionException, org.davic.mpeg.sections.ConnectionLostException

Description: Start section filtering to filter out all sections carried in the transport packet specified by the pid parameter. The parent filter group org.davic.mpeg.sections.SectionFilterGroup object must first be associated with the transport stream.

Parameter: appData – A java.lang.Object object, indicating the additional data object provided by the application, as a part of all filter event org.davic.mpeg.sections.SectionFilterEvent objects generated by this method call, the object will be passed to the registered section filter event Listener, the application can use this object for internal communication between applications. If the application does not require any additional data, this parameter can be set to null;

pid – Int type, indicating the transport packet identifier (TS_PID) to be filtered.

Return: boolean type, returning true upon success and false upon failure.

Exception: org.davic.mpeg.sections.FilterResourceException – When this method is called, if the total number of section filters started by the parent filter group org.davic.mpeg.sections.SectionFilterGroup is equal to the number of section filters occupied when the filter group is created, this exception is thrown. This exception applies regardless of whether the parent filter group is associated with the TS stream;

org.davic.mpeg.NotAuthorizedException – If the section to be filtered is scrambled and has no descrambling permission, this exception is thrown;

org.davic.mpeg.sections.IllegalFilterDefinitionException – If org.davic.mpeg.sections. TableSectionFilter calls this method, this exception is thrown;

org.davic.mpeg.sections.ConnectionLostException – This exception is thrown if the parent filter group org.davic.mpeg.sections.SectionFilterGroup is in a state of losing connection due to lack of resources or if the section filter cannot complete the method call.

B.4.4.1.5 startFiltering

Prototype: public boolean startFiltering(java.lang.Object appData, int pid, int tableID) throws org.davic.mpeg.sections.FilterResourceException, org.davic.mpeg.NotAuthorizedException, org.davic.mpeg.sections.ConnectionLostException, org.davic.mpeg.sections.IllegalFilterDefinitionException

Description: Start section filtering, filter out the section (section) carried in the transport packet specified by the pid parameter, and the table ID matches the tableID parameter. The parent filter group org.davic.mpeg.sections.SectionFilterGroup object must first be associated with the transport stream.

Parameter: appData – A java.lang.Object object, indicating the additional data object provided by the application, as a part of all the filter event org.davic.mpeg.sections.SectionFilterEvent objects generated by this method call, the object will be passed to the registered section filter event Listener, the application can use this object for internal communication between applications. If the application does not require any additional data, this parameter can be set to null;

pid-Int type, indicating the transport stream packet identifier (TS_PID) to be filtered;

tableID-Int type, indicating the table ID (table_id) to be filtered.

Return: boolean type, returning true upon success and false upon failure.

Exception: org.davic.mpeg.sections.FilterResourceException – When this method is called, if the total number of section filters started by the parent filter group org.davic.mpeg.sections.SectionFilterGroup is equal to the number of section filters occupied when the filter group is created, this exception is thrown. This exception applies regardless of whether the parent filter group is associated with the TS stream;

org.davic.mpeg.NotAuthorizedException – If the section to be filtered is scrambled and has no descrambling permission, this exception is thrown;

org.davic.mpeg.sections.ConnectionLostException – If the parent filter group org.davic.mpeg.sections.SectionFilterGroup is in a state of losing connection, this exception is thrown due to lack of resources or when the method cannot be called by the section filter;

org.davic.mpeg.sections.IllegalFilterDefinitionException – If the pid or tableID parameter is negative or out of the range defined by MPEG-2, this exception is thrown.

B.4.4.1.6 startFiltering

Prototype: public boolean startFiltering(java.lang.Object appData, int pid, int tableID, byte[] posFilterDef, byte[] posFilterMask) throws org.davic.mpeg.sections.FilterResourceException, org.davic.mpeg.sections.IllegalFilterDefinitionException, org.davic.mpeg.NotAuthorizedException, org.davic.mpeg.sections.ConnectionLostException

Description: Start section filtering, filter out the section (section) carried in the transport packet specified by the parameter pid, and the table ID matches the tableID parameter, the section data matches the given byte filter, and the first byte of the filter byte array corresponds to the third byte of the section data. The parent filter group org.davic.mpeg.sections.SectionFilterGroup object must first be associated with the transport stream. If the following conditions are met, it means that the section data matches the given byte filter:

(posFilterDef[i] & posFilterMask[i]) == (section data [3+i] & posFilterMask[i])

Parameter: appData – A java.lang.Object object, indicating the additional data object provided by the application, as a part of all the filter event org.davic.mpeg.sections.SectionFilterEvent objects generated by this method call, the object will be passed to the registered section filter event listener. This object can be used for internal communication between applications. If the application does not require any additional data, this parameter can be set to null;

pid – Int type, indicating the transport stream packet identifier (TS_PID) to be filtered;

tableID – Int type, indicating the table ID (table_id) to be filtered;

posFilterDef – A byte array, indicating the value of bit matching in the section data;

posFilterMask – A byte array, indicating the mask of bit matching in the section data, that is, only the bits with the mask value of '1' are compared.

Return: Boolean type, returning true upon success and false upon failure.

Exception: org.davic.mpeg.sections.FilterResourceException – When this method is called, if the total number of section filters started by the parent filter group org.davic.mpeg.sections.SectionFilterGroup is equal to the number of section filters occupied when the filter group is created, this exception is thrown. This exception applies regardless of whether the parent filter group is associated with the TS stream;

org.davic.mpeg.sections.IllegalFilterDefinitionException – If an illegal filter is defined, or the length of the parameter posFilterDef array and the posFilterMask array are inconsistent, or the length exceeds the system's ability to filter, this exception is thrown;

org.davic.mpeg.NotAuthorizedException – If the section to be filtered is scrambled and has no descrambling permission, this exception is thrown;

org.davic.mpeg.sections.ConnectionLostException – This exception is thrown if the parent filter group org.davic.mpeg.sections.SectionFilterGroup is in a state of losing connection due to lack of resources or when the section filter cannot complete the method call.

B.4.4.1.7 startFiltering

Prototype: public boolean startFiltering(java.lang.Object appData, int pid, int tableID,

byte[] posFilterDef, byte[] posFilterMask,

byte[] negFilterDef, byte[] negFilterMask)

throws org.davic.mpeg.sections.FilterResourceException, org.davic.mpeg.sections.IllegalFilterDefinitionException,

org.davic.mpeg.NotAuthorizedException, org.davic.mpeg.sections.ConnectionLostException

Description: Start section filtering, filter out the section (section) carried in the transport packet specified by the pid parameter, and the table ID matches the tableID parameter, the section data matches the given byte filter, and the first byte of the filter byte array corresponds to the third byte of the section data. The parent filter group org.davic.mpeg.sections.SectionFilterGroup object must first be associated with the transport stream. If the following conditions are met, it means that the section data matches the given byte filter:

((posFilterDef[i] & posFilterMask[i]) == (section data [3+i] & posFilterMask[i])) &&

((negFilterDef[i] & negFilterMask[i]) != (section data [3+i] & negFilterMask[i]))

Parameter: appData – A java.lang.Object object, indicating the additional data object provided by the application, as a part of all filter event SectionFilterEvent objects generated by this method call, the object will be passed to the registered section filter event Listener, the application can use this object for internal communication between applications. If the application does not require any additional data, this parameter can be set to null;

pid – Int type, indicating the transport stream packet identifier (TS_PID) to be filtered;

tableID – Int type, indicating the table ID (table_id) to be filtered;

posFilterDef – A byte array, indicating the value of bit matching in the section data;

posFilterMask – A byte array, indicating the mask of bit matching in the section data, that is, only the bits with the mask value of '1' are compared;

negFilterDef – A byte array, indicating the value of bit matching in the section data;

negFilterMask – A byte array, indicating the mask of bit matching in the section data, that is, only the bits with the mask value of '1' are compared;

Return: boolean type, returning true upon success and false upon failure.

Exception: org.davic.mpeg.sections.FilterResourceException – When this method is called, if the total number of section filters started by the parent filter group org.davic.mpeg.sections.SectionFilterGroup is equal to the number of section filters occupied when the filter group is created, this exception is thrown. This exception applies regardless of whether the parent section filter group is associated with the TS stream;

org.davic.mpeg.sections.IllegalFilterDefinitionException – If an illegal filter is defined, or the lengths of the parameter posFilterDef array and the posFilterMask array are inconsistent, or the length exceeds the system's ability to filter, this exception is thrown;

org.davic.mpeg.NotAuthorizedException – If the section to be filtered is scrambled and has no descrambling permission, this exception is thrown;

org.davic.mpeg.sections.ConnectionLostException – This exception is thrown if the parent filter group org.davic.mpeg.sections.SectionFilterGroup is in a state of losing connection due to lack of resources or when the section filter cannot complete the method call.

B.4.4.1.8 startFiltering

Prototype: public boolean startFiltering(java.lang.Object appData, int pid, int tableID,

int offset, byte[] posFilterDef, byte[] posFilterMask)

throws org.davic.mpeg.sections.FilterResourceException, org.davic.mpeg.sections.IllegalFilterDefinitionException,

org.davic.mpeg.NotAuthorizedException, org.davic.mpeg.sections.ConnectionLostException

Description: Start the section filtering, filter out the section (section) carried in the transport packet specified by the pid parameter, and the table ID matches the tableID parameter, the section data matches the given byte filter, and the offset of section data corresponding to the first byte of the filter byte array is specified by the parameter offset. The parent filter group org.davic.mpeg.sections.SectionFilterGroup object must first be associated with the transport stream. If the following conditions are met, it means that the section data matches the given byte filter:

(posFilterDef[i] & posFilterMask[i]) == (section data [offset+i] & posFilterMask[i])

Parameter: appData – A java.lang.Object object, indicating the additional data object provided by the application, as a part of all the filter events org.davic.mpeg.sections.SectionFilterEvent objects generated by this method call, the object will be passed to the registered section filter event listener, and the application can use this object for internal communication between applications. If the application does not require any additional data, this parameter can be set to null;

pid – Int type, indicating the transport stream packet identifier (TS_PID) to be filtered;

tableID – Int type, indicating the table ID (table_id) to be filtered;

offset – Int type, indicating the offset of the section data corresponding to the first byte of the filter byte array, the value is greater than or equal to 3 and less than 31;

posFilterDef – A byte array, indicating the value of bit matching in the section data;

posFilterMask – A byte array, indicating the mask of bit matching in the section data, that is, only the bits with the mask value of '1' are compared.

Return: boolean type, returning true upon success and false upon failure.

Exception: org.davic.mpeg.sections.FilterResourceException – When this method is called, if the total number of section filters started by the parent filter group org.davic.mpeg.sections.SectionFilterGroup is equal to the number of section filters occupied when the filter group is created, this exception is thrown. This exception applies regardless of whether the parent section filter group is associated with the TS stream;

org.davic.mpeg.sections.IllegalFilterDefinitionException – If an illegal filter is defined, or the lengths of the parameter posFilterDef array and the posFilterMask array are inconsistent, or the length exceeds the system's ability to filter, this exception is thrown;

org.davic.mpeg.NotAuthorizedException – If the section to be filtered is scrambled and has no descrambling permission, this exception is thrown;

org.davic.mpeg.sections.ConnectionLostException – This exception is thrown if the parent filter group org.davic.mpeg.sections.SectionFilterGroup is in a state of losing connection due to lack of resources or when the section filter cannot complete the method call.

B.4.4.1.9 startFiltering

Prototype: public boolean startFiltering(java.lang.Object appData,

int pid, int tableID, int offset,

byte[] posFilterDef, byte[] posFilterMask,

byte[] negFilterDef, byte[] negFilterMask)

throws org.davic.mpeg.sections.FilterResourceException, org.davic.mpeg.sections.IllegalFilterDefinitionException,

org.davic.mpeg.NotAuthorizedException, org.davic.mpeg.sections.ConnectionLostException

Description: Start section filtering, filter out the section (section) carried in the transport packet specified by the pid parameter, and the table ID matches the tableID parameter, the section data matches the given byte filter, and the offset of the section data corresponding to the first byte of the filter byte array is specified by the parameter offset. The parent filter group org.davic.mpeg.sections.SectionFilterGroup object must first be associated with the transport stream. If the following conditions are met, it means that the section data matches the given byte filter:

((posFilterDef[i] & posFilterMask[i]) == (section data [offset+i] & posFilterMask[i])) &&

((negFilterDef[i] & negFilterMask[i]) != (section data [offset+i] & negFilterMask[i]))

Parameter: appData – A java.lang.Object object, indicating additional data objects provided by the application, as a part of all the filter events org.davic.mpeg.sections.SectionFilterEvent objects generated by this method call, this object will be passed to the registered section filter event listener, and the application can use this object for internal communications between applications.

If the application does not require any additional data, this parameter can be set to null;

pid – Int type, indicating the transport stream packet identifier (TS_PID) to be filtered;

tableID – Int type, indicating the table ID (table_id) to be filtered;

offset – Int type, indicating the offset of the section data corresponding to the first byte of the filter byte array, the value is greater than or equal to 3 and less than 31;

posFilterDef – A byte array, indicating the value of bit matching in the section data;

posFilterMask – A byte array, indicating the mask of bit matching in the section data, that is, only the bits with the mask value of '1' are compared;

negFilterDef – A byte array, indicating the value of bit matching in the section data;

negFilterMask – A byte array, indicating the mask of bit matching in the section data, that is, only the bits with the mask value of '1' are compared;

Return: boolean type, returning true upon success and false upon failure.

Exception: org.davic.mpeg.sections.FilterResourceException – When calling this method, if the total number of section filters started by the parent filter group org.davic.mpeg.sections.SectionFilterGroup is equal to the number of section filters occupied when the filter group is created, this exception is thrown. This exception applies regardless of whether the parent filter group is associated with the TS stream;

org.davic.mpeg.sections.IllegalFilterDefinitionException – If an illegal filter is defined, or the lengths of the parameter posFilterDef array and the posFilterMask array are inconsistent, or the length exceeds the system's ability to filter, this exception is thrown;

org.davic.mpeg.NotAuthorizedException – If the section to be filtered is scrambled and has no descrambling permission, this exception is thrown;

org.davic.mpeg.sections.ConnectionLostException – This exception is thrown if the parent filter group org.davic.mpeg.sections.SectionFilterGroup is in a state of losing connection due to lack of resources or when the section filter cannot complete the method call.

B.4.4.1.10 stopFiltering

Prototype: public boolean stopFiltering()

Description: Stop section filtering. If the parent filter group org.davic.mpeg.sections.SectionFilterGroup object is connected to a transport stream, the section filtering that matches this filter object will stop. When using filters and filter groups, the order of function calling is: filter group attach, filter startFiltering, filter stopFiltering, filter group detach. If the filter has been stopped, calling stopFiltering again will directly return false.

Parameter: boolean type, returning true upon success and false upon failure.

Return: None.

B.4.5 Class org.davic.mpeg.sections.SimpleSectionFilter

Prototype: public class org.davic.mpeg.sections.SimpleSectionFilter extends org.davic.mpeg.sections.SectionFilter

Description: Simple section filter class, inherits org.davic.mpeg.sections.SectionFilter class. Simple section filter: Only used once. When a section that matches the specified filter condition is found, the simple section filter stops filtering, as if the SectionFilter.stopFiltering() method was called.

B.4.5.1 Method

B.4.5.1.1 getSection

Prototype: public Section getSection() throws

org.davic.mpeg.sections.FilteringInterruptedException

Description: Getting the section data filtered by this filter. The section obtained by this method describes the last MPEG-2 section that satisfies the filter condition. If the simple filter is filtering, this method is blocked until the end of the filtering. If the startFiltering() method is not called again during this period, and this method is called repeatedly, the same section object will be returned. Every time a new filtering operation is started, a new section object is generated. The old section object will be deleted unless the application maintains the old section object itself. All operations that access the previous section object will throw the Exception org.davic.mpeg.sections.NoDataAvailableException.

Parameter: None.

Return: None.

Exception: org.davic.mpeg.sections.FilteringInterruptedException – If the filtering operation stops before the matching section is found, this exception is thrown.

B.4.6 Class org.davic.mpeg.sections.TableSectionFilter

Prototype: public class org.davic.mpeg.sections.TableSectionFilter extends org.davic.mpeg.sections.SectionFilter

Description: Table section filter class, inherits org.davic.mpeg.sections.SectionFilter class. Table section filter: Receive complete tables with minimal application intervention. When the table section filter is activated, it will filter the first section that meets the given filter conditions. Once the section is found, the last_section_number field will be used to determine the number of section objects required to form the entire table. After the number of section objects is determined, the filter is restarted to receive all sections in the table. Each time a section is received, an org.davic.mpeg.sections.SectionAvailableEvent event will be generated. After receiving the entire table, the end of filtering org.davic.mpeg.sections.EndOfFilteringEvent event will be generated. The version numbers of all sections in the table should be the same. If the version number of a filtered section is different from the first section, an org.davic.mpeg.sections.VersionChangeDetectedEvent event will be generated, the newly captured section will be ignored, and the original version of the section will continue to be filtered.

Note that when setting segment filtering parameters: If there are too many constraints, the filter cannot stop automatically; and if the constraints are too wide, the filter will produce inconsistent results (for example: use org.davic.mpeg.sections.TableSectionFilter to filter some sections). When the API finds that the filter parameters are not fully defined, which may result in filter blocking or MPEG-2 incompatibility, the org.davic.mpeg.sections.IncompleteFilteringEvent event will be generated to stop the filtering.

B.4.6.1 Method

B.4.6.1.1 getSections

Prototype: public Section[] getSections() throws org.davic.mpeg.sections.FilteringInterruptedException

Description: Getting all the section data of the table filtered by this filter. This method will return an array of all section objects of the table. The array elements are arranged according to the section number section_number. If some sections have not been filtered until this method is called, the corresponding record in the array is set to null. If no sections are filtered out, this method is blocked until at least one section is obtained or the filtering ends. If the startFiltering() method is not called again in the meantime, and this method is called repeatedly, the same array will be returned. Every time a new filtering operation is started, an array of section objects is created. The old section array object will be deleted unless the application maintains the old section array object. All operations that access the previous section array object will throw exception org.davic.mpeg.sections.NoDataAvailableException.

Parameter: None.

Return: None.

Exception: org.davic.mpeg.sections.FilteringInterruptedException – If the filtering operation stops before the matching section is found, this exception is thrown.

B.4.7 Class org.davic.mpeg.sections.RingSectionFilter

Prototype: public class org.davic.mpeg.sections.RingSectionFilter extends org.davic.mpeg.sections.SectionFilter

Description: Ring section filter class, inherits org.davic.mpeg.sections.SectionFilter class.

Ring section filter class: Once started, there is no need to stop and reset, it is used to capture continuous MPEG-2 section data. An org.davic.mpeg.sections.RingSectionFilter object has a predetermined number of section objects. The section data captured one after another will be loaded into these section objects in turn. Filtering will continue when there are empty section objects. If the application wants to filter continuously, it needs to use the setEmpty method of the Section object to mark the section object as empty before the new object arrives. If the filtering operation encounters a non-empty section object, it will stop. Each time startFiltering is called, the section will be filtered from the beginning of the array. When the ring section filter is created for the first time, all sections are set to empty, after which the application will clear them. Starting the ring section filter will no longer clear the section.

B.4.7.1 Method

B.4.7.1.1 getSections

Prototype: public org.davic.mpeg.sections.Section[] getSections()

Description: Getting all section data filtered by this filter.

Parameter: None.

Return: None.

Return: An Org.davic.mpeg.sections.Section object array, indicating the section data filtered by this filter.

Org.davic.mpeg.sections.Section object array is always full of data, and the application is responsible for checking which data is valid.

Before the new filter data arrives, calling this method repeatedly will get the same result.

B.4.8 Event org.davic.mpeg.sections.SectionFilterEvent

Prototype: public class org.davic.mpeg.sections.SectionFilterEvent

extends java.util.EventObject

Description: Section filtering events, a group of base classes of section filtering events defined by this package.

B.4.8.1 Method

B.4.8.1.1 getSource

Prototype: public java.lang.Object getSource()

Description: Getting the SectionFilter object that generated this event.

Rewrite: A getSource() method of the java.util.EventObject class.

Parameter: None.

Return: An org.davic.mpeg.sections.SectionFilter object, indicating the source for generating this event.

B.4.8.1.2 getAppData

Prototype: public java.lang.Object getAppData()

Description: Getting the application data passed through the startFiltering() method of the SectionFilter object.

Parameter: None.

Return: A Java.lang.Object object, indicating additional application data.

B.4.9 Event org.davic.mpeg.sections.SectionAvailableEvent

Prototype: public class org.davic.mpeg.sections.SectionAvailableEvent

extends org.davic.mpeg.sections.SectionFilterEvent

Description: Section data available event, inherits the org.davic.mpeg.sections.SectionFilterEvent class, reports filtering to a complete section. When a section that meets the filter conditions is successfully filtered from the transport stream, the event is generated by the org.davic.mpeg.sections.SimpleSectionFilter, org.davic.mpeg.sections.TableSectionFilter or org.davic.mpeg.sections.RingSectionFilter object.

B.4.10 Event org.davic.mpeg.sections.VersionChangeDetectedEvent

Prototype: public class org.davic.mpeg.sections.VersionChangeDetectedEvent

extends org.davic.mpeg.sections.SectionFilterEvent

Description: A section filter version change event, inherits the org.davic.mpeg.sections.SectionFilterEvent class. This event is generated by org.davic.mpeg.sections.TableSectionFilter, reporting that the received section is inconsistent with the previous section version. This event can only be generated once per filtering operation. Sections with different versions will be discarded.

B.4.10.1 Method

B.4.10.1.1 getOriginalVersion

Prototype: public int getOriginalVersion()

Description: Getting the old version number of the section data.

Parameter: None.

Return: Int type, indicating the old version number of the section data.

B.4.10.1.2 getNewVersion

Prototype: public int getNewVersion()

Description: Getting the new version number of the section data.

Parameter: None.

Return: Int type, indicating the new version number of the section data.

B.4.11 Event org.davic.mpeg.sections.EndOfFilteringEvent

Prototype: public class org.davic.mpeg.sections.EndOfFilteringEvent extends org.davic.mpeg.sections.SectionFilterEvent

Description: Section filtering end event, inherits the org.davic.mpeg.sections.SectionFilterEvent class, reports the end of section filtering. When the filtering ends, the event is generated by the org.davic.mpeg.sections.RingSectionFilter and org.davic.mpeg.sections.TableSectionFilter objects. This event is not generated when the org.davic.mpeg.sections.SimpleSectionFilter filtering ends.

B.4.12 Event org.davic.mpeg.sections.IncompleteFilteringEvent

Prototype: public class org.davic.mpeg.sections.IncompleteFilteringEvent extends org.davic.mpeg.sections.EndOfFilteringEvent

Description: Section filtering incomplete event, inherits org.davic.mpeg.sections.EndOfFilteringEvent class. It is used to report the end of the filtering operation generated by org.davic.mpeg.sections.TableSectionFilter. This event is generated when the API finds that the section parameters are not fully defined, which will result in blocking the filter or incompatibility with MPEG-2.

B.4.13 Event org.davic.mpeg.sections.TimeOutEvent

Prototype: public class org.davic.mpeg.sections.TimeOutEvent extends org.davic.mpeg.sections.EndOfFilteringEvent

Description: Section filtering timeout event, inherits org.davic.mpeg.sections.EndOfFilteringEvent class. This event is generated when the section filtering operation times out.

– For org.davic.mpeg.sections.SimpleSectionFilter, this event will be generated if the section data is not filtered within the specified time;

– For org.davic.mpeg.sections.TableSectionFilter, this event is generated if the entire table is not filtered within the specified time;

– For org.davic.mpeg.sections.RingSectionFilter, if the specified time is exceeded after the last section is successfully filtered, this event is generated.

B.4.14 Event org.davic.mpeg.sections.FilterResourcesAvailableEvent

Prototype: public class org.davic.mpeg.sections.FilterResourcesAvailableEvent extends org.davic.resources.ResourceStatusEvent

Description: Filter resource available event, inherits the org.davic.resources.ResourceStatusEvent class. This event indicates that there are sufficient section filter resources for the section filter group. For example: if a section filter group is created with 4 filters, at least 4 filters are available when the event occurs. Note that they may no mlonger be available when the application tries to connect the section filter group again. Therefore, this event is a useful reminder for applications that try to connect to the section filter group again. This event only occurs after the org.davic.mpeg.sections.ForcedDisconnectedEvent event and before the application successfully connects to the section filter group again.

B.4.15 Event org.davic.mpeg.sections.ForcedDisconnectedEvent

Prototype: public class org.davic.mpeg.sections.ForcedDisconnectedEvent

extends org.davic.resources.ResourceStatusEvent

Description: Forced disconnected event between the section filter group and the transport stream inherits the org.davic.resources.ResourceStatusEvent class. Report that the previously available transport stream is no longer available, or the section filter resource has been removed from the connected section filter group. In the second case, in addition to generation of this event, a notifyRelease() method of org.davic.resources.ResourceClient is also called at the same time.

B.4.15.1 Method

B.4.15.1.1 getSource

Prototype: public java.lang.Object getSource()

Description: Getting org.davic.mpeg.sections.SectionFilterGroup object that generated the event.

Rewrite: GetSource() method of the ResourceStatusEvent class.

Return: A SectionFilterGroup object, indicating the section filter group that generated the event.

B.4.16 Exception org.davic.mpeg.sections.SectionFilterException

Prototype: public class org.davic.mpeg.sections.SectionFilterException extends java.lang.Exception

Description: Section filter exception, a base class of a group of section filter exception defined by this package.

B.4.17 Exception org.davic.mpeg.sections.ConnectionLostException

Prototype: public class org.davic.mpeg.sections.ConnectionLostException extends org.davic.mpeg.sections.SectionFilterException

Description: Connection loss exception, inherits org.davic.mpeg.sections.SectionFilterException class. This exception indicates that org.davic.mpeg.sections.SectionFilterGroup has lost connection or resources and cannot be called to the startFiltering() method. It is only generated by the SectionFilterGroup object in the lost connection state.

B.4.18 Exception org.davic.mpeg.sections.FilteringInterruptedException

Prototype: public class org.davic.mpeg.sections.FilteringInterruptedException extends org.davic.mpeg.sections.SectionFilterException

Description: Filtering interrupted exception, inherits org.davic.mpeg.sections.SectionFilterException class. This exception indicates that the filtering operation was interrupted before the data was filtered.

B.4.19 Exception org.davic.mpeg.sections.FilterResourceException

Prototype: public class org.davic.mpeg.sections.FilterResourceException extends org.davic.mpeg.sections.SectionFilterException

Description: Filter resource exception, inherits org.davic.mpeg.sections.SectionFilterException class. This exception indicates that when org.davic.mpeg.sections.SectionFilterGroup is in a connected or disconnected state, there are insufficient resources to complete the operation.

B.4.20 Exception org.davic.mpeg.sections.IllegalFilterDefinitionException

Prototype: public class org.davic.mpeg.sections.IllegalFilterDefinitionException extends org.davic.mpeg.sections.SectionFilterException

Description: Illegal filter definition exception, inherits S org.davic.mpeg.sections.ectionFilterException class. This exception indicates that the filter definition of the filter is invalid.

B.4.21 Exception org.davic.mpeg.sections.InvalidSourceException

Prototype: public class org.davic.mpeg.sections.InvalidSourceException extends org.davic.mpeg.sections.SectionFilterException

Description: Section data source invalid exception, inherits org.davic.mpeg.sections.SectionFilterException class.

B.4.22 Exception org.davic.mpeg.sections.NoDataAvailableException

Prototype: public class org.davic.mpeg.sections.NoDataAvailableException extends org.davic.mpeg.sections.SectionFilterException

Description: Section object no available data exception, inherits org.davic.mpeg.sections.SectionFilterException class.

B.5 URL package module

URL package module provides URL package reference method.

The summary of URL package module is shown in Table B.4.

Table B.4 – Summary of URL package module

Class	
Locator	Resource locator class, packages the URL into a locator object.
Exception	
InvalidLocatorException	Invalid locator exception.

B.5.1 Class org.davic.net.Locator

Prototype: public class org.davic.net.Locator

Description: Resource locator class, packages the URL into a locator object.

B.5.1.1 Method

B.5.1.1.1 Locator

Prototype: public Locator(java.lang.String url)

Description: Construction method creates a resource locator object.

Parameter: url – A java.lang.String object, indicating a URL string.

B.5.1.1.2 hasMultipleTransformations

Prototype: public boolean hasMultipleTransformations()

Description: Indicate whether this locator is mapped to multiple transport-related locator formats.

Parameter: None.

Return: Boolean type, true indicating this locator is mapped to multiple transport-related locator formats, false indicating No.

B.5.1.1.3 toExternalForm

Prototype: public abstract java.lang.String toExternalForm()

Description: Getting the URL string corresponding to the locator. If a non-empty but invalid URL is used to create a locator instance, the behavior of the system depends on the specific implementation.

Parameter: None.

Return: A java.lang.String object, indicating the URL string corresponding to this locator.

B.5.1.1.4 toString

Prototype: public java.lang.String toString()

Description: Getting URL string.

Rewrite: A toString() method of the java.lang.Object class.

Parameter: None.

Return: A java.lang.String object, indicating a URL string.

B.5.1.1.5 equals

Prototype: public boolean equals(Object obj)

Description: Determine whether the obj object is the same as this example.

Rewrite: An equals() method of the Object class.

Parameter: An Obj-Locator object, indicating the locator to be compared.

Return: Boolean type, true value indicating that the obj object is the same as this instance, and false indicating that it is not the same.

B.5.2 Exception org.davic.net.InvalidLocatorException

Prototype: public class org.davic.net.InvalidLocatorException extends java.lang.Exception

Description: Invalid locator exception. This exception is thrown when one or more parameters of the Locator object are invalid.

B.6 DVB locator module

DVB locator module provides a reference method for accessing DVB broadcasting services and their contents.

The summary of DVB locator module is shown in Table B.5.

Table B.5 – Summary of DVB locator module

Class	
DvbLocator	DVB locator class, packages URL in DVB format into locator object.
DvbNetworkBoundLocator	DVB locator class bound to the network, this type of object uniquely identifies a given entity and transport system.

B.6.1 Class org.davic.net.dvb.DvbLocator

Prototype: public class org.davic.net.dvb.DvbLocator extends org.davic.net.Locator

Description: DVB locator class, packages URL in DVB format into locator object.

B.6.1.1 Method

B.6.1.1.1 DvbLocator

Prototype: public DvbLocator(int onid, int tsid) throws Org.davic.net.InvalidLocatorException

Description: Construction method creates a DVB locator object in the URL format "dvb://original_network_id.transport_stream_id".

Parameter: onid – Int type, indicating an original network identifier (original_network_id);

tsid – Int type, indicating a transport stream identifier (transport_stream_id).

Exception: org.davic.net.InvalidLocatorException – If the parameter cannot identify a valid locator (for example, the value identifier is out of range), this exception is thrown.

B.6.1.1.2 DvbLocator

Prototype: public DvbLocator(int onid, int tsid, int serviceid) throws org.davic.net.InvalidLocatorException

Description: Construction method creates a DVB locator object in the URL format "dvb://original_network_id.transport_stream_id.service_id".

Parameter: onid – Int type, indicating an original network identifier (original_network_id);

tsid – Int type, indicating a transport stream identifier (transport_stream_id). If the value is -1, it means that the locator does not contain the transport stream identifier (transport_stream_id);

serviceid – Int type, indicating a service identifier (service_id).

Exception: org.davic.net.InvalidLocatorException – If the parameter cannot identify a valid locator (for example, the value identifier is out of range), this exception is thrown.

B.6.1.1.3 DvbLocator

Prototype: public DvbLocator(int onid, int tsid, int serviceid, int eventid)

throws org.davic.net.InvalidLocatorException

Description: Construction method creates a DVB locator object in the URL format "dvb://original_network_id.transport_stream_id.service_id;event_id".

Parameter: onid – Int type, indicating an original network identifier (original_network_id);

tsid – Int type, indicating a transport stream identifier (transport_stream_id). If the value is -1, it means that the locator does not contain the transport stream identifier (transport_stream_id);

serviceid – Int type, indicating a service identifier (service_id);

eventid – Int type, indicating an event identifier (event_id).

Exception: org.davic.net.InvalidLocatorException – If the parameter cannot identify a valid locator (for example, the value identifier is out of range), this exception is thrown.

B.6.1.1.4 DvbLocator

Prototype: public DvbLocator(int onid, int tsid, int serviceid, int eventid, int componenttag)

throws org.davic.net.InvalidLocatorException

Description: Construction method creates a DVB locator object in the "dvb://original_network_id.transport_stream_id.service_id.

component_tag;eventid" or "dvb://original_network_id.transport_stream_id.

service_id.component_tag" URL format.

Parameter: onid – Int type, indicating an original network identifier (original_network_id);

tsid – Int type, indicating a transport stream identifier (transport_stream_id).

If the value is -1, it means that the locator does not contain the transport stream identifier (transport_stream_id);

serviceid – Int type, indicating a service identifier (service_id);

eventid – Int type, indicating an event identifier (event_id). If the value is -1, it means that the locator does not contain the event identifier (event_id);

componenttag – Int type, indicating an elementary stream component tag (component_tag).

Exception: org.davic.net.InvalidLocatorException – If the parameter cannot identify a valid locator (for example, the value identifier is out of range), this exception is thrown.

B.6.1.1.5 DvbLocator

Prototype: public DvbLocator(int onid, int tsid, int serviceid,

int eventid, int[] componenttags)

throws org.davic.net.InvalidLocatorException

Description: Construction method creates a DVB locator object in the "dvb://original_network_id.transport_stream_id.service_id.component_

tag{&component_tag};event_id" or "dvb://original_network_id.transport_stream_id.

service_id.component_tag{&component_tag}" URL format.

Parameter: onid-Int type, indicating an original network identifier (original_network_id);

tsid-Int type, indicating a transport stream identifier (transport_stream_id). If the value is -1, it means that the locator does not contain the transport stream identifier (transport_stream_id);

serviceid – Int type, indicating a service identifier (service_id);

eventid – Int type, indicating an event identifier (event_id). If the value is -1, it means that the locator does not contain the event identifier (event_id);

componenttags – An int type array, indicating an array of elementary stream component tags (component_tag).

Exception: org.davic.net.InvalidLocatorException – If the parameter cannot identify a valid locator (for example, the value identifier is out of range), this exception is thrown.

B.6.1.1.6 DvbLocator

Prototype: public DvbLocator(int onid, int tsid, int serviceid, int eventid,

int[] componenttags, java.lang.String filePath)

throws org.davic.net.InvalidLocatorException

Description: Construction method creates a DVB locator object in the "dvb://original_network_id.transport_stream_id.service_id.component_

tag{&component_tag};event_id/filepath" or "dvb://original_network_id.transport_

stream_id.service_id.component_tag{&component_tag}/filepath" URL format.

Parameter: onid – Int type, indicating an original network identifier (original_network_id);

tsid – Int type, indicating a transport stream identifier (transport_stream_id). If the value is -1, it means that the locator does not contain the transport stream identifier (transport_stream_id);

serviceid – Int type, indicating a service identifier (service_id);

eventid – Int type, indicating an event identifier (event_id). If the value is -1, it means that the locator does not contain the event identifier (event_id);

componenttags – An int type array, indicating an array of elementary stream component tags (component_tag);

filePath – A java.lang.String object, indicating the file path string, including the initial slash.

Exception: org.davic.net.InvalidLocatorException – If the parameter cannot identify a valid locator (for example, the value identifier is out of range), this exception is thrown.

B.6.1.1.7 DvbLocator

Prototype: public DvbLocator(java.lang.String url) throws org.davic.net.InvalidLocatorException

Description: Construction method creates a DVB locator object based on the input string.

Parameter: url – A java.lang.String object, indicating a DVB URL string.

Exception: org.davic.net.InvalidLocatorException – If the parameter cannot identify a valid locator (for example: the numerical identifier is out of range, does not conform to the DVB URL format, etc.), this exception is thrown.

B.6.1.1.8 getComponentTags

Prototype: public int[] getComponentTags()

Description: Obtain an elementary stream component tag array information contained in the DVB locator.

Parameter: None.

Return: An int type array, indicating the array of elementary stream component tags, if the locator does not contain component tag information, the length of the array is 0.

B.6.1.1.9 getEventId

Prototype: public int getEventId()

Description: Getting the event identifier information contained in the DVB locator.

Parameter: None.

Return: Int type, indicating an event identifier, if not, it returns -1.

B.6.1.1.10 getFilePath

Prototype: public java.lang.String getFilePath()

Description: Getting the file name and path information contained in the DVB locator.

Parameter: None.

Return: A Java. Lang. string object, indicating a path string, including the beginning slash. If the locator has no path information, null is returned.

B.6.1.1.11 getOriginalNetworkId

Prototype: public int getOriginalNetworkId()

Description: Getting an original network identifier information contained in the DVB locator.

Parameter: None.

Return: int type, indicating an original network identifier, if not, -1 is returned.

B.6.1.1.12 getTransportStreamId

Prototype: public int getTransportStreamId()

Description: Getting a transport stream identifier information contained in the DVB locator.

Parameter: None.

Return: int type, indicating a transport stream identifier, if not, -1 is returned.

B.6.1.1.13 getServiceId

Prototype: public int getServiceId()

Description: Getting the service identifier information contained in DVB locator.

Parameter: None.

Return: int type, indicating a service identifier, if not, -1 is returned.

B.6.1.1.14 toExternalForm

Prototype: public java.lang.String toExternalForm()

Description: Getting the URL string corresponding to the locator. If a non-empty but invalid URL is used to create an instance of the locator, the behavior of the system depends on the specific implementation. Implement an toexternalform () method of the locator class.

Parameter: None.

Return: A java.lang.String object, indicating an URL string corresponding to this locator.

B.6.2 Class org.davic.net.dvb.DvbNetworkBoundLocator

Prototype: public class org.davic.net.dvb.DvbNetworkBoundLocator extends org.davic.net.DvbLocator

Description: DVB locator class bound to network, uniquely identifies a given entity and transport system. For example, a service may be transported in satellite and ground network, dvblocor object may be the same, but dvbnetworkboundlocator object is different.

B.6.2.1 Method

B.6.2.1.1 DvbNetworkBoundLocator

Prototype: public DvbNetworkBoundLocator(org.davic.net.dvb.DvbLocator unboundLocator, int networkId)

throws org.davic.net.InvalidLocatorException

Description: Construct method creates a DVB locator object bound to the network.

Parameter: unboundLocator – An org.david.net.dvb.dvblocator object, indicating a DVB locator not bound with the transport network;

networkId – int type, indicating network identifier.

Exception: org.davic.net.InvalidLocatorException – If the parameter cannot identify a valid locator (for example, the numeric identifier is out of range), this exception is thrown.

B.6.2.1.2 getNetworkId

Prototype: public int getNetworkId()

Description: Getting a network identifier.

Parameter: None.

Return: int type, indicating network identifier.

B.7 Broadcast protocol processing module

The broadcast protocol processing module defines the java interface related to DVB broadcast protocol processing.

See Table B.6 for the summary of broadcast protocol processing module.

Table B.6 – Summary of the broadcast protocol processing module

Interface	
SICommonInformation	A PSI / Si public information interface, providing a method to obtain psi / Si public features.
SINetwork	A network information interface, providing a method to obtain a network information. Each SINetwork object is uniquely identified by the network_ID.
SIBouquet	A service group information interface, providing a method to obtain the information of the service group (bouquet). Each SIBouquet object is uniquely identified by the network_id, bouquet_id.

Table B.6 – Summary of the broadcast protocol processing module

SIService	A service information interface, providing a method to obtain the service information. Each SIService object is uniquely identified by the network_id, original_network_id, transport_stream_id and service_id.
SIEvent	A program event information interface, providing a method to obtain the event information. Each SIEvent object is common uniquely identified by network_ id, original_network_id, transport_stream_id, service_id and event_id.
SITransportStream	A transport stream information interface, providing a method to obtain transport stream information, each SITransportStream object is common uniquely identified by network_id, original_network_id and transport_stream_id.
SIElementaryStream	An elementary stream information interface, providing a method to obtain elementary stream information, each SIElementaryStream object is common uniquely identified by network_id, original_network_id, transport_stream_id, service_id and component_tag(or elementary_PID).
SITime	A time information interface, providing a method to obtain time information. The time information is obtained from TDT or TOT, each SITime object is uniquely identified by network_id.
SIDescriptor	A Descriptor information interface, providing a method related to descriptor access.
SIRequest	PSI / Si information request interface, describes a PSI / SI information retrieval request generated by the application, the application can cancel the request through this object.
SIRetrieveListener	SI information acquisition event listener is implemented by application program.
SIUpdateListener	PSI / SI table update event listener is implemented by application program.
SIDescriptorTag	Descriptors tag constant definition interface, values see GB / T 28161-2011.
SIRunningStatus	For the definition interface of running state constant of broadcasting service or event, please refer to GB / T 28161-2011.
SIServiceType	Service type constant definition interface, see GB / T 28161-2011 for values.
SIStreamType	For the definition interface of stream type constant, see GB / T 17975.1-2010 for values.
Class	
SIDatabase	A PSI / SI information database, providing a management and operation method of PSI / SI information database in DVB mode. It is the entry class for application to obtain PSI / SI information.
SIRequestFailureType	The reason why the PSI / SI information request failed.
Event	
SIRetrieveEvent	PSI / SI information request event is the base class of a group of events related to PSI / SI information request defined in this package. One PSI / SI information request will only generate one such event.
SISuccessRetrieveEvent	PSI / SI information request success event inherits SIRetrieveEvent class.
SIFailureRetrieveEvent	PSI / SI information request failure event inherits SIRetrieveEvent class.
SIUpdateEvent	PSI/SI table update event.
Exception	
InvalidPeriodException	This exception is thrown when the specified date is invalid.

B.7.1 Interface org.ngb.broadcast.dvb.si.SICommonInformation

Prototype: public interface org.ngb.broadcast.dvb.si.SICommonInformation

Description: A PSI / SI public information interface, providing a method to obtain PSI / SI public features.

B.7.1.1 Constant field – PSI / SI object type

B.7.1.1.1 SI_BOUQUET

Prototype: public static final int SI_BOUQUET = 0

Description: SI information identification – Service group information.

B.7.1.1.2 SI_NETWORK

Prototype: public static final int SI_NETWORK = 1

Description: SI information identification – Network information.

B.7.1.1.3 SI_SERVICE

Prototype: public static final int SI_SERVICE = 2

Description: SI information identification – Service information.

B.7.1.1.4 SI_TS

Prototype: public static final int SI_TS = 3

Description: SI information identification – Transport stream (TS) information.

B.7.1.1.5 SI_ES

Prototype: public static final int SI_ES = 4

Description: SI information identification – Elementary stream (ES) information.

B.7.1.1.6 SI_EVENT

Prototype: public static final int SI_EVENT = 5

Description: SI information identification – Event (event) information.

B.7.1.1.7 SI_TIME

Prototype: public static final int SI_TIME = 6

Description: SI information identification – Time information.

B.7.1.2 Method

B.7.1.2.1 getType

Prototype: public int getType()

Description: Getting SI object type that implements this interface.

Parameter: None.

Return: Int type, indicating SI object type that implements this interface. For the value, please refer to the "PSI/SI Object Type" constant field definition of the org.ngb.broadcast.dvb.si.SICommonInformation interface.

B.7.1.2.2 getSIDatabase

Prototype: public org.ngb.broadcast.dvb.si.SIDatabase getSIDatabase()

Description: Getting PSI/SI database to which the SI object implementing this interface belongs.

Parameter: None.

Return: An SIDatabase object, indicating PSI/SI database to which the SI object implementing this interface belongs.

B.7.2 Interface org.ngb.broadcast.dvb.si.SINetwork

Prototype: public interface org.ngb.broadcast.dvb.si.SINetwork

extends org.ngb.broadcast.dvb.si.SICommonInformation

Description: A Network information interface, providing a method for obtaining network information. Each SINetwork object is uniquely identified by network_id. The receiving terminal may access multiple broadcast networks at the same time, for example, access to the national standard terrestrial wireless network and wired network at the same time, and distinguish network objects by network_id.

B.7.2.1 Method

B.7.2.1.1 getNetworkID

Prototype: public int getNetworkID()

Description: Getting network identifier (ie the network_id field of the NIT).

Parameter: None.

Return: Int type, indicating the unique identifier (network_id) of the unidirectional broadcast network object.

B.7.2.1.2 getNetworkName

Prototype: public java.lang.String getNetworkName()

Description: Getting the full name of the network.

Return: A java.lang.String object, indicating the network name.

B.7.2.1.3 getShortNetworkName

Prototype: public java.lang.String getShortNetworkName()

Description: Getting abbreviation of the network name.

Return: A java.lang.String object, indicating the abbreviation of the network name, or null is returned if there is no abbreviation of the network name.

Example: "[0x86]Asterix[0x87] Digital Satellite TV Network"

Full name of the network:"Asterix Digital Satellite TV Network".

Abbreviation of the network name:"Asterix".

B.7.2.1.4 getServicesLocators

Prototype: public org.ngb.broadcast.dvb.si.DvbNetworkBoundLocator[] getServicesLocators()

Description: Obtain locators of all services belonging to the network.

Parameter: None.

Return: An org.davic.net.dvb.DvbNetworkBoundLocator object array, indicating locators of all services under the network. If not, the length of the returned array is 0.

B.7.2.1.5 getService

Prototype: public org.ngb.broadcast.dvb.si.SIService getService

(org.davic.net.dvb.DvbLocator locator)

Description: According to the specified service locator, the service objects belonging to the network are obtained.

Parameter: locator – An org.davic.net.dvb.DvbLocator object, indicating the locator of the service.

Return: An org.davic.net.dvb.si.SIService object, indicating the service object under the network.

B.7.3 Interface org.ngb.broadcast.dvb.si.SIBouquet

Prototype: public interface org.ngb.broadcast.dvb.si.SIBouquet extends org.ngb.broadcast.dvb.si.SICommonInformation

Description: Service group (bouquet) information interface, each org.ngb.broadcast.dvb.si.SIBouquet object is uniquely identified by network_id and bouquet_id, that is, if service groups with the same bouquet_id appear in multiple networks, this specification stipulates that multiple org.ngb.broadcast.dvb.si.SIBouquet object instances should be created for such service groups, and they are distinguished by network_id.

B.7.3.1 Method

B.7.3.1.1 getNetwork

Prototype: public org.ngb.broadcast.dvb.si.SINetwork getNetwork()

Description: Getting the network of this org.ngb.broadcast.dvb.si.sibouquet object.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.SINetwork object, indicating the network instance to which the org.ngb.broadcast.dvb.si.sibouquet object belongs.

B.7.3.1.2 getNetworkID

Prototype: public int getNetworkID()

Description: Getting network identifier to which the service group belongs.

Parameter: None.

Return: Int type, indicating the network identifier (network_id).

B.7.3.1.3 getBouquetID

Prototype: public int getBouquetID()

Description: Obtain a service group identifier (that is, the bouquet_id field of the BAT table).

Parameter: None.

Return: Int type, indicating the service group identifier (bouquet_id).

B.7.3.1.4 getBouquetName

Prototype: public java.lang.String getBouquetName()

Description: Getting the full name of the service group.

Parameter: None.

Return: A java.lang.String object, indicating the name of the service group. If no information is available, null will be returned.

B.7.3.1.5 getShortBouquetName

Prototype: public java.lang.String getShortBouquetName()

Description: Getting short name of the service group.

Parameter: None.

Return: A Java.lang.String object, indicating the short name of the service group. If the short name of the service group does not exist, null will be returned.

B.7.3.1.6 getService

Prototype: public SIService getService(org.davic.net.dvb.DvbLocator locator)

Description: According to the specified service locator, obtain the specified service in the service group.

Parameter: locator – An org.davic.net.dvb.DvbLocator object, indicating the locator of the specified service.

Return: An SIService object, indicating the service object under the service group. If it does not exist, it returns null.

B.7.3.1.7 getServicesLocators

Prototype: public DvbNetworkBoundLocator[] getServicesLocators()

Description: Obtain the locators of all services belonging to the service group.

Parameter: None.

Return: An org.davic.net.dvb.DvbNetworkBoundLocator object array, indicating the locators of all services under the service group. If not, the length of the returned array is 0.

B.7.4 Interface org.ngb.broadcast.dvb.si.SIService

Prototype: public interface org.ngb.broadcast.dvb.si.SIService extends org.ngb.broadcast.dvb.si.SICommonInformation

Description: Service information interface, each org.ngb.broadcast.dvb.si.SIService object is common uniquely identified by the four elements of network_id, original_network_id, transport_stream_id and service_id. In the same network, an org.ngb.broadcast.dvb.si.SIService object is uniquely determined by the three elements of original_network_id, transport_stream_id, and service_id. However, a service may belong to multiple networks. This specification stipulates that multiple org.ngb.broadcast.dvb.si.SIService objects should be created for such services and are distinguished by network_id, that is, in the application scenario of receiving multiple network signals, four elements are used to uniquely determine a service.

B.7.4.1 Method

B.7.4.1.1 getServiceName

Prototype: public java.lang.String getServiceName()

Description: Getting the full name of the service.

Parameter: None.

Return: A Java.lang.String object, indicating the service name, if no information is available, null is returned.

B.7.4.1.2 getShortServiceName

Prototype: public java.lang.String getShortServiceName()

Description: Getting abbreviation of the service name.

Parameter: None.

Return: A Java.lang.String object, indicating the abbreviation of the service name. If the abbreviation of the service name does not exist, null is returned.

Example: The [0x86]P[0x87]ay [0x86]M[0x87]ovie [0x86]C[0x87]hannel

Full name of the service – "The Pay Movie Channel".

Abbreviation of the service name – "PMC".

B.7.4.1.3 getServiceProviderName

Prototype: public java.lang.String getServiceProviderName()

Description: Getting full name of the service provider.

Parameter: None.

Return: A Java.lang.String object, indicating the name of the broadcast service provider. If no information is available, null will be returned.

B.7.4.1.4 getShortServiceProviderName

Prototype: public java.lang.String getShortServiceProviderName()

Description: Getting abbreviation of the service provider.

Parameter: None.

Return: A Java.lang.String object, indicating the abbreviation of the service provider's name. If there is no abbreviation information, null will be returned.

B.7.4.1.5 getServiceType

Prototype: public int getServiceType()

Description: Obtain the broadcast service type (service_type) from the service_descriptor descriptor in the SDT table.

Parameter: None.

Return: Short type, indicating the broadcast service type. For the value, see the "Service Type" constant field definition of the SIServiceType interface.

B.7.4.1.6 getChannelNumber

Prototype: public int getChannelNumber()

Description: Getting a logical channel number of the service. The method of obtaining the logical channel number is determined by the system (depending on the operator).

Parameter: None.

Return: Int type, indicating the logical channel number of the service.

NOTE – The logical channel number is unique in the same network.

B.7.4.1.7 getDvbLocator

Prototype: public org.davic.net.dvb.DvbNetworkBoundLocator getDvbLocator()

Description: Getting a locator of the service object.

Parameter: None.

Return: An org.davic.net.dvb.DvbNetworkBoundLocator object, indicating the locator of this service object.

B.7.4.1.8 getNetworkID

Prototype: public int getNetworkID()

Description: Getting the network identifier to which the service object belongs.

Parameter: None.

Return: Int type, indicating the network identifier (network_id).

B.7.4.1.9 getOriginalNetworkID

Prototype: public int getOriginalNetworkID()

Description: Getting the original network identifier to which the service object belongs.

Parameter: None.

Return: Int type, indicating the original network identifier to which the service object belongs.

B.7.4.1.10 getTransportStreamID

Prototype: public int getTransportStreamID()

Description: Getting a transport stream identifier to which the service object belongs.

Parameter: None.

Return: Int type, indicating the transport stream identifier to which the service object belongs.

B.7.4.1.11 getServiceID

Prototype: public int getServiceID()

Description: Getting the service identifier.

Parameter: None.

Return: Int type, indicating the service identifier.

B.7.4.1.12 getNetwork

Prototype: public org.ngb.broadcast.dvb.si.SINetwork getNetwork()

Description: Getting SINetwork object to which this org.ngb.broadcast.dvb.si.SIService object belongs.

Parameter: None.

Return: An SINetwork object, indicating the SINetwork object to which this SIService object belongs.

B.7.4.1.13 getTransportStream

Prototype: public org.ngb.broadcast.dvb.si.SITransportStream getTransportStream()

Description: Getting org. ngb.broadcast.dvb.si.SITransportStream object to which this org.ngb.broadcast.dvb.si.SIService object belongs.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.SITransportStream object, indicating this org.ngb.broadcast.dvb.si.SITransportStream object to which the org.ngb.broadcast.dvb.si.

SIService object belongs.

B.7.4.1.14 getBouquets

Prototype: public org.ngb.broadcast.dvb.si.SIBouquet[] getBouquets()

Description: Getting all org.ngb.broadcast.dvb.si.SIBouquet objects to which this org.ngb.broadcast.dvb.si.SIService object belongs.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.SIBouquet object array, indicating all org.ngb.broadcast.dvb.si.SIBouquet objects to which this org.ngb.broadcast.dvb.si.

SIService object belongs.

B.7.4.1.15 getFreeCAMode

Prototype: public boolean getFreeCAMode()

Description: Obtain the free_CA_mode flag from the SDT table.

Parameter: None.

Return: Boolean type, indicating the free_CA_mode flag, true value indicating that the service is scrambled and the receiving is controlled by the CA; false value indicating that the service is not scrambled and the receiving is free.

B.7.4.1.16 getPcrPID

Prototype: public int getPcrPID()

Description: Obtain the TS_PID of the PCR referenced by the service (that is, the PCR_PID field carried by the PMT).

Parameter: None.

Return: Int type, indicating the PCR_PID of the broadcast service.

B.7.4.1.17 getEITPresentFollowingFlag

Prototype: public boolean getEITPresentFollowingFlag()

Description: Obtain the EIT_present_following_flag flag from the SDT table.

Parameter: None.

Return: Boolean type, indicating the EIT_present_follwing_flag flag, true value indicatingindicating that the service has current/follow-up information of EIT, and false value indicating that the service has no current/follow-up information of EIT.

B.7.4.1.18 getEITScheduleFlag

Prototype: public booelan getEITScheduleFlag()

Description: Obtain the EIT_schedule_flag flag from the SDT table.

Parameter: None.

Return: Boolean type, indicating the EIT_schedule_flag flag, true value indicating that the service has EIT schedule information, and false value indicating that the service does not have EIT schedule information.

B.7.4.1.19 retrieveElementaryStreams

Prototype: public SIRequest retrieveElementaryStreams(java.lang.Object appData,

org.ngb.broadcast.dvb.si.SIRetrieveListener listener,

short[] someDescriptorTags)

throws java.lang.IllegalArgumentException

Description: An Asynchronous method, obtaining the elementary stream information contained in the broadcast service.

– When the interface successfully completes the acquisition action, it will send an org.ngb.broadcast.dvb.si.SISuccessRetrieveEvent event to the listener, and return an array of org.ngb.broadcast.dvb.si.SICommonInformation objects through the getResult() method of the event object, from this array, all retrieved elementary stream objects (SIElementaryStream) can be obtained;
– If the elementary stream object that meets the conditions is not retrieved, the org.ngb.broadcast.dvb.si.SIFailureRetrieveEvent event should be sent to the listener, and the failure reason org.ngb.broadcast.dvb.si.SIRequestFailureType can be obtained from the event object.

Parameter: appData – A java.lang.Object object, indicating additional information provided by the application. When the retrieval action is completed, this object will be passed to the listening interface by the system. The application can use this object for internal communication. If the application does not require any additional information, this parameter can be set to null;

listener – An org.ngb.broadcast.dvb.si.SIRetrieveListener object, is used to receive retrieval notification events;

someDescriptorTags – A Short type array, indicating a set of tag values of descriptors that the application cares about. The interface should retrieve all descriptor information that the application cares about, represented by an org.ngb.broadcast.dvb.si.SIDescriptor object.

– If the array contains only one element and the value is -1, it indicates that the application cares about all the descriptors in the PMT table;
– If the array object is null, it indicates that the application does not care about any descriptors.

Return: An SIRequest object, indicating an information retrieval request session.

Exception: java.lang.IllegalArgumentException – If the input parameter is invalid, the exception is thrown.

B.7.5 Interface org.ngb.broadcast.dvb.si.SITransportStream

Prototype: public interface org.ngb.broadcast.dvb.si.SITransportStream extends org.ngb.broadcast.dvb.si.SICommonInformation

Description: Transport stream (transport_stream) information interface. Each org.ngb.broadcast.dvb.si.SITransportStream object is uniquely determined by network_id, original_network_id and transport_stream_id. In the same network, the transport stream is uniquely determined by both original_network_id and transport_stream_id; However, a transport stream may be transmitted on multiple networks. This specification stipulates that multiple org.ngb.broadcast.dvb.si.SITransportStream instances should be created for such transport stream and are distinguished by network_id, that is, in application scenarios in which multiple network signals are received, an org.ngb.broadcast.dvb.si.SITransportStream object is uniquely determined through network_id, original_network_id, and transport_stream_id.

B.7.5.1 Method

B.7.5.1.1 getNetworkID

Prototype: public int getNetworkID()

Description: Getting a network identifier to which the transport stream object belongs.

Parameter: None.

Return: Int type, indicating the network identifier to which the transport stream object belongs.

B.7.5.1.2 getOriginalNetworkID

Prototype: public int getOriginalNetworkID()

Description: Getting an original network identifier to which the SI object implementing the interface belongs.

Parameter: None.

Return: Int type, indicating the original network identifier to which the SI object that implements the interface belongs.

B.7.5.1.3 getTransportStreamID

Prototype: public int getTransportStreamID()

Description: Getting the transport stream identifier to which the SI object that implements the interface belongs.

Parameter: None.

Return: Int type, indicating the transport stream identifier to which the SI object that implements the interface belongs.

B.7.5.1.4 getNetwork

Prototype: public org.ngb.broadcast.dvb.si.SINetwork getNetwork()

Description: Getting SINetwork object to which this org.ngb.broadcast.dvb.si.SITransportStream object belongs.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.SINetwork object, indicating the SINetwork object to which this org.ngb.broadcast.dvb.si.SITransportStream object belongs.

B.7.6 Interface org.ngb.broadcast.dvb.si.SIElementaryStream

Prototype: public interface org.ngb.broadcast.dvb.si.SIElementaryStream extends org.ngb.broadcast.dvb.si.SICommonInformation

Description: Elementary stream (elementary_stream) information interface of the broadcast service. Each org.ngb.broadcast.dvb.si.SIElementaryStream object is uniquely determined by the network_id, original_network_id, transport_stream_id, service_id, and component_tag (or elementary_PID) identifiers. An elementary stream may belong to multiple services. This specification stipulates that multiple org.ngb.broadcast.dvb.si.SIElementaryStream objects should be created for such elementary stream, it is jointly and uniquely determined by network_id, original_network_id, transport_stream_id, service_id and component_tag (or elementary_PID).

B.7.6.1 Method

B.7.6.1.1 getComponentTag

Prototype: public byte getComponentTag()

Description: Getting a tag of the elementary stream component.

Parameter: None.

Return: Byte type, indicating the elementary stream component tag. If the elementary stream does not carry stream_identifier_descriptor, its component tag value defaults to -2.

B.7.6.1.2 getElementaryPID

Prototype: public short getElementaryPID()

Description: Obtain the transport stream packet identifier (TS_PID) that carries the elementary stream.

Parameter: None.

Return: Short type, indicating the transport stream packet identifier that carries the elementary stream.

B.7.6.1.3 getNetworkID

Prototype: public int getNetworkID()

Description: Getting the network identifier to which the elementary stream object belongs.

Parameter: None.

Return: Int type, indicating the network identifier (network_id).

B.7.6.1.4 getOriginalNetworkID

Prototype: public int getOriginalNetworkID()

Description: Getting an original network identifier to which the elementary stream object belongs.

Parameter: None.

Return: Int type, indicating the original network identifier to which the elementary stream object that implements the interface belongs.

B.7.6.1.5 getTransportStreamID

Prototype: public int getTransportStreamID()

Description: Getting the transport stream identifier to which the elementary stream object belongs.

Parameter: None.

Return: Int type, indicating the transport stream identifier to which the elementary stream object belongs.

B.7.6.1.6 getServiceID

Prototype: public int getServiceID()

Description: Getting the service identifier to which the elementary stream object belongs.

Parameter: None.

Return: Int type, indicating the service identifier to which the elementary stream object belongs.

B.7.6.1.7 getStreamType

Prototype: public byte getStreamType()

Description: Getting the elementary stream type.

Parameter: None.

Return: Byte type, indicating the elementary stream type. For the value, see the stream type constant field definition of the org.ngb.broadcast.dvb.si.SIStreamType interface.

B.7.6.1.8 getService

Prototype: public SIService getService()

Description: Getting the org.ngb.broadcast.dvb.si.SIService object to which this org.ngb.broadcast.dvb.si.SIElementaryStream object belongs.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.SIService object, indicating the SIService object to which this org.ngb.broadcast.dvb.si.SIElementaryStream object belongs.

B.7.7 Interface org.ngb.broadcast.dvb.si.SIEvent

Prototype: public interface org.ngb.broadcast.dvb.si.SIEvent

extends org.ngb.broadcast.dvb.si.SICommonInformation

Description: Program event information interface, each org.ngb.broadcast.dvb.si.SIEvent object is uniquely identified by network_id, original_network_id, transport_stream_id, service_id, and event_id. A program event may belong to multiple services. This specification stipulates that multiple org.ngb.broadcast.dvb.si.SIEvent objects should be created for such a program event, it is uniquely identified by network_id, original_network_id, transport_stream_id, service_id, and event_id.

B.7.7.1 Method

B.7.7.1.1 getNetworkID

Prototype: public int getNetworkID()

Description: Getting the network identifier to which the elementary stream object belongs.

Parameter: None.

Return: Int type, indicating the network identifier (network_id).

B.7.7.1.2 getOriginalNetworkID

Prototype: public int getOriginalNetworkID()

Description: Getting the original network identifier to which the event object belongs.

Parameter: None.

Return: Int type, indicating the original network identifier to which the event object belongs.

B.7.7.1.3 getTransportStreamID

Prototype: public int getTransportStreamID()

Description: Getting the transport stream identifier to which the event object belongs.

Parameter: None.

Return: Int type, indicating the transport stream identifier to which the event object belongs.

B.7.7.1.4 getServiceID

Prototype: public int getServiceID()

Description: Getting the service identifier to which the event object belongs.

Parameter: None.

Return: Int type, indicating the service identifier to which the event object belongs.

B.7.7.1.5 getEventID

Prototype: public int getEventID()

Description: Getting the event identifier.

Parameter: None.

Return: Int type, indicating the event identifier.

B.7.7.1.6 getDvbLocator

Prototype: public org.davic.net.dvb.DvbNetworkBoundLocator getDvbLocator()

Description: Getting the locator of the event object.

Parameter: None.

Return: An org.davic.net.dvb.DvbNetworkBoundLocator object, indicating the locator of the event object.

B.7.7.1.7 getService

Prototype: public org.ngb.broadcast.dvb.si.SIService getService()

Description: Getting the org.ngb.broadcast.dvb.si. SIService object to which this org.ngb.broadcast.dvb.si.SIEvent object belongs.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.SIService object, indicating the org.ngb.broadcast.dvb.si.SIService object to which this org.ngb.broadcast.dvb.si.SIEvent object belongs.

B.7.7.1.8 getNibbles

Prototype: public byte[] getNibbles()

Description: Getting the program content classification information, which comes from the content descriptor (content_descriptor) in the EIT.

Parameter: None.

Return: Byte type array, indicating the content classification information associated with this program event. If the content descriptor (content_descriptor) does not exist, null is returned.

– byte[0] – The upper 4 bits represent the second-level content classification (content_nibble_level_2); the lower 4 bits represent the first-level content classification (content_nibble_level_1);

– byte[1] – The upper 4 bits represent the second-level custom classification (user_nibble_level_2); the lower 4 bits represent the first-level custom classification (user_nibble_level_1).

B.7.7.1.9 getStartTime

Prototype: public java.util.Date getStartTime()

Description: Getting the start time of the event.

Parameter: None.

Return: A java.util.Date object, indicating the start time of the broadcast program event.

B.7.7.1.10 getDuration

Prototype: public long getDuration()

Description: Getting the duration of the event.

Parameter: None.

Return: Long type, indicating the duration of the event, in seconds.

B.7.7.1.11 getEndTime

Prototype: public Date getEndTime()

Description: Getting the end time of the event.

Parameter: None.

Return: Java.util.Date object, indicating the end time of the event.

B.7.7.1.12 getEventName

Prototype: String getEventName()

Description: Getting the full name of the event (event_name).

Parameter: None.

Return: Java.lang.String object, indicating the full name of the event.

B.7.7.1.13 getShortEventName

Prototype: public java.lang.String getShortEventName()

Description: Getting the abbreviation of the event name.

Parameter: None.

Return: Java.lang.String object, indicates the abbreviation of the event name. If there is no abbreviation information, null is returned.

B.7.7.1.14 getEventDescription

Prototype: public java.lang.String getEventDescription()

Description: Getting the description of the event.

Parameter: None.

Return: Java.lang.String object, indicates the description of the event.

B.7.7.1.15 getFreeCAMode

Prototype: public boolean getFreeCAMode()

Description: Getting the scrambling flag (free_CA_mode) of the broadcast program event.

Parameter: None.

Return: Boolean type, indicating whether the broadcast program event is scrambled, true value indicating that the event is scrambled and receiving is controlled by CA; false value indicating that it is not scrambled and can be received freely.

B.7.7.1.16 getRunningStatus

Prototype: public byte getRunningStatus()

Description: Getting the running status information of the event.

Parameter: None.

Return: Byte type, indicating the running status of the event, for the value, please refer to the "running status" constant field definition of the org.ngb.broadcast.dvb.si.SIRunningStatus interface.

B.7.8 Interface org.ngb.broadcast.dvb.si.SITime

Prototype: public interface org.ngb.broadcast.dvb.si.SITime

extends org.ngb.broadcast.dvb.si.SICommonInformation

Description: A Time information interface, providing a method to obtain time information, the time information is obtained from TDT or TOT.

B.7.8.1 Method

B.7.8.1.1 getUTCTime

Prototype: java.util.Date getUTCTime()

Description: Getting UTC time and date.

Parameter: None.

Return: Java.util.Date object, indicating UTC time, the information is obtained from TDT or TOT.

B.7.9 Interface org.ngb.broadcast.dvb.si.SIDescriptor

Prototype: public interface org.ngb.broadcast.dvb.si.SIDescriptor

Description: Descriptor information interface. The descriptor consists of three parts, as shown in Figure B.3:

– Tag (descriptor_tag) – Uniquely identifies a descriptor;
– Content length (descriptor_length) – Indicating the number of bytes in the content part;
– Content: Byte array, the specific syntax and semantics are related to the tag type.

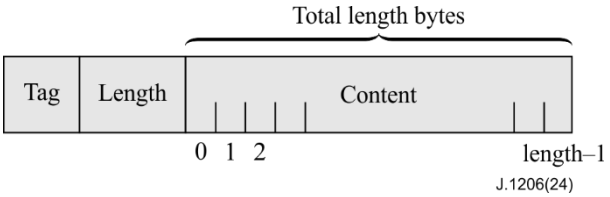


Figure B.3 – Schematic diagram of SIDescriptor structure

B.7.9.1 Method

B.7.9.1.1 getByteAt

Prototype: public byte getByteAt(int index) throws java.lang.IndexOutOfBoundsException

Description: Getting the byte at the specified index position in the content.

Parameter: index-int type, the index value of the descriptor content position, position 0 corresponds to the first byte after the descriptor_length field.

Return: Byte type, indicating the byte data at the specified position.

Exception: java.lang.IndexOutOfBoundException – If the parameter index is less than 0 or index is greater than or equal to descriptor_length, this exception is thrown.

B.7.9.1.2 getContent

Prototype: public byte[] getContent()

Description: Getting the content of the descriptor.

Parameter: None.

Return: A byte array, indicating the content part of the descriptor, that is, all bytes after the descriptor_length field.

B.7.9.1.3 getContentLength

Prototype: public short getContentLength()

Description: Getting the length of the descriptor content.

Parameter: None.

Return: Short type, indicating the length of the descriptor content, that is, the value of the descriptor_length field, in bytes.

B.7.9.1.4 getTag

Prototype: public short getTag()

Description: Getting the descriptor tag (descriptor_tag).

Parameter: None.

Return: Short type, indicating the value of the descriptor tag field. For the value, see the "descriptor tag" constant field definition of the SIDescriptorTag interface.

B.7.10 Interface org.ngb.broadcast.dvb.si.SIRequest

Prototype: public interface org.ngb.broadcast.dvb.si.SIRequest

Description: PSI/SI information request interface, describes a PSI/SI information retrieval request generated by the application. The application can cancel the request through this object.

B.7.10.1 Method

B.7.10.1.1 cancelRequest

Prototype: public void cancelRequest()

Description: Cancel this request action.

Parameter: None.

Return: None.

B.7.11 Interface org.ngb.broadcast.dvb.si.SIRetrieveListener

Prototype: public interface org.ngb.broadcast.dvb.si.SIRetrieveListener extends java.util.EventListener

Description: SI information acquisition event listener is implemented by the application program.

B.7.11.1 Method

B.7.11.1.1 postEvent

Prototype: public void postEvent(org.ngb.broadcast.dvb.si.SIRetrieveEvent event)

Description: Send SI information to get the event.

Parameter: event – An org.ngb.broadcast.dvb.si.SIRetrieveEvent object, indicating SI information acquisition event. The application should call the instanceOf method to further determine the object type of the event, which may be an org.ngb.broadcast.dvb.si.SISuccessRetrieveEvent object or an org.ngb.broadcast.dvb.si.SIFailureRetrieveEvent object.

Return: None.

B.7.12 Interface org.ngb.broadcast.dvb.si.SIUpdateListener

Prototype: public interface org.ngb.broadcast.dvb.si.SIUpdateListener extends java.util.EventListener

Description: PSI/SI table update event listener is implemented by the application program.

B.7.12.1 Method

B.7.12.1.1 postEvent

Prototype: public void postEvent(org.ngb.broadcast.dvb.si.SIUpdateEvent event)

Description: PSI/SI update event callback method is used to obtain the changed PSI/SI object.

Parameter: event – An org.ngb.broadcast.dvb.si.SIUpdateEvent object, indicating an event in which PSI/SI information changes.

Return: None.

B.7.13 Interface org.ngb.broadcast.dvb.si.SIDescriptorTag

Prototype: public interface org.ngb.broadcast.dvb.si.SIDescriptorTag

Description: Descriptor tag constant interface, see GB/T 28161-2011 for details.

B.7.13.1 Constant field – descriptor tag

B.7.13.1.1 TAG_NETWORK_NAME

Prototype: public static final short TAG_NETWORK_NAME = 64

Description: Network name descriptor tag.

B.7.13.1.2 TAG_SERVICE_LIST

Prototype: public static final short TAG_SERVICE_LIST = 65

Description: Service list descriptor tag.

B.7.13.1.3 TAG_STUFFING

Prototype: public static final short TAG_STUFFING = 66

Description: Fill the descriptor tag.

B.7.13.1.4 TAG_SATELLITE_DELIVERY_SYSTEM

Prototype: public static final short TAG_SATELLITE_DELIVERY_SYSTEM = 67

Description: Satellite delivery system descriptor tag.

B.7.13.1.5 TAG_CABLE_DELIVERY_SYSTEM

Prototype: public static final short TAG_CABLE_DELIVERY_SYSTEM = 68

Description: Wire delivery system descriptor tag.

B.7.13.1.6 TAG_VBI_DATA

Prototype: public static final short TAG_VBI_DATA = 69

Description: VBI data descriptor tag.

B.7.13.1.7 TAG_TELETEXT

Prototype: public static final short TAG_TELETEXT = 70

Description: Teletext descriptor tag.

B.7.13.1.8 TAG_BOUQUET_NAME

Prototype: public static final short TAG_BOUQUET_NAME = 71

Description: Service group name descriptor tag.

B.7.13.1.9 TAG_SERVICE

Prototype: public static final short TAG_SERVICE = 72

Description: Service descriptor tag.

B.7.13.1.10 TAG_COUNTRY_AVAILABILITY

Prototype: public static final short TAG_COUNTRY_AVAILABILITY = 73

Description: Descriptor for country/region service approval.

B.7.13.1.11 TAG_LINKAGE

Prototype: public static final short TAG_LINKAGE = 74

Description: Link descriptor tag.

B.7.13.1.12 TAG_NVOD_REFERENCE

Prototype: public static final short TAG_NVOD_REFERENCE = 75

Description: Quasi-video-on-demand reference descriptor tag.

B.7.13.1.13 TAG_TIME_SHIFTED_SERVICE

Prototype: public static final short TAG_TIME_SHIFTED_SERVICE = 76

Description: Time-shifted service descriptor tag.

B.7.13.1.14 TAG_SHORT_EVENT

Prototype: public static final short TAG_SHORT_EVENT = 77

Description: Short event descriptor tag.

B.7.13.1.15 TAG_EXTENDED_EVENT

Prototype: public static final short TAG_EXTENDED_EVENT = 78

Description: Extended event descriptor tag.

B.7.13.1.16 TAG_TIME_SHIFTED_EVENT

Prototype: public static final short TAG_TIME_SHIFTED_EVENT = 79

Description: Time-shifted event descriptor tag.

B.7.13.1.17 TAG_COMPONENT

Prototype: public static final short TAG_COMPONENT = 80

Description: Component descriptor tag.

B.7.13.1.18 TAG_MOSAIC

Prototype: public static final short TAG_MOSAIC = 81

Description: Mosaic descriptor tag.

B.7.13.1.19 TAG_STREAM_IDENTIFIER

Prototype: public static final short TAG_STREAM_IDENTIFIER = 82

Description: Stream identification descriptor tag.

B.7.13.1.20 TAG_CA_IDENTIFIER

Prototype: public static final short TAG_CA_IDENTIFIER = 83

Description: Conditional receiving identification descriptor tag.

B.7.13.1.21 TAG_CONTENT

Prototype: public static final short TAG_CONTENT = 84

Description: Content descriptor tag.

B.7.13.1.22 TAG_PARENTAL_RATING

Prototype: public static final short TAG_PARENTAL_RATING = 85

Description: Parental rating descriptor tag.

B.7.13.1.23 TAG_VBI_TELETEXT

Prototype: public static final short TAG_VBI_TELETEXT = 86

Description: VBI teletext descriptor.

B.7.13.1.24 TAG_TELEPHONE

Prototype: public static final short TAG_TELEPHONE = 87

Description: Phone descriptor tag.

B.7.13.1.25 TAG_LOCAL_TIME_OFFSET

Prototype: public static final short TAG_LOCAL_TIME_OFFSET = 88

Description: Local time offset descriptor tag.

B.7.13.1.26 TAG_SUBTITLING

Prototype: public static final short TAG_SUBTITLING = 89

Description: Subtitle descriptor tag.

B.7.13.1.27 TAG_TERRESTRIAL_DELIVERY_SYSTEM

Prototype: public static final short TAG_TERRESTRIAL_DELIVERY_SYSTEM = 90

Description: Ground delivery system descriptor tag.

B.7.13.1.28 TAG_MULTILINGUAL_NETWORK_NAME

Prototype: public static final short TAG_MULTILINGUAL_NETWORK_NAME = 91

Description: Multilingual network name descriptor tag.

B.7.13.1.29 TAG_MULTILINGUAL_BOUQUET_NAME

Prototype: public static final short TAG_MULTILINGUAL_BOUQUET_NAME = 92

Description: Multilingual service group descriptor tag.

B.7.13.1.30 TAG_MULTILINGUAL_SERVICE_NAME

Prototype: public static final short TAG_MULTILINGUAL_SERVICE_NAME = 93

Description: Multilingual service name descriptor tag.

B.7.13.1.31 TAG_MULTILINGUAL_COMPONENT

Prototype: public static final short TAG_MULTILINGUAL_COMPONENT = 94

Description: Multilingual component descriptor tag.

B.7.13.1.32 TAG_PRIVATE_DATA_SPECIFIER

Prototype: public static final short TAG_PRIVATE_DATA_SPECIFIER = 95

Description: Private data specifier descriptor tag.

B.7.13.1.33 TAG_SERVICE_MOVE

Prototype: public static final short TAG_SERVICE_MOVE = 96

Description: Service move descriptor tag.

B.7.13.1.34 TAG_SHORT_SMOOTHING_BUFFER

Prototype: public static final short TAG_SHORT_SMOOTHING_BUFFER = 97

Description: Short smoothing buffer descriptor tag.

B.7.13.1.35 TAG_FREQUENCY_LIST

Prototype: public static final short TAG_FREQUENCY_LIST = 98

Description: Frequency list descriptor tag.

B.7.13.1.36 TAG_PARTIAL_TRANSPORT_STREAM

Prototype: public static final short TAG_PARTIAL_TRANSPORT_STREAM = 99

Description: Frequency list descriptor tag.

B.7.13.1.37 TAG_DATA_BROADCAST

Prototype: public static final short TAG_DATA_BROADCAST = 100

Description: Data broadcast descriptor tag.

B.7.13.1.38 TAG_CA_SYSTEM

Prototype: public static final short TAG_CA_SYSTEM = 101

Description: CA system descriptor tag.

B.7.13.1.39 TAG_DATA_BROADCAST_ID

Prototype: public static final short TAG_DATA_BROADCAST_ID = 102

Description: Data broadcast identification descriptor tag.

B.7.13.1.40 TAG_TRANSPORT_STREAM

Prototype: public static final short TAG_TRANSPORT_STREAM = 103

Description: Transport stream descriptor tag.

B.7.13.1.41 TAG_DSNG

Prototype: public static final short TAG_DSNG = 104

Description: Digital satellite news gathering descriptor tag.

B.7.13.1.42 TAG_PDC

Prototype: public static final short TAG_PDC = 105

Description: Program delivery control descriptor tag.

B.7.13.1.43 TAG_AC_3

Prototype: public static final short TAG_AC_3 = 106

Description: AC3 descriptor tag.

B.7.13.1.44 TAG_ANCILLARY_DATA

Prototype: public static final short TAG_ANCILLARY_DATA = 107

Description: Ancillary data descriptor tag.

B.7.13.1.45 TAG_ANNOUNCEMENT_SUPPORT

Prototype: public static final short TAG_ANNOUNCEMENT_SUPPORT = 110

Description: Announcement support descriptor tag.

B.7.14 Interface org.ngb.broadcast.dvb.si.SIRunningStatus

Prototype: public interface org.ngb.broadcast.dvb.si.SIRunningStatus

Description: Running state constant definition interface of the broadcast service (service) or program (event), see GB/T 28161-2011 for the value.

B.7.14.1 Constant field – running status

B.7.14.1.1 UNDEFINED

Prototype: public static final byte UNDEFINED = 0

Description: Running status-undefined.

B.7.14.1.2 NOT_RUNNING

Prototype: public static final byte NOT_RUNNING = 1

Description: Running status-not running.

B.7.14.1.3 STARTS_IN_A_FEW_SECONDS

Prototype: public static final byte STARTS_IN_A_FEW_SECONDS = 2

Description: Running status-about to run.

B.7.14.1.4 PAUSING

Prototype: public static final byte PAUSING = 3

Description: Running state-pausing state.

B.7.14.1.5 RUNNING

Prototype: public static final byte RUNNING = 4

Description: Running status-running.

B.7.15 Interface org.ngb.broadcast.dvb.si.SIServiceType

Prototype: public interface org.ngb.broadcast.dvb.si.SIServiceType

Description: Service type (service_type) constant definition interface, and the value is detailed in GB/T 28161-2011.

B.7.15.1 Constant field – service type

B.7.15.1.1 SERVICE_TYPE_RESERVED

Prototype: public static final short SERVICE_TYPE_RESERVED = 0

Description: Service type-reserved for use.

B.7.15.1.2 SERVICE_TYPE_DIGITAL_TELEVISION

Prototype: public static final short SERVICE_TYPE_DIGITAL_TELEVISION = 1

Description: Service type – digital television broadcasting service.

B.7.15.1.3 SERVICE_TYPE_DIGITAL_RADIO_SOUND

Prototype: public static final short SERVICE_TYPE_DIGITAL_RADIO_SOUND = 2

Description: Service type – digital sound broadcasting service.

B.7.15.1.4 SERVICE_TYPE_TELETEXT

Prototype: public static final short SERVICE_TYPE_TELETEXT = 3

Description: Service type – teletext service.

B.7.15.1.5 SERVICE_TYPE_NVOD_REFERENCE

Prototype: public static final short SERVICE_TYPE_NVOD_REFERENCE = 4

Description: Service type – NVOD reference service.

B.7.15.1.6 SERVICE_TYPE_NVOD_TIME_SHIFTED

Prototype: public static final short SERVICE_TYPE_NVOD_TIME_SHIFTED = 5

Description: Service type – NVOD time shifted service.

B.7.15.1.7 SERVICE_TYPE_MOSAIC

Prototype: public static final short SERVICE_TYPE_MOSAIC = 6

Description: Service type – mosaic service.

B.7.15.1.8 SERVICE_TYPE_DATA_BROADCAST

Prototype:static final short SERVICE_TYPE_DATA_BROADCAST = 12

Description: Service type – data broadcasting service.

B.7.16 Interface org.ngb.broadcast.dvb.si.SIStreamType

Prototype: public interface org.ngb.broadcast.dvb.si.SIStreamType

Description: Elementary stream stream type constant (stream_type) definition interface, for the value of the stream type constant, please refer to the standard definition of GB/T 17975.1-2010 "General Coding of Information Technology Moving Pictures and Its Accompanying Information Part 1: System".

B.7.16.1 Constant – ES Stream Tpe

B.7.16.1.1 ES_PRIVATE_RESERVED

Prototype: public static final byte ES_PRIVATE_RESERVED = 0

Description: ES stream type-reserved.

B.7.16.1.2 ES_MPEG1_VIDEO

Prototype: public static final byte ES_MPEG1_VIDEO = 1

Description: ES stream type – GB/T 17191.2 video.

B.7.16.1.3 ES_MPEG2_VIDEO

Prototype: public static final byte ES_MPEG2_VIDEO = 2

Description: ES stream type – GB/T 17975.2 video.

B.7.16.1.4 ES_MPEG1_AUDIO

Prototype: public static final byte ES_MPEG1_AUDIO = 3

Description: ES stream type – GB/T 17191.2 audio.

B.7.16.1.5 ES_MPEG2_AUDIO

Prototype: public static final byte ES_MPEG2_AUDIO = 4

Description: ES stream type – GB/T 17975.2 audio.

B.7.16.1.6 ES_PRIVATE_SECTION

Prototype: public static final byte ES_PRIVATE_SECTION = 5

Description: ES stream type-private section.

B.7.16.1.7 ES_PRIVATE_DATA

Prototype: public static final byte ES_PRIVATE_DATA = 6

Description: ES stream type – PES packet containing private data.

B.7.16.1.8 ES_MHEG

Prototype: public static final byte ES_MHEG = 7

Description: ES stream type – ISO/IEC 13522-1 MHEG.

B.7.16.1.9 ES_DSMCC

Prototype: public static final byte ES_DSMCC = 8

Description: Stream type – ISO/IEC 13818-1 Appendix B DSMCC.

B.7.16.1.10 ES_DSMCC_A

Prototype: public static final byte ES_DSMCC_A = 10

Description: ES stream type – ISO/IEC 13818-6 Type A.

B.7.16.1.11 ES_DSMCC_B

Prototype: public static final byte ES_DSMCC_B = 11

Description: ES stream type – ISO/IEC 13818-6 Type B.

B.7.16.1.12 ES_DSMCC_C

Prototype: public static final byte ES_DSMCC_C = 12

Description: ES stream type – ISO/IEC 13818-6 Type C.

B.7.16.1.13 ES_DSMCC_D

Prototype: public static final byte ES_DSMCC_D = 13

Description: ES stream type – ISO/IEC 13818-6 Type D.

B.7.16.1.14 ES_AVS_VIDEO

Prototype: public static final byte ES_AVS_VIDEO = 66

Description: ES stream type – GB/T 20090.2 – 2006 video type.

B.7.17 Class org.ngb.broadcast.dvb.si.SIDatabase

Prototype: public class org.ngb.broadcast.dvb.si.SIDatabase

Description: A PSI/SI information database, providing the management and operation methods of the PSI/SI information database in DVB mode, and is an entry class for applications to obtain PSI/SI information. Each physical broadcast network interface corresponds to a SIDatabase object. If the

receiving terminal has only one physical broadcast network interface, there is only one org.ngb.broadcast.dvb.si.SIDatabase object. When implementing the system, the following special application scenarios need to be considered:

– Scenario 1 – The same physical interface is connected to multiple broadcast networks. For example, terrestrial wireless Tuner may be connected to wireless networks of different operators. At this time, the physical interface corresponds to only one org.ngb.broadcast.dvb.si.SIDatabase object;

– Scenario 2 – Multiple physical interfaces are connected to the same broadcast network. For example, a receiver with PVR function has two Tuners connected to the wired network at the same time. In this case, the system should implement multiple org.ngb.broadcast.dvb.si.+SIDatabase objects. The information maintained by each SIDatabase is the same.

B.7.17.1 Method

B.7.17.1.1 getDatabase

Prototype: public static org.ngb.broadcast.dvb.si.SIDatabase[] getDatabase()

Description: Obtain SI database instances. The receiving terminal may have multiple broadcast network interfaces (for example, simultaneous connection to a wired network and a national standard terrestrial wireless network). Each network interface corresponds to one SI database instance. If there are multiple network interfaces, this method returns multiple SI database instances.

Return: org.ngb.broadcast.dvb.si.SIDatabase object array, indicating SI database instance.

NOTE – When implementing the system, the following special scenarios need to be considered:

– Scenario 1 – The same physical network interface is connected to multiple broadcast networks. For example, the ground wireless Tuner may be connected to the wireless networks of different operators. In this scenario, there is only one org.ngb.broadcast.dvb.si.SIDatabase instance;

– Scenario 2 – Multiple physical network interfaces are connected to the same broadcast network. For example, a receiver with PVR function has two or more Tuners connected to the same network at the same time. In this scenario, there are multiple org.ngb.broadcast.dvb.si. SIDatabase instances.

B.7.17.1.2 getID

Prototype: public int getID()

Description: Getting a globally unique identifier of the org.ngb.broadcast.dvb.si.SIDatabase object.

Parameter: None.

Return: Int type, indicating the globally unique identifier of the org.ngb.broadcast.dvb.si.SIDatabase object.

B.7.17.1.3 addNITUpdateListener

Prototype: public boolean addNITUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener, int networkId) throws java.lang.IllegalArgumentException

Description: Register the NIT update event listener. When the NIT is updated, a SIUpdateEvent event will be sent to the registered listener object. Whole process listening, when the network interface associated with the org.ngb.broadcast.dvb.si.SIDatabase object is tuned to another transport stream, the listening process does not stop.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be registered to receive the NIT update event;

networkId – Int type, indicating the unique identifier of NIT.

Return: Boolean type, indicating the registration result, true value indicating that the registration is successful, and false value indicating that the registration fails.

Exception: java.lang.IllegalArgumentException – If the network specified by the parameter network_id does not exist, this exception is thrown.

B.7.17.1.4 removeNITUpdateListener

Prototype: public boolean removeNITUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener, int network_id) throws java.lang.IllegalArgumentException

Description: Unregister the NIT update event listener. If the listener associated with the parameter network_id does not exist, this method fails, but no exception is thrown.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be unregistered that receives the NIT update event;

network_id – Int type, indicating the unique identifier of NIT.

Return: Boolean type, indicating the unregister result, true value indicating that the unregister is successful, and false value indicating that the unregister fails.

Exception: java.lang.IllegalArgumentException – If the network specified by the parameter network_id does not exist, this exception is thrown.

B.7.17.1.5 addBATUpdateListener

Prototype: public boolean addBATUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener, int networkId, int bouquetId)

throws java.lang.IllegalArgumentException

Description: Register BAT update event listener. When the BAT is updated, an org.ngb.broadcast.dvb.si.SIUpdateEvent event will be sent to the registered listener object. Whole process listening, when the network interface associated with the org.ngb.broadcast.dvb.si.SIDatabase object is tuned to another transport stream, the listening process does not stop.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be registered to receive the BAT update event;

networkId – Int type, indicating the network identifier of the network where BAT is located;

bouquetId – Int type, indicating the unique identifier of BAT.

Return: Boolean type, indicating the registration result, true value indicating that the registration is successful, and false value indicating that the registration fails.

Exception: java.lang.IllegalArgumentException –If the service group specified by the parameters network_id and bouquet_id does not exist, this exception is thrown.

B.7.17.1.6 removeBATUpdateListener

Prototype: public boolean removeBATUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener, int networkId, int bouquetId) throws java.lang.IllegalArgumentException

Description: Unregister the BAT update event listener. If the listener associated with the parameter bouquet_id does not exist, this method fails, but no exception is thrown.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be unregistered that receives the BAT update event;

networkId – Int type, indicating the network identifier of the network where BAT is located;

bouquetId – Int type, indicating the unique identifier of BAT.

Return: Boolean type, indicating the unregister result, true value indicating that the unregister is successful, and false value indicating that the unregister failed.

Exception: java.lang.IllegalArgumentException – If the service group specified by the parameters networkId and bouquetId does not exist, this exception is thrown.

B.7.17.1.7 addPATUpdateListener

Prototype: public void addPATUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener)

Description: Register the PAT update event listener. When the PAT is updated, a SIUpdateEvent event will be sent to the registered listener object. Only the PAT in the current transport stream is listened. When the network interface associated with the org.ngb.broadcast.dvb.si.SIDatabase object is tuned to another transport stream, the listening process should stop in the background.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be registered that receive the PAT update event.

Return: None.

B.7.17.1.8 removePATUpdateListener

Prototype: public void removePATUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener)

Description: Unregister the PAT update event listener.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be unregistered that receives PAT update events.

Return: None.

B.7.17.1.9 addPMTUpdateListener

Prototype: public boolean addPMTUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener, int networkId, int originalNetworkId, int transportStreamId, int serviceId) throws java.lang.IllegalArgumentException

Description: Register the PMT update event listener. If the services specified by networkId, originalNetworkId, transportStreamId and serviceId are carried in the current transport stream, when the PMT is updated, an org.ngb.broadcast.dvb.si.SIUpdateEvent object will be sent to the registered listener object. Only the PMT in the current transport stream is listened. When the network interface associated with the org.ngb.broadcast.dvb.si.SIDatabase object is tuned to another transport stream, the listening process should stop in the background.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be registered that receives the PMT update event;

networkId – Int type, indicating the network identifier;

originalNetworkId – Int type, indicating the original network identifier;

transportStreamId – Int type, indicating the transport stream identifier;

serviceId – Int type, indicating the service identifier.

Return: Boolean type, indicating the registration result, true value indicating that the registration is successful, and false value indicating that the registration fails.

Exception: java.lang.IllegalArgumentException – If the service specified by the parameters networkId, originalNetworkId, transportStreamId, and serviceId does not exist, this exception is thrown.

B.7.17.1.10 removePMTUpdateListener

Prototype: public boolean removePMTUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener, int networkId, int originalNetworkId, int transportStreamId,int serviceId) throws java.lang.IllegalArgumentException

Description: Unregister the PMT update event listener.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be unregistered that receives the PMT update event;

networkId – Int type, indicating the network identifier;

originalNetworkId – Int type, indicating the original network identifier;

transportStreamId – Int type, indicating the transport stream identifier;

serviceId – Int type, indicating the service identifier.

Return: Boolean type, indicating the unregister result, true value indicating that the unregister is successful, and false value indicating that the unregister failed.

Exception: java.lang.IllegalArgumentException – If the service specified by the parameters networkId, originalNetworkId, transportStreamId, and serviceId does not exist, this exception is thrown.

B.7.17.1.11 addSDTUpdateListener

Prototype: public boolean addSDTUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener, int networkId, int originalNetworkId, int transportStreamId) throws java.lang.IllegalArgumentException

Description: Register the SDT update event listener. When the SDT specified by the parameters network_id, originalNetworkId, and transportStreamId is updated, an SIUpdateEvent event will be sent to the registered listener object. Only the SDT in the current transport stream is listened. When the network interface associated with the SIDatabase object starts to tune to another transport stream, the listening process should stop in the background.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be registered that receives the SDT update event;

networkId – Int type, indicating the network identifier;

originalNetworkId – Int type, indicating the original network identifier;

transportStreamId – Int type, indicating the transport stream identifier.

Return: Boolean type, indicating the registration result, true value indicating that the registration is successful, and false value indicating that the registration fails.

Exception: java.lang.IllegalArgumentException – If the SDT specified by the parameters networkId, originalNetworkId, and transportStreamId does not exist, this exception is thrown.

B.7.17.1.12 removeSDTUpdateListener

Prototype: public boolean removeSDTUpdateListener(org.ngb.broadcast.dvb.si.SIUpdateListener listener, int networkId, int originalNetworkId, int transportStreamId) throws java.lang.IllegalArgumentException

Description: Unregister the SDT update event listener.

Parameter: listener – An org.ngb.broadcast.dvb.si.SIUpdateListener object, indicating the listener object to be unregistered that receives the SDT update event;

networkId – Int type, indicating the network identifier;

originalNetworkId – Int type, indicating the original network identifier;

transportStreamId – Int type, indicating the transport stream identifier.

Return: Boolean type, indicating the unregister result, true value indicating that the unregister is successful, and false value indicating that the unregister failed.

Exception: java.lang.IllegalArgumentException – If the SDT specified by the parameters networkId, originalNetworkId, and transportStreamId does not exist, this exception is thrown.

B.7.17.1.13 getAllNetworks

Prototype: public org.ngb.broadcast.dvb.si.SINetwork[] getAllNetworks()

Description: Getting current network information.

Return: org.ngb.broadcast.dvb.si.SINetwork array

B.7.17.1.14 getAllServices

Prototype: public org.ngb.broadcast.dvb.si.SIService[] getAllServices()

Description: Getting service information.

Return: org.ngb.broadcast.dvb.si.SIService array

B.7.17.1.15 getAllTransportStreams

Prototype: public org.ngb.broadcast.dvb.si.SITransportStream[] getAllTransportStreams()

Description: Getting all transport stream information.

Return: org.ngb.broadcast.dvb.si.SITransportStream array

B.7.17.1.16 getSIBouquets

Prototype: public org.ngb.broadcast.dvb.si.SIBouquet[] getSIBouquets(int network_id, int original_network_id, int transport_stream_id, int service_id)

Description: Getting a frequency point and the org.ngb.broadcast.dvb.si.SIBouquet where the program is located. If the parameters are all -1, all org.ngb.broadcast.dvb.si.SIBouquet will be returned.

Parameter: network_id – Int type, indicating the network identifier, if it is -1, it means to retrieve the network information described in the current and other network NIT tables;

original_network_id – int type, the original network identifier;

transport_stream_id – int type, the transport stream identifier;

service_id – Program identifier, that is, channel program number, if it is -1, it means it does not match the program identifier;

Return: org.ngb.broadcast.dvb.si.SIBouquet object array.

B.7.17.1.17 getSIElementStreams

Prototype: public org.ngb.broadcast.dvb.si.SIElementaryStream[] getSIElementStreams(int network_id, int original_network_id, int transport_stream_id, int service_id)

Description: Getting the SIElementaryStream of a certain program, if the parameters are all -1, then return the SIElementaryStream of all programs.

Parameter: network_id – Int type, indicating the network identifier, if it is -1, it means to retrieve the network information described in the current and other network NIT tables;

original_network_id – int type, the original network identifier;

transport_stream_id – int type, the transport stream identifier;

service_id – Program identifier, that is, channel program number, if it is -1, it means it does not match the program identifier;

Return: org.ngb.broadcast.dvb.si.SIElementaryStream object array.

B.7.17.1.18 getSIServices

Prototype: public org.ngb.broadcast.dvb.si.SIService[] getSIServices(int network_id,int original_network_id,int transport_stream_id,int service_id)

Description: Getting the specified service information org.ngb.broadcast.dvb.si.SIService.

Parameter: network_id-Int type, indicating the network identifier, if it is -1, it means to retrieve the network information described in the current and other network NIT tables;

original_network_id – int type, the original network identifier;

transport_stream_id – int type, the transport stream identifier;

service_id – Program identifier, that is, channel program number, if it is -1, it means it does not match the program identifier;

Return: org.ngb.broadcast.dvb.si.SIService object array.

B.7.17.1.19 getPreferredLanguage

Prototype: public java.lang.String getPreferredLanguage()

Description: Getting the default language type for querying SI text information set by the application.

Parameter: None.

Return: A java.lang.String object, indicating the default language type for querying SI text information set by the application. The three-letter language code follows GB/T 4880.2-2000.

B.7.17.1.20 setPreferredLanguage

Prototype: public void setPreferredLanguage(java.lang.String iso639code)

Description: Setting the default language type for the application to query SI text information.

Parameter: iso639code – A java.lang.String object, indicating the language type of SI text information, and the three-letter language code follows GB/T 4880.2-2000.

Return: None.

B.7.18 Class org.ngb.broadcast.dvb.si.SIRequestFailureType

Prototype: public class org.ngb.broadcast.dvb.si.SIRequestFailureType

Description: Reason for failure of PSI/SI information retrieval.

B.7.18.1 Constant field – failure reason

B.7.18.1.1 UNKNOWN

Prototype: public static final int UNKNOWN = 0

Description: Reason for retrieval failure-unknown.

B.7.18.1.2 CANCELED

Prototype: public static final int CANCELED = 1

Description: Reason for failure of retrieval – the request was cancelled by the application.

B.7.18.1.3 DATA_UNAVAILABLE

Prototype: public static final int DATA_UNAVAILABLE = 2

Description: Reason for failure of retrieval – data is unavailable.

B.7.18.1.4 INSUFFICIENT_RESOURCES

Prototype: public static final int INSUFFICIENT_RESOURCES = 3

Description: Reason for failure of retrieval – insufficient resources.

B.7.18.2 Method

B.7.18.2.1 getCode

Prototype: public int getCode()

Description: Getting the reason code of the information retrieval failure.

Parameter: None.

Return: int type, indicating the reason code for the failure of information retrieval,

for the value, see the the constant field definition "Failure Reason" of the org.ngb.broadcast.dvb.si.SIRequestFailureType interface.

B.7.18.2.2 toString

Prototype: public java.lang.String toString()

Description: Getting a text description of the reason for the failure of information retrieval.

Rewrite: toString() method of the java.lang.Object class.

Parameter: None.

Return: A java.lang.String object, indicating a text description of the reason for the failure of information retrieval.

B.7.19 Event org.ngb.broadcast.dvb.si.SIRetrieveEvent

Prototype: public class org.ngb.broadcast.dvb.si.SIRetrieveEvent extends java.util.EventObject

Description: PSI/SI information request event is the base class of a group of events related to the PSI/SI information request defined by this package. Only one such event can be generated in one PSI/SI information request.

B.7.19.1 Method

B.7.19.1.1 getSource

Prototype: public java.lang.Object getSource()

Description: Getting the SIRequest object that generated the event.

Rewrite: getSource() method of the java.util.EventObject class.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.SIRequest object, indicating the SIRequest object that generated the event.

B.7.19.1.2 getAppData

Prototype: public java.lang.Object getAppData()

Description: Getting additional application data information.

Parameter: None.

Return: A java.lang.Object object, indicating additional application data.

B.7.20 Event org.ngb.broadcast.dvb.si.SISuccessRetrieveEvent

Prototype: public class org.ngb.broadcast.dvb.si.SISuccessRetrieveEvent extends org.ngb.broadcast.dvb.si.SIRetrieveEvent

Description: PSI/SI information or descriptor request success event.

B.7.20.1 Method

B.7.20.1.1 getResult

Prototype: public java.util.Enumeration getResult()

Description: Getting the successful retrieval result.

Parameter: None.

Return: A java.util.Enumeration object, indicating the retrieval result. Since both PSI/SI information and descriptor information retrieved through asynchronous methods are returned through this event object, the enumeration object element may be of org.ngb.broadcast.dvb.si.SICommonInformation type or org.ngb.broadcast.dvb.si .SIDescriptor type, the application should further determine the type of the enumeration object element through the instanceof method.

B.7.21 Event org.ngb.broadcast.dvb.si.SIFailureRetrieveEvent

Prototype: public class org.ngb.broadcast.dvb.si.SIFailureRetrieveEvent

extends org.ngb.broadcast.dvb.si.SIRetrieveEvent

Description: PSI/SI information request failure event.

B.7.21.1 Method

B.7.21.1.1 getReason

Prototype: public org.ngb.broadcast.dvb.si.SIRequestFailureType getReason()

Description: Getting the reason for the failure to retrieve PSI/SI information.

Parameter: None.

Return: org.ngb.broadcast.dvb.si.SIRequestFailureType object, indicates the reason for the failure to retrieve PSI/SI information.

B.7.22 Event org.ngb.broadcast.dvb.si.SIUpdateEvent

Prototype: public class org.ngb.broadcast.dvb.si.SIUpdateEvent extends java.util.EventObject

Description: PSI/SI table update event. The application program should first call the getTableID() method to determine which PSI/SI table is updated, and then obtain other parameters according to the specific table type.

B.7.22.1 Method

B.7.22.1.1 getTableID

Prototype: public int getTableID()

Description: Getting the updated PSI/SI table identifier (table_id).

Return: Int type, indicating the updated PSI/SI table identifier.

B.7.22.1.2 getBouquetID

Prototype: public int getBouquetID()

Description: Getting the service group identifier (bouquet_id).

Parameter: None.

Return: Int type, indicating the service group identifier. The return value is meaningful when the BAT is updated.

B.7.22.1.3 getNetworkID

Prototype: public int getNetworkID()

Description: Getting the identifier of the network.

Parameter: None.

Return: Int type, indicating the network identifier. The return value is meaningful when the NIT is updated.

B.7.22.1.4 getOriginalNetworkID

Prototype: public int getOriginalNetworkID()

Description: Getting the original network identifier.

Parameter: None.

Return: Int type, indicating the original network identifier. The return value is meaningful when PMT/SDT/EIT is updated.

B.7.22.1.5 getServiceID

Prototype: public int getServiceID()

Description: Getting the service identifier.

Return: Int type, indicating the service identifier (service_id). The return value is meaningful when the PMT/EIT is updated.

B.7.22.1.6 getTransportStreamID

Prototype: public int getTransportStreamID()

Description: Getting the transport stream identifier.

Parameter: None.

Return: Int type, indicating the transport stream identifier. The return value is meaningful when PAT/PMT/SDT/EIT is updated.

B.7.23 Exception org.ngb.broadcast.dvb.si.InvalidPeriodException

Prototype: public class org.ngb.broadcast.dvb.si.InvalidPeriodException

extends java.lang.Exception

Description: Invalid period exception. When the specified date is invalid, the exception is thrown.

Annex C

JAVA-Two-way broadband network access unit

(This annex forms an integral part of this Recommendation.)

C.1 Overview

This annex defines the JAVA interface related to two-way broadband network access, including Ethernet management module and WiFi management module.

C.2 Ethernet management module

The Ethernet management module provides DHCP information and management information for the Ethernet.

The summary of the Ethernet management module is shown in Table C.1.

Table C.1 – Summary of Ethernet management module

Interface	
Listener	Ethernet status change event listener, implemented by the application layer
Class	
DhcpInfo	Provide a method to obtain DHCP configuration information.
EthernetManager	Provide a method for Ethernet management.

C.2.1 Interface org.tvos.net.Listener

Prototype: public interface org.tvos.net.Listener

Description: The Ethernet status change event listener is implemented by the application layer.

C.2.1.1 Method

C.2.1.1.1 onAvailabilityChanged

Prototype: public void onAvailabilityChanged(boolean isAvailable)

Description: The interface is notified of whether the Ethernet is available.

Parameter: isAvailable – Boolean type, true indicating available, false indicating unavailable.

Return: None.

C.2.2 Class org.tvos.net.DhcpInfo

Prototype: public class org.tvos.net.DhcpInfo implements org.tvos.os.Parcelable

Description: A DHCP configuration information class, providing a method for obtaining DHCP configuration information.

C.2.2.1 Attributes

C.2.2.1.1 ipAddress

Prototype: public int ipAddress

Description: Indicating IP address information.

C.2.2.1.2 gateway

Prototype: public int gateway

Description: Indicating gateway information.

C.2.2.1.3 dns1

Prototype: public int dns1

Description: Indicating dns1 information.

C.2.2.1.4 dns2

Prototype: public int dns2

Description: Indicating dns2 information.

C.2.2.1.5 serverAddress

Prototype: public int serverAddress

Description: Indicating serverAddress information.

C.2.3 Class org.tvos.net.EthernetManager

Prototype: public org.tvos.net.EthernetManager(org.tvos.content.Context context, org.tvos.net.IEthernetManager service)

Description: An Ethernet connection management class, providing a method of connecting to the Ethernet network.

C.2.3.1 Constant field

C.2.3.1.1 ETHERNET_STATE_DISABLED

Prototype: public static final int ETHERNET_STATE_DISABLED = 0

Description: Ethernet disable constant definition.

C.2.3.1.2 ETHERNET_STATE_ENABLED

Prototype: public static final int ETHERNET_STATE_ENABLED = 1

Description: Ethernet enable constant definition.

C.2.3.1.3 ETHERNET_STATE_UNKNOWN

Prototype: public static final int ETHERNET_STATE_UNKNOWN = 2

Description: Ethernet UNKNOWND constant definition, usually represented in the initialized phase.

C.2.3.1.4 EVENT_DHCP_CONNECT_SUCCESSED

Prototype: public static final int EVENT_DHCP_CONNECT_SUCCESSED = 10

Description: DHCP connection is successful.

C.2.3.1.5 EVENT_DHCP_CONNECT_FAILED

Prototype: public static final int EVENT_DHCP_CONNECT_FAILED = 11

Description: DHCP connection failed.

C.2.3.2 Method

C.2.3.2.1 isAvailable

Prototype: public boolean isAvailable()

Description: Whether the current Ethernet interface is available.

Parameter: None.

Return: Boolean type, true indicating available, false indicating unavailable.

C.2.3.2.2 addListener

Prototype: public void addListener(org.tvos.net.Listener listener)

Description: Add Ethernet status change event listening interface.

Parameter: listener – org.tvos.net.Listener type, listening callback function, implemented by the application.

Return: None.

C.2.3.2.3 removeListener

Prototype: public void removeListener(org.tvos.net.Listener listener)

Description: Delete the Ethernet status change event listening interface.

Parameter: listener – org.tvos.net.Listener type, listening callback function, implemented by the application.

Return: None.

C.2.3.2.4 getConfiguration

Prototype: public org.tvos.net.IpConfiguration getConfiguration()

Description: Getting network configuration information.

Parameter: None.

Return: An org.tvos.net.IpConfiguration object, indicating network configuration information.

C.2.3.2.5 setConfiguration

Prototype: public void setConfiguration(org.tvos.net.IpConfiguration config)

Description: Setting the network configuration information.

Parameter: A config – org.tvos.net.IpConfiguration object, indicating the configuration information of the network.

Return: None.

C.2.3.2.6 setEthernetEnabled

Prototype: public void setEthernetEnabled(boolean enable)

Description: Setting Ethernet enable.

Parameter: enable-boolean type, true indicating enable; false indicating disable.

Return: None.

C.2.3.2.7 getEthernetState

Prototype: public int getEthernetState()

Description: Getting the Ethernet status.

Parameter: None.

Return: Int type, indicating the Ethernet status.

– ETHERNET_STATE_DISABLED, Ethernet disable status;
– ETHERNET_STATE_ENABLED, Ethernet enabling status;
– ETHERNET_STATE_UNKNOWN, unknown status.

C.2.3.2.8 getDhcpInfo

Prototype: public org.tvos.net.DhcpInfo getDhcpInfo()

Description: Getting Ethernet DHCP information.

Parameter: None.

Return: An org.tvos.net.DhcpInfo object, indicating DHCP information of the Ethernet.

C.2.3.2.9 getNetLinkStatus

Prototype: public boolean getNetLinkStatus()

Description: Getting physical link status of the network.

Parameter: None.

Return: Boolean type, true indicating connected, false indicating not connected.

C.2.3.2.10 getNetLinkStatus

Prototype: public int getNetLinkStatus()

Description: Specify the network port to get the physical link status of the network.

Parameter: None.

Return: int type, 1 indicating connected; 0 indicating not connected; -1 indicating the specified network port is not found.

C.2.3.2.11 getInterfaceName

Prototype: public java.lang.String getInterfaceName()

Description: Getting the network name.

Parameter: None.

Return: A java.lang.String type, indicating the network name.

C.2.3.2.12 setEthernetMode

Prototype: public void setEthernetMode(java.lang.String mode, org.tvos.net.DhcpInfo dhcpInfo)

Description: Setting the network mode.

Parameter: mode – java.lang.String type, can have four values: ETHERNET_CONNECT_MODE_DHCP, ETHERNET_CONNECT_MODE_MANUAL, ETHERNET_CONNECT_MODE_PPPOE, ETHERNET_CONNECT_MODE_NONE;

A dhcpInfo-org.tvos.net.DhcpInfo object, indicating DHCP information.

Return: None.

C.2.3.2.13 getEthernetMode

Prototype: public java.lang.String getEthernetMode()

Description: Getting the network mode.

Parameter: None.

Return: java.lang.Sting type, indicating the network mode, which can have four values: ETHERNET_CONNECT_MODE_DHCP, ETHERNET_CONNECT_MODE_MANUAL, ETHERNET_CONNECT_MODE_PPPOE, and ETHERNET_CONNECT_MODE_NONE.

C.2.3.2.14 getDeviceNameList

Prototype: public java.lang.String[] getDeviceNameList()

Description: Getting the name of the available physical network port.

Parameter: None.

Return: A java.lang.String array, indicating the list of network port names.

C.2.3.2.15 getTotalInterface

Prototype: public int getTotalInterface()

Description: Getting the number of available physical network ports.

Parameter: None.

Return: Int type, indicating the number of network ports.

C.2.3.2.16 enableEthernet

Prototype: public void enableEthernet(boolean enable)

Description: Enabled network.

Parameter: enable-boolean type, true indicating enable; false indicating disable.

Return: None.

C.2.3.2.17 setInterfaceName

Prototype: public boolean setInterfaceName(java.lang.String iface)

Description: Setting the name of the network port.

Parameter: iface-java.lang.String type, the value being eth0, eth1, etc.

Return: Boolean type, true indicating success; false indicating failure.

C.2.3.2.18 getDhcpOption60State

Prototype: public int getDhcpOption60State()

Description: Getting the status of Dhcp Option60/Option61.

Parameter: None.

Return: int type, can take the following three values OPTION60_STATE_DISABLED, OPTION60_STATE_ENABLED, OPTION60_STATE_UNKNOWN.

C.2.3.2.19 getDhcpOption60Login

Prototype: public java.lang.String getDhcpOption60Login()

Description: Getting the username of Dhcp Option61.

Parameter: None.

Return: java.lang.String type, indicating the Option 61 username.

C.2.3.2.20 getDhcpOption60Password

Prototype: public java.lang.String getDhcpOption60Password()

Description: Getting the Dhcp Option60 password.

Parameter: None.

Return: java.lang.String type, indicating the password.

C.2.3.2.21 setDhcpOption60

Prototype: public void setDhcpOption60(boolean enable, java.lang.String login, java.lang.String password)

Description: Setting the username and password of Dhcp Option60.

Parameter: enable – boolean type, indicating enable or disable;

login – java.lang.String type, indicating the username;

password – java.lang.String, indicating the password.

Return: None.

C.2.3.2.22 getDhcpOption125State

Prototype: public int getDhcpOption125State()

Description: Getting the status of DHCP Option125.

Parameter: None.

Return: int type, the value being OPTION125_STATE_DISABLED or OPTION125_STATE_ENABLED or OPTION125_STATE_UNKNOWN.

C.2.3.2.23 getDhcpOption125Info

Prototype: public java.lang.String getDhcpOption125Info()

Description: Getting DHCP Option125 information.

Parameter: None.

Return: java.lang.String type, indicating DHCP Option125 information.

C.2.3.2.24 getDhcpOption125Info

Prototype: public void setDhcpOption125(boolean enable, java.lang.String option125Info)

Description: Setting DHCP Option125 information.

Parameter: enable-boolean type, indicating enable or disable.

option125Info – java.lang.String type, indicating Option125 information.

Return: None.

C.2.3.2.25 enableIpv6

Prototype: public void enableIpv6(boolean enable)

Description: Enable IPv6.

Parameter: enable – boolean type, indicating enable or disable.

Return: None.

C.2.3.2.26 getIpv6PersistedState

Prototype: public int getIpv6PersistedState()

Description: Getting DHCPv6 status.

Parameter: None.

Return: int type, the value being DHCPV6_STATE_ENABLED, DHCPV6_STATE_DISABLED, DHCPV6_STATE_UNKNOWN.

C.2.3.2.27 setEthernetMode6

Prototype: public void setEthernetMode6(java.lang.String mode)

Description: Setting the mode for obtaining IPv6 addresses.

Parameter: mode-java.lang.String type, the value being ETHERNET_CONNECT_MODE_DHCP or ETHERNET_CONNECT_MODE_MANUAL.

Return: None.

C.3 WiFi management module

The WiFi management module provides classes and methods related to WiFi network interface control.

The summary of WiFi management module is shown in Table C.2.

Table C.2 – Summary of WiFi management module

Interface	
ActionListener	WiFi status change event listener is implemented by the application layer.
Class	
WifiInfo	An WiFi connection information class, providing a method to obtain WiFi connection information.
WifiManager	An WiFi manager, providing the management function of WiFi wireless network: 1) Find, scan and manage currently available wireless network access points (AP); 2) Manage currently active network access links, such as establishing connection, disconnecting connection, disabling connection, deleting connection, etc.
ScanResult	WiFi scan result describes a connection access point information found by the WiFi scan.

C.3.1 Interface org.tvos.net.wifi.ActionListener

Prototype: public interface org.tvos.net.wifi.ActionListener

Description: WiFi status change event listener interface is implemented by the application layer.

C.3.1.1 Method

C.3.1.1.1 onSuccess

Prototype: public void onSuccess()

Description: WiFi connection is successful.

Parameter: None.

Return: None.

C.3.1.1.2 onFailure

Prototype: public void onFailure(int reason)

Description: WiFi connection failed.

Parameter: reason – int type, indicating the reason for the connection failure. For the value, please refer to the constant field definition of the "WiFi Status" of WifiManager.

Return: None.

C.3.2 Class org.tvos.net.wifi.WifiInfo

Prototype: public class org.tvos.net.wifi.WifiInfo implements org.tvos.os.Parcelable

Description: WiFi connection information class, describes the WiFi connection status.

C.3.2.1 Method

C.3.2.1.1 getMacAddress

Prototype: public java.lang.String getMacAddress()

Description: Getting the hotspot routing MAC address.

Parameter: None.

Return: java.lang.String object, indicating the hotspot routing MAC address.

C.3.2.1.2 getSSID

Prototype: public java.lang.String getSSID()

Description: Getting the SSID of the current network connection.

Parameter: None.

Return: java.lang.String object, indicating the SSID of the current network connection.

C.3.2.1.3 isHiddenSSID

Prototype: public boolean isHiddenSSID()

Description: Determine whether the current connection hides the SSID (that is, does not broadcast the SSID).

Parameter: None.

Return: boolean type, indicating whether the connection is to hide the SSID, true value indicating that the SSID is hidden, and the false value indicating that the SSID is broadcast.

C.3.2.1.4 getLinkSpeed

Prototype: public int getLinkSpeed()

Description: Getting the speed of the network connection.

Parameter: None.

Return: Int type, indicating the network connection speed in megabits per second (Mb/s).

C.3.2.1.5 getNetworkId

Prototype: public int getNetworkId()

Description: Getting the identifier number of the current network connection.

Parameter: None.

Return: Int type, indicating the network identifier of the current network connection.

C.3.2.1.6 getBSSID

Prototype: public java.lang.String getBSSID()

Description: Getting the BSSID of the access point.

Parameter: None.

Return: java.lang.String type, returning the BSSID of the access point.

C.3.2.1.7 getIpAddress

Prototype: public int getIpAddress()

Description: Getting the ip address of the current connection.

Parameter: None.

Return: Int type, indicating the IP address of the current connection.

C.3.2.1.8 getFrequency

Prototype: public int getFrequency()

Description: Getting the transmit frequency of the current connection.

Parameter: None.

Return: Int type, indicating the transmitting frequency of the current connection, in MHz.

C.3.3 Class org.tvos.net.wifi.WifiManager

Prototype: public class org.tvos.net.wifi.WifiManager

Description: A WiFi manager, providing the management function of WiFi wireless network:

1) Find, scan and manage currently available wireless network access points (AP);
2) Manage the currently active network access links, such as establishing a connection, disconnecting the connection, disabling the connection, and deleting the connection.

C.3.3.1 Constant field – WiFi status

C.3.3.1.1 ERROR_AUTHENTICATING

Prototype: public static final int ERROR_AUTHENTICATING = 1

Description: The error code returned when authentication fails.

C.3.3.1.2 WIFI_STATE_DISABLING

Prototype: public static final int WIFI_STATE_DISABLING = 0

Description: Intermediate state when wifi is disabled.

C.3.3.1.3 WIFI_STATE_DISABLED

Prototype: public static final int WIFI_STATE_DISABLED = 1

Description: WiFi disabled state.

C.3.3.1.4 WIFI_STATE_ENABLING

Prototype: public static final int WIFI_STATE_ENABLING = 2

Description: Intermediate state of wifi enabled.

C.3.3.1.5 WIFI_STATE_ENABLED

Prototype: public static final int WIFI_STATE_ENABLED = 3

Description: WiFi enabled state.

C.3.3.1.6 WIFI_STATE_UNKNOWN

Prototype: public static final int WIFI_STATE_UNKNOWN = 4

Description: wifi unknown status. When returning to this status, it means that there is usually an error.

C.3.3.2 Method

C.3.3.2.1 WifiManager

Prototype: public WifiManager(org.tvos.content.Context context, org.tvos.net.wifi.IWifiManager service)

Description: Getting an example of the WiFi wireless network manager implemented by the system.

Parameter: context-org.tvos.content.Context object, indicating the context of the application;

service-org.tvos.net.wifi.Iwifimanager object, indicating WiFi service.

Return: None.

C.3.3.2.2 getConfiguredNetworks

Prototype: public List<org.tvos.net.wifi.WifiConfiguration> getConfiguredNetworks()

Description: Getting the currently available WiFi network connection configuration.

Parameter: None.

Return: org.tvos.net.wifi.WifiConfiguration object array, indicating the currently available WiFi network connection configuration. If there is no network connection configuration available, the length of the returned array is 0.

C.3.3.2.3 addNetwork

Prototype: public int addNetwork(org.tvos.net.wifi.WifiConfiguration config)

Description: Add a network configuration. The application can get the available network configuration through the getConfiguredNetworks method, and select a configuration to set the connection password, and re-add it to the WiFi manager.

Parameter: A wifiConfiguration-org.tvos.net.wifi.WifiConfiguration object, indicating the network configuration object.

Return: int type, returning the network configuration identifier (ID) if it succeeds, and returns -1 if it fails.

C.3.3.2.4 removeNetwork

Prototype: public boolean removeNetwork(int netId)

Description: Remove a specified network configuration.

Parameter: networkId – Int type, indicating the network configuration identifier.

Return: None.

C.3.3.2.5 enableNetwork

Prototype: public boolean enableNetwork(int netId, boolean disableOthers)

Description: Enable a certain network connection. Associate WiFi to a specified network connection configuration and start the network connection at the same time. Call this interface to generate network connection status change events.

Parameter: networkId – Int type, indicating the identifier of a certain network connection, obtained from the WifiConfiguration object;

disableOthers – boolean type, indicating whether to disable other networks, true value indicating that other networks are disabled, and false indicating that the use of other networks is not affected.

Return: None.

C.3.3.2.6 disableNetwork

Prototype: public boolean disableNetwork(int netId)

Description: Disable a certain network connection, the network connection configuration is no longer a valid candidate connection. Call this interface to generate network connection status change events.

Parameter: networkId –Int type, indicating the identifier of a certain network connection, which is obtained from the WifiConfiguration object.

Return: None.

C.3.3.2.7 getConnectionInfo

Prototype: public org.tvos.net.wifi.WifiInfo getConnectionInfo()

Description: Getting current connection information.

Parameter: None.

Return: An org.tvos.net.wifi.WifiInfo object, indicating the current connection status information.

C.3.3.2.8 isWifiEnabled

Prototype: public boolean isWifiEnabled()

Description: Check whether WiFi is turned on.

Parameter: None.

Return: boolean type, true value indicating that WiFi is on, and false indicating that WiFi is off.

C.3.3.2.9 setWifiEnabled

Prototype: public boolean setWifiEnabled(boolean enabled)

Description: Setting WiFi on or off.

Parameter: enabled-boolean type, true value indicating that WiFi is turned on, and false value indicating that WiFi is turned off.

Return: boolean type, indicating the result of the operation, true value indicating that the operation is successful, and false value indicating that the operation fails.

C.3.3.2.10 saveConfiguration

Prototype: public boolean saveConfiguration()

Description: Saving the network configuration. This interface informs the bottom layer to save the currently configured network hotspot information so that it can be automatically connected when it is turned on next time.

Parameter: None.

Return: None.

C.3.3.2.11 getScanResult

Prototype: public List< org.tvos.net.wifi.ScanResult> getScanResults()

Description: Getting the last scan result.

Parameter: None.

Return: An org.tvos.net.wifi.ScanResult object array, indicating the result of the last scan. If there is no scan result, the length of the returned array is 0.

C.3.3.2.12 startScan

Prototype: public boolean startScan()

Description: The method returns immediately, and the scan result is returned to the caller in the form of an event.

Parameter: None.

Return: boolean type, true value indicating that the scan started successfully; false indicating that the scan started failed.

C.3.3.2.13 disconnect

Prototype: public boolean disconnect()

Description: Disconnect from the current network connection point. Call this interface to generate network connection status change events.

Parameter: None.

Return: None.

C.3.3.2.14 reassociate

Prototype: public boolean reassociate()

Description: Re-associate the current network connection point, even if it is connected to the current network. Call this interface to generate network connection status change events.

Parameter: None.

Return: None.

C.3.3.2.15 reconnect

Prototype: public boolean reconnect()

Description: If the current network connection has been disconnected, reconnect to the current network. Call this interface to generate network connection status change events.

Parameter: None.

Return: None.

C.3.3.2.16 getDhcpInfo

Prototype: public org.tvos.net.DhcpInfo getDhcpInfo()

Description: Obtain the result information from the last DHCP request.

Parameter: None.

Return: org.tvos.net.DhcpInfo object.

C.3.3.2.17 getWifiState

Prototype: public int getWifiState()

Description: Getting the status of wifi.

Parameter: None.

Return: See the constant field definition, which can take the following values:

WIFI_STATE_DISABLING, WIFI_STATE_ENABLED, WIFI_STATE_ENABLING, WIFI_STATE_UNKNOWN.

C.3.4 Class org.tvos.net.wifi.ScanResult

Prototype: public class org.tvos.net.wifi.ScanResult implements org.tvos.os.Parcelable

Description: Describes the information of a connection access point in the WiFi scan result.

C.3.4.1 Attributes

C.3.4.1.1 SSID

Prototype: public java.lang.String SSID

Description: WiFi wireless network connection access point name.

C.3.4.1.2 BSSID

Prototype: public java.lang.String BSSID

Description: WiFi wireless network connection access point address.

C.3.4.1.3 capabilities

Prototype: public java.lang.String capabilities

Description: Ability configuration of WiFi wireless network connection access point.

C.3.4.1.4 frequency

Prototype: public int frequency

Description: Transmit frequency of the WiFi wireless network connection access point, in megahertz (MHz).

C.3.4.1.5 level

Prototype: public int level

Description: Signal strength of the WiFi wireless network connection access point, in dBm.

Annex D

JAVA-Human-computer interaction unit

(This annex forms an integral part of this Recommendation.)

D.1 Overview

This annex defines the JAVA interface definition related to human-computer interaction.

D.2 Human-computer interaction module

The human-computer interaction module provides classes and methods related to human-computer interaction, including user input and output.

The summary of the human-computer interaction module is shown in Table D.1.

Table D.1 – Summary of human-computer interaction module

Interface	
UserInput	The user enters the message definition.
NgbKeyListener	The key event listener interface is implemented by the application program that need to listen to KeyEvent.
NgbMouseListener	The mouse event listener interface is implemented by the application program that need to listen to MouseEvent.
NgbVoiceListener	The voice event listener interface is implemented by the application program that need to listen to voice recognition.
Class	
FrontPanel	Front panel information display output control, including LED indicator and LED digital tube display control.
NgbInputManager	Input control manager, used to listen and receive events such as keystrokes, mouse, and injection control.
NgbVoiceManager	The voice-related control manager is used for the control and realization of related functions such as voice recognition and voice broadcast.
Event	
NgbInputEvent	The NGB input event class is the base class of other NGB extended input events in this package.
KeyEvent	The key event class inherits the NgbInputEvent class.
MouseEvent	The mouse event class inherits the NgbInputEvent class.

User input refers to the user sending user instructions to the receiving terminal through some input devices such as remote control, mouse, keyboard, front panel keys, etc. These user instructions are uniformly packaged into key messages for processing. This specification stipulates the key values of the keys:

– The key values of the mouse and keyboard are compatible with KeyEvent and MouseEvent;

– The key values of the remote control and the front panel are partly unified with the key values of the keyboard, and the rest are defined by this specification.

The user input message contains an indication of the source of the message to distinguish the source of the message.

User output refers to the receiving terminal to feedback information to the user through the front panel or display screen. This specification mainly stipulates the information output of the front panel.

D.2.1 Interface org.ngb.interact.UserInput

Prototype: public interface org.ngb.interact.UserInput

Description: User input interface.

D.2.1.1 Constant field – user input message source

D.2.1.1.1 INPUT_UNKNOWN

Prototype: public static final int INPUT_UNKNOWN = 0x0(0)

Description: Source of key message-indicating unknown.

D.2.1.1.2 INPUT_FRONTPANEL

Prototype: public static final int INPUT_FRONTPANEL = 0x1(1)

Description: Source of key message-indicating the key input on the front panel of the receiver.

D.2.1.1.3 INPUT_KEYBOARD

Prototype: public static final int INPUT_KEYBOARD = 0x2(2)

Description: Source of key message-indicating the PC keyboard key input.

D.2.1.1.4 INPUT_MOUSE

Prototype: public static final int INPUT_MOUSE = 0x3(3)

Description: Source of key message-indicating mouse key input.

D.2.1.1.5 INPUT_REMOTECTRL

Prototype: public static final int INPUT_REMOTECTRL = 0x4(4)

Description: Source of key message-indicating the remote control key input.

D.2.1.1.6 INPUT_USER

Prototype: public static final int INPUT_USER = 0x5(5)

Description: Source of key message-indicating user input.

D.2.1.2 Constant field – keyboard key status

D.2.1.2.1 KEY_TYPED

Prototype: public static final int KEY_TYPED = 0x0190(400)

Description: Key state constant – indicating the keystroke status, compatible with KeyEvent.KEY_TYPED.

D.2.1.2.2 KEY_PRESSED

Prototype: public static final int KEY_PRESSED = 0x0191(401)

Description: Key state constant – indicating the pressed state, compatible with KeyEvent.KEY_PRESSED.

D.2.1.2.3 KEY_RELEASED

Prototype: public static final int KEY_RELEASED = 0x0192(402)

Description: Key state constant – indicating the release state, compatible with KeyEvent.KEY_RELEASED.

D.2.1.3 Constant field – mouse pressed state

D.2.1.3.1 MOUSE_PRESSED

Prototype: public static final int MOUSE_PRESSED = 0x01F5(501)

Description: Mouse key state constant – indicating the pressed state, compatible with MouseEvent.MOUSE_PRESSED.

D.2.1.3.2 MOUSE_RELEASED

Prototype: public static final int MOUSE_RELEASED = 0x01F6(502)

Description: Mouse key state constant – indicating the released state, compatible with MouseEvent.MOUSE_RELEASED.

D.2.1.3.3 MOUSE_MOVED

Prototype: public static final int MOUSE_MOVED = 0x01F7(503)

Description: Mouse key state constant – indicating moving status, compatible with MouseEvent.MOUSE_MOVED.

D.2.1.4 Constant field – mouse button message code

D.2.1.4.1 MOUSE_NOBUTTON

Prototype: public static final int MOUSE_NOBUTTON = 0x00(0)

Description: Mouse button message code – indicating an invalid button, compatible with MouseEvent.NOBUTTON.

D.2.1.4.2 MOUSE_LBUTTON

Prototype: public static final int MOUSE_LBUTTON = 0x01(1)

Description: Mouse button message code – indicating the left mouse button, compatible with MouseEvent.BUTTON1.

D.2.1.4.3 MOUSE_MBUTTON

Prototype: public static final int MOUSE_MBUTTON = 0x02(2)

Description: Mouse button message code – indicating the middle mouse button, compatible with MouseEvent.BUTTON2.

D.2.1.4.4 MOUSE_RBUTTON

Prototype: public static final int MOUSE_RBUTTON = 0x03(3)

Description: Mouse button message code – indicating the right mouse button, compatible with MouseEvent.BUTTON3.

D.2.1.5 Constant field – button message code

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
0	KEYCODE_UNKNOWN	unknown
1	KEYCODE_SOFT_LEFT	
2	KEYCODE_SOFT_RIGHT	
3	KEYCODE_HOME	Home

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
4	KEYCODE_BACK	return
5	KEYCODE_CALL	dial
6	KEYCODE_ENDCALL	Hanging-up
7	KEYCODE_0	0
8	KEYCODE_1	1
9	KEYCODE_2	2
10	KEYCODE_3	3
11	KEYCODE_4	4
12	KEYCODE_5	5
13	KEYCODE_6	6
14	KEYCODE_7	7
15	KEYCODE_8	8
16	KEYCODE_9	9
17	KEYCODE_STAR	*
18	KEYCODE_POUND	#
19	KEYCODE_DPAD_UP	Navigation key up
20	KEYCODE_DPAD_DOWN	Navigation key down
21	KEYCODE_DPAD_LEFT	Navigation key left
22	KEYCODE_DPAD_RIGHT	Navigation key right
23	KEYCODE_DPAD_CENTER	Navigation key middle
24	KEYCODE_VOLUME_UP	Volume up
25	KEYCODE_VOLUME_DOWN	Volume down
26	KEYCODE_POWER	Power
27	KEYCODE_CAMERA	Camera
28	KEYCODE_CLEAR	
29	KEYCODE_A	A
30	KEYCODE_B	B
31	KEYCODE_C	C
32	KEYCODE_D	D
33	KEYCODE_E	E
34	KEYCODE_F	F
35	KEYCODE_G	G
36	KEYCODE_H	H
37	KEYCODE_I	I
38	KEYCODE_J	J
39	KEYCODE_K	K
40	KEYCODE_L	L
41	KEYCODE_M	M
42	KEYCODE_N	N

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
43	KEYCODE_O	O
44	KEYCODE_P	P
45	KEYCODE_Q	Q
46	KEYCODE_R	R
47	KEYCODE_S	S
48	KEYCODE_T	T
49	KEYCODE_U	U
50	KEYCODE_V	V
51	KEYCODE_W	W
52	KEYCODE_X	X
53	KEYCODE_Y	Y
54	KEYCODE_Z	Z
55	KEYCODE_COMMA	,
56	KEYCODE_PERIOD	.
57	KEYCODE_ALT_LEFT	Left ALT
58	KEYCODE_ALT_RIGHT	Right ALT
59	KEYCODE_SHIFT_LEFT	Left SHIFT
60	KEYCODE_SHIFT_RIGHT	Right SHIFT
61	KEYCODE_TAB	Tab
62	KEYCODE_SPACE	Space
63	KEYCODE_SYM	Symbol modifier
64	KEYCODE_EXPLORER	Browse
65	KEYCODE_ENVELOPE	Mail
66	KEYCODE_ENTER	Enter
67	KEYCODE_DEL	Backspace
68	KEYCODE_GRAVE	`
69	KEYCODE_MINUS	–
70	KEYCODE_EQUALS	=
71	KEYCODE_LEFT_BRACKET	[
72	KEYCODE_RIGHT_BRACKET	]
73	KEYCODE_BACKSLASH	\
74	KEYCODE_SEMICOLON	;
75	KEYCODE_APOSTROPHE	'
76	KEYCODE_SLASH	/
77	KEYCODE_AT	@
78	KEYCODE_NUM	Num
79	KEYCODE_HEADSETHOOK	Headphone answer key
80	KEYCODE_FOCUS	Camera focus
81	KEYCODE_PLUS	+

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
82	KEYCODE_MENU	Menu
83	KEYCODE_NOTIFICATION	Notification
84	KEYCODE_SEARCH	Search
85	KEYCODE_MEDIA_PLAY_PAUSE	Multimedia button Play/Pause
86	KEYCODE_MEDIA_STOP	Multimedia button stop
87	KEYCODE_MEDIA_NEXT	Multimedia button next song
88	KEYCODE_MEDIA_PREVIOUS	Multimedia button Previous song
89	KEYCODE_MEDIA_REWIND	Multimedia button Fast Backward
90	KEYCODE_MEDIA_FAST_FORWARD	Multimedia button Fast Forward
91	KEYCODE_MUTE	Mute the microphone
92	KEYCODE_PAGE_UP	Page up
93	KEYCODE_PAGE_DOWN	Page down
94	KEYCODE_PICTSYMBOLS	Picture Symbols modifier
95	KEYCODE_SWITCH_CHARSET	Switch Charset modifier
96	KEYCODE_BUTTON_A	Gamepad button A
97	KEYCODE_BUTTON_B	Gamepad button B
98	KEYCODE_BUTTON_C	Gamepad button C
99	KEYCODE_BUTTON_X	Gamepad button X
100	KEYCODE_BUTTON_Y	Gamepad button Y
101	KEYCODE_BUTTON_Z	Gamepad button Z
102	KEYCODE_BUTTON_L1	Gamepad button L1
103	KEYCODE_BUTTON_R1	Gamepad button L2
104	KEYCODE_BUTTON_L2	Gamepad button R1
105	KEYCODE_BUTTON_R2	Gamepad button R2
106	KEYCODE_BUTTON_THUMBL	Left Thumb Button
107	KEYCODE_BUTTON_THUMBR	Right Thumb Button
108	KEYCODE_BUTTON_START	Gamepad button Start
109	KEYCODE_BUTTON_SELECT	Gamepad button Select
110	KEYCODE_BUTTON_MODE	Gamepad button Mode
111	KEYCODE_ESCAPE	ESC key
112	KEYCODE_FORWARD_DEL	Delete
113	KEYCODE_CTRL_LEFT	Left CTRL
114	KEYCODE_CTRL_RIGHT	Right CTRL
115	KEYCODE_CAPS_LOCK	Caps Lock
116	KEYCODE_SCROLL_LOCK	Scroll Lock
117	KEYCODE_META_LEFT	Left meta
118	KEYCODE_META_RIGHT	Right meta
119	KEYCODE_FUNCTION	Fn
120	KEYCODE_SYSRQ	System request/Screenshot

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
121	KEYCODE_BREAK	Rest/Pause
122	KEYCODE_MOVE_HOME	Move the cursor to the beginning
123	KEYCODE_MOVE_END	Move the cursor to the end
124	KEYCODE_INSERT	Insert
125	KEYCODE_FORWARD	Move the cursor forward
126	KEYCODE_MEDIA_PLAY	Multimedia button Play
127	KEYCODE_MEDIA_PAUSE	Multimedia button Pause
128	KEYCODE_MEDIA_CLOSE	Multimedia button Close
129	KEYCODE_MEDIA_EJECT	Multimedia button Eject
130	KEYCODE_MEDIA_RECORD	Multimedia button Record
131	KEYCODE_F1	F1
132	KEYCODE_F2	F2
133	KEYCODE_F3	F3
134	KEYCODE_F4	F4
135	KEYCODE_F5	F5
136	KEYCODE_F6	F6
137	KEYCODE_F7	F7
138	KEYCODE_F8	F8
139	KEYCODE_F9	F9
140	KEYCODE_F10	F10
141	KEYCODE_F11	F11
142	KEYCODE_F12	F12
143	KEYCODE_NUM_LOCK	Numeric keypad lock
144	KEYCODE_NUMPAD_0	Numeric keypad 0
145	KEYCODE_NUMPAD_1	Numeric keypad 1
146	KEYCODE_NUMPAD_2	Numeric keypad 2
147	KEYCODE_NUMPAD_3	Numeric keypad 3
148	KEYCODE_NUMPAD_4	Numeric keypad 4
149	KEYCODE_NUMPAD_5	Numeric keypad 5
150	KEYCODE_NUMPAD_6	Numeric keypad 6
151	KEYCODE_NUMPAD_7	Numeric keypad 7
152	KEYCODE_NUMPAD_8	Numeric keypad 8
153	KEYCODE_NUMPAD_9	Numeric keypad 9
154	KEYCODE_NUMPAD_DIVIDE	Numeric keypad /
155	KEYCODE_NUMPAD_MULTIPLY	Numeric keypad *
156	KEYCODE_NUMPAD_SUBTRACT	Numeric keypad -
157	KEYCODE_NUMPAD_ADD	Numeric keypad +
158	KEYCODE_NUMPAD_DOT	Numeric keypad.
159	KEYCODE_NUMPAD_COMMA	Numeric keypad,

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
160	KEYCODE_NUMPAD_ENTER	Numeric keypad Enter
161	KEYCODE_NUMPAD_EQUALS	Numeric keypad =
162	KEYCODE_NUMPAD_LEFT_PAREN	Numeric keypad (
163	KEYCODE_NUMPAD_RIGHT_PAREN	Numeric keypad)
164	KEYCODE_VOLUME_MUTE	Mute speaker
165	KEYCODE_INFO	Info
166	KEYCODE_CHANNEL_UP	Channel up
167	KEYCODE_CHANNEL_DOWN	Channel down
168	KEYCODE_ZOOM_IN	Zoom in
169	KEYCODE_ZOOM_OUT	Zoom out
170	KEYCODE_TV	TV button Live
171	KEYCODE_WINDOW	TV button Picture in Picture
172	KEYCODE_GUIDE	TV button Program guide
173	KEYCODE_DVR	TV button recording
174	KEYCODE_BOOKMARK	TV button bookmark
175	KEYCODE_CAPTIONS	Text subtitle switch
176	KEYCODE_SETTINGS	System settings
177	KEYCODE_TV_POWER	TV button Power switch
178	KEYCODE_TV_INPUT	TV button Input source
179	KEYCODE_STB_POWER	TV button External set-top box power switch
180	KEYCODE_STB_INPUT	TV button External set-top box input source
181	KEYCODE_AVR_POWER	TV button Home theater power switch
182	KEYCODE_AVR_INPUT	TV button Home theater input source
183	KEYCODE_PROG_RED	TV button Red
184	KEYCODE_PROG_GREEN	TV button Green
185	KEYCODE_PROG_YELLOW	TV button Yellow
186	KEYCODE_PROG_BLUE	TV button Blue
187	KEYCODE_APP_SWITCH	Application switch key
188	KEYCODE_BUTTON_1	Universal gamepad button #1
189	KEYCODE_BUTTON_2	Universal gamepad button #2
190	KEYCODE_BUTTON_3	Universal gamepad button #3
191	KEYCODE_BUTTON_4	Universal gamepad button #4
192	KEYCODE_BUTTON_5	Universal gamepad button #5
193	KEYCODE_BUTTON_6	Universal gamepad button #6
194	KEYCODE_BUTTON_7	Universal gamepad button #7
195	KEYCODE_BUTTON_8	Universal gamepad button #8
196	KEYCODE_BUTTON_9	Universal gamepad button #9

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
197	KEYCODE_BUTTON_10	Universal gamepad button #10
198	KEYCODE_BUTTON_11	Universal gamepad button #11
199	KEYCODE_BUTTON_12	Universal gamepad button #12
200	KEYCODE_BUTTON_13	Universal gamepad button #13
201	KEYCODE_BUTTON_14	Universal gamepad button #14
202	KEYCODE_BUTTON_15	Universal gamepad button #15
203	KEYCODE_BUTTON_16	Universal gamepad button #16
204	KEYCODE_LANGUAGE_SWITCH	Input method language switch key
205	KEYCODE_MANNER_MODE	Mute/vibration mode switch
206	KEYCODE_3D_MODE	2D/3D switch
207	KEYCODE_CONTACTS	Address book
208	KEYCODE_CALENDAR	Calendar
209	KEYCODE_MUSIC	Music
210	KEYCODE_CALCULATOR	Calculator
211	KEYCODE_ZENKAKU_HANKAKU	Japanese full-width/half-width key
212	KEYCODE_EISU	Japanese alphanumeric key
213	KEYCODE_MUHENKAN	Japanese non-conversion key
214	KEYCODE_HENKAN	Japanese switch key
215	KEYCODE_KATAKANA_HIRAGANA	Japanese Katakana/Hiragana key
216	KEYCODE_YEN	JPY
217	KEYCODE_RO	Japanese XX
218	KEYCODE_KANA	Japanese XX
219	KEYCODE_ASSIST	Assist
220	KEYCODE_BRIGHTNESS_DOWN	Brightness down
221	KEYCODE_BRIGHTNESS_UP	Brightness up
222	KEYCODE_MEDIA_AUDIO_TRACK	Audio track
223	KEYCODE_SLEEP	Sleep
224	KEYCODE_WAKEUP	Weak up
225	KEYCODE_PAIRING	Peripheral device pairing
226	KEYCODE_MEDIA_TOP_MENU	Top menu
227	KEYCODE_11	11
228	KEYCODE_12	12
229	KEYCODE_LAST_CHANNEL	Previous channel
230	KEYCODE_TV_DATA_SERVICE	Data service
231	KEYCODE_VOICE_ASSIST	Voice assist
232	KEYCODE_TV_RADIO_SERVICE	Video and audio switching
233	KEYCODE_TV_TELETEXT	Teletext switch
234	KEYCODE_TV_NUMBER_ENTRY	Switch numeric key
235	KEYCODE_TV_TERRESTRIAL_ANALOG	Analog terrestrial television

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
236	KEYCODE_TV_TERRESTRIAL_DIGITAL	Digital terrestrial television
237	KEYCODE_TV_SATELLITE	Digital satellite television
238	KEYCODE_TV_SATELLITE_BS	Japan BS Digital Satellite TV
239	KEYCODE_TV_SATELLITE_CS	Japan CS Digital Satellite TV
240	KEYCODE_TV_SATELLITE_SERVICE	BS/CS switch
241	KEYCODE_TV_NETWORK	Two-way/broadcast switch
242	KEYCODE_TV_ANTENNA_CABLE	Antenna/wire switch
243	KEYCODE_TV_INPUT_HDMI_1	HDMI input 1
244	KEYCODE_TV_INPUT_HDMI_2	HDMI input 2
245	KEYCODE_TV_INPUT_HDMI_3	HDMI input 3
246	KEYCODE_TV_INPUT_HDMI_4	HDMI input 4
247	KEYCODE_TV_INPUT_COMPOSITE_1	CVBS input 1
248	KEYCODE_TV_INPUT_COMPOSITE_2	CVBS input 2
249	KEYCODE_TV_INPUT_COMPONENT_1	YpbPr input 1
250	KEYCODE_TV_INPUT_COMPONENT_2	YpbPr input 2
251	KEYCODE_TV_INPUT_VGA_1	VGA input
252	KEYCODE_TV_AUDIO_DESCRIPTION	Audio Descriptor Switch
253	KEYCODE_TV_AUDIO_DESCRIPTION_MIX_UP	Audio descriptor Volume up
254	KEYCODE_TV_AUDIO_DESCRIPTION_MIX_DOWN	Audio descriptor Volume down
255	KEYCODE_TV_ZOOM_MODE	Zoom in mode
256	KEYCODE_TV_CONTENTS_MENU	Content menu
257	KEYCODE_TV_MEDIA_CONTEXT_MENU	Context menu
258	KEYCODE_TV_TIMER_PROGRAMMING	Time adjustment
259	KEYCODE_HELP	Help
260	KEYCODE_NAVIGATE_PREVIOUS	Navigation Previous
261	KEYCODE_NAVIGATE_NEXT	Navigation Next
262	KEYCODE_NAVIGATE_IN	Navigation In
263	KEYCODE_NAVIGATE_OUT	Navigation Out
264	KEYCODE_STEM_PRIMARY	Wear Power/Restart
265	KEYCODE_STEM_1	Wear Universal key 1
266	KEYCODE_STEM_2	Wear Universal key 2
267	KEYCODE_STEM_3	Wear Universal key 3
268	KEYCODE_DPAD_UP_LEFT	Navigation Up left
269	KEYCODE_DPAD_DOWN_LEFT	Navigation Down left
270	KEYCODE_DPAD_UP_RIGHT	Navigation Up right
271	KEYCODE_DPAD_DOWN_RIGHT	Navigation Down right
272	KEYCODE_MEDIA_SKIP_FORWARD	Multimedia Fast forward

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
273	KEYCODE_MEDIA_SKIP_BACKWARD	Multimedia Fast backward
274	KEYCODE_MEDIA_STEP_FORWARD	Multimedia Step forward
275	KEYCODE_MEDIA_STEP_BACKWARD	Multimedia Step backward
276	KEYCODE_SOFT_SLEEP	Soft sleep
277	KEYCODE_CUT	Cut
278	KEYCODE_COPY	Copy
279	KEYCODE_PASTE	Paste
280	KEYCODE_SYSTEM_NAVIGATION_UP	System navigation Up
281	KEYCODE_SYSTEM_NAVIGATION_DOWN	System navigation Down
282	KEYCODE_SYSTEM_NAVIGATION_LEFT	System navigation Left
283	KEYCODE_SYSTEM_NAVIGATION_RIGHT	System navigation Right
1000	KEYCODE_VK_CANCEL	Cancel key.
1001	KEYCODE_VK_PRINT	Print key.
1002	KEYCODE_VK_EXECUTE	Execution key.
1003	KEYCODE_VK_SEPARATOR	Separator symbol key.
1004	KEYCODE_VK_AMPERSAND	"&"key
1005	KEYCODE_VK_QUOTEDBL	"""key.
1006	KEYCODE_VK_LESS	"<" key.
1007	KEYCODE_VK_GREATER	">" key.
1008	KEYCODE_VK_BRACELEFT	"{" key.
1009	KEYCODE_VK_BRACERIGHT	"}" key.
1010	KEYCODE_PICTUREMODE	Picture mode
1011	KEYCODE_SOURCE	Information source
1012	KEYCODE_TVSETUP	TV Setup
1013	KEYCODE_RECALL	
1014	KEYCODE_HEADSET_IN	Headphone In
1015	KEYCODE_HEADSET_OUT	Headphone Out
1016	KEYCODE_MIC_ON	Mic on
1017	KEYCODE_ICLOUD	
1018	KEYCODE_VOIP	VOIP Quick Launch Shorcut
1019	KEYCODE_RESOLUTION_RATIO	Resolution
1020	KEYCODE_AUDIO	Audio mode switch (stereo, left/right channel)
1021	KEYCODE_NPVR	Quasi-video recording
1022	KEYCODE_PQ	
1023	KEYCODE_QUIT	Quit
1024	KEYCODE_SERVICE	Service
1025	KEYCODE_CHANNEL_ALTERNATE	Channel switching
1026	KEYCODE_FAST_REVERSE	

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
1027	KEYCODE_SD	SD
1028	KEYCODE_HD	HD
1029	KEYCODE_SOUND_EFFECT	Sound effect
1030	KEYCODE_SMART	Smart
1031	KEYCODE_REFRESH	Refresh button, real-time web page refresh
1032	KEYCODE_VK_COLON	":"key.
1033	KEYCODE_VK_CIRCUMFLEX	"^"key.
1034	KEYCODE_VK_DOLLAR	"$"key.
1035	KEYCODE_VK_EURO_SIGN	Euro symbol key.
1036	KEYCODE_VK_EXCLAMATION_MARK	"!" key.
1037	KEYCODE_VK_INVERTED_EXCLAMATION_MARK	Inverted "!"key.
1038	KEYCODE_VK_UNDERSCORE	"_"key.
1039	KEYCODE_RCK_SELECT	Select key.
1040	KEYCODE_RCK_FAVORITE	"Favorite"key.
1041	KEYCODE_RCK_LANGUAGE	Language selection key.
1042	KEYCODE_RCK_SOFT_KEYBOARD	Soft keyboard.
1043	KEYCODE_RCK_RESUME	Resume key.
1044	KEYCODE_RCK_REVIEW	Review key.
1045	KEYCODE_RCK_REWIND	Rewind key.
1046	KEYCODE_RCK_GOTO	Goto key.
1047	KEYCODE_RCK_TITLE	"Title" key.
1048	KEYCODE_RCK_POSITION	Position key.
1049	KEYCODE_RCK_ANGLE	"Angle" select key.
1050	KEYCODE_RCK_SLOW	Slow key.
1051	KEYCODE_RCK_PROGRAM_PARADE	Program parade key.
1052	KEYCODE_RCK_RECOMMEND	Recommend key.
1053	KEYCODE_RCK_DESCRIPTION	"Description" key.
1054	KEYCODE_RCK_HISTORY_FORWORD	History forword key.
1055	KEYCODE_RCK_HISTORY_BACKWORD	History backword key.
1056	KEYCODE_RCK_RADIO	Radio key.
1057	KEYCODE_RCK_VOD	Video on Demand
1058	KEYCODE_RCK_NVOD	Near Video On Demand
1059	KEYCODE_RCK_MOVIE	Movie key.
1060	KEYCODE_RCK_ALBUM	Album key.
1061	KEYCODE_RCK_WEB	WEB key.
1062	KEYCODE_RCK_STOCK	Stock key.
1063	KEYCODE_RCK_MESSAGING	Messaging key.

Table D.2 – Key-value mapping table of button message code

Key-value	Message name	Message description
1064	KEYCODE_RCK_READER	Reader key.
1065	KEYCODE_RCK_GAME	Game key.
1066	KEYCODE_RCK_MOSAIC	Mosaic key.
1067	KEYCODE_RCK_LIST	List key.
1068	KEYCODE_RCK_REFRESH	Refresh key.
1069	KEYCODE_RCK_BUSINESS	Business key.
1070	KEYCODE_RCK_BUY	Buy key.
1071	KEYCODE_RCK_FORETELL	Foretell key.
1072	KEYCODE_RCK_STATUS	Status key.
1073	KEYCODE_RCK_SECURITIES	Securities key.
1074	KEYCODE_RCK_CLASS	Class key.
1075	KEYCODE_RCK_DISPLAY	"Display" key.

D.2.2 Interface org.ngb.interact.NgbKeyListener

Prototype: public interface org.ngb.interact.NgbKeyListener

Description: Key event listener.

D.2.2.1 Method

D.2.2.1.1 notifyKeyEvent

Prototype: public boolean notifyKeyEvent(org.ngb.interact.KeyEvent event)

Description: Key event notification.

Parameter: An event – An org.ngb.interact. KeyEvent event object indicating a key event.

Return: Boolean type, a key event processing result, true indicating the processing is successful, false indicating unprocessed or processing failed.

D.2.3 Interface org.ngb.interact.NgbMouseListener

Prototype: public interface org.ngb.interact.NgbMouseListener

Description: Mouse event listener.

D.2.3.1 Method

D.2.3.1.1 notifyMouseEvent

Prototype: public boolean notifyMouseEvent(org.ngb.interact.MouseEvent event)

Description: Mouse event notification.

Parameter: event – An org.ngb.interact.MouseEvent event object, which identifies a mouse event.

Return: Boolean type, indicating the mouse event processing result, true indicating the processing is successful, false indicating unprocessed or the processing failed.

D.2.4 Interface org.ngb.interact.NgbVoiceListener

Prototype: public interface org.ngb.interact.NgbVoiceListener

Description: ngb voice event listener.

D.2.4.1 Constant field – voice error code

D.2.4.1.1 ERROR_NETWORK_TIMEOUT

Prototype: public static final int ERROR_NETWORK_TIMEOUT = 1

Description: Error code-network timeout.

D.2.4.1.2 ERROR_NETWORK

Prototype: public static final int ERROR_NETWORK = 2

Description: Error code-other network related errors.

D.2.4.1.3 ERROR_AUDIO

Prototype: public static final int ERROR_AUDIO = 3

Description: Error code-audio recording error.

D.2.4.1.4 ERROR_SERVER

Prototype: public static final int ERROR_SERVER = 4

Description: Error code-the server returns an error status.

D.2.4.1.5 ERROR_CLIENT

Prototype: public static final int ERROR_CLIENT = 5

Description: Error code-An error occurred on the terminal side.

D.2.4.1.6 ERROR_SPEECH_TIMEOUT

Prototype: public static final int ERROR_SPEECH_TIMEOUT = 6

Description: Error code-no voice input.

D.2.4.1.7 ERROR_NO_MATCH

Prototype: public static final int ERROR_NO_MATCH = 7

Description: Error code-no matching voice recognition result was found.

D.2.4.1.8 ERROR_RECOGNIZER_BUSY

Prototype: public static final int ERROR_RECOGNIZER_BUSY = 8

Description: Error code-The voice recognition service is busy.

D.2.4.1.9 ERROR_INSUFFICIENT_PERMISSIONS

Prototype: public static final int ERROR_INSUFFICIENT_PERMISSIONS = 9

Description: Error code-insufficient permissions.

D.2.4.2 Method

D.2.4.2.1 onVoiceStart

Prototype: public void onVoiceStart()

Description: Voice input starts.

Parameter: None.

Return: None.

D.2.4.2.2 onVoiceEnd

Prototype: public void onVoiceStart()

Description: End of voice input.

Parameter: None.

Return: None.

D.2.4.2.3 onVoiceResult

Prototype: public void onVoiceResult(java.lang.String msg)

Description: The result of voice recognition.

Parameter: String type, indicating the result of this voice recognition, the result format is json type, including the specific original voice and semantic instruction.

Return: None.

D.2.4.2.4 onError

Prototype: public void onError(int code)

Description: Errors in the voice process.

Parameter: Integer, the value of the error code in the voice process, see the constant field definition of "Voice Error Code" for details.

Return: None.

D.2.5 Class org.ngb.interact.FrontPanel

Prototype: public class org.ngb.interact.FrontPanel

Description: Front panel information display output control, including LED indicator and LED digital tube display control.

D.2.5.1 Constant field – alignment

D.2.5.1.1 ALIGN_CENTER

Prototype: public static final int ALIGN_CENTER = 0

Description: Character alignment on the front panel-horizontally centered.

D.2.5.1.2 ALIGN_LEFT

Prototype: public static final int ALIGN_LEFT = 1

Description: Character alignment on the front panel-horizontally to the left.

D.2.5.1.3 ALIGN_RIGHT

Prototype: public static final int ALIGN_RIGHT = 2

Description: Character alignment on the front panel-horizontally to the right.

D.2.5.2 Constant field – indication status

D.2.5.2.1 STATUS_OFF

Prototype: public static final int STATUS_OFF = 0

Description: Indicating status-off.

D.2.5.2.2 STATUS_ON

Prototype: public static final int STATUS_ON = 1

Description: Indicating status-on.

D.2.5.2.3 STATUS_UNKNOWN

Prototype: public static final int STATUS_UNKNOWN = 2

Description: Indicating status-unknown.

D.2.5.3 Constant field – indication type

D.2.5.3.1 TYPE_MAIL

Prototype: public static final int TYPE_MAIL = 0

Description: Indication type-mail.

D.2.5.3.2 TYPE_SIGNAL

Prototype: public static final int TYPE_SIGNAL = 1

Description: Indication type-signal.

D.2.5.3.3 TYPE_POWER

Prototype: public static final int TYPE_POWER = 2

Description: Indication type-power.

D.2.5.3.4 TYPE_RADIO

Prototype: public static final int TYPE_RADIO = 3

Description: Indication type-broadcast.

D.2.5.4 Method

D.2.5.4.1 getInstance

Prototype: public static org.ngb.interact.FrontPanel getInstance()

Description: Getting the only instance of the front panel class implemented by the system.

Parameter: None.

Return: An org.ngb.interact.FrontPanel object, indicating the front panel class singleton implemented by the system.

D.2.5.4.2 clear

Prototype: public boolean clear()

Description: Clear the information displayed on the front panel, including the string information displayed on the front panel, time and date information, etc.

Parameter: None.

Return: boolean type, indicating the result of the clear, true value indicating that the clear is successful, and false value indicating that the clear fails.

D.2.5.4.3 displayDate

Prototype: public boolean displayDate(java.util.Date date)

Description: Display the current time and date information.

Parameter: date – java.util.Date type, indicating the date and time.

Return: boolean type, indicating the display result, true value indicating that the display is successful, and false value indicating that the display fails. If the terminal does not support time and date display, it can do not respond to the call of this method and returns false.

D.2.5.4.4 displayText

Prototype: public boolean displayText(java.lang.String str)

Description: The display string is displayed in horizontal center alignment by default.

Parameter: str – A java.lang.String object, indicating the string to be displayed.

Return: Boolean type, true value indicating the display is successful, and false indicating the display fails.

D.2.5.4.5 displayText

Prototype: public boolean displayText(java.lang.String str, int align)

Description: Display the string with the specified alignment.

Parameter: str – A java.lang.String object, indicating the string to be displayed;

align – Int type, indicating the horizontal alignment. For the value, see the "alignment" constant field definition of the org.ngb.interact.FrontPanel class.

Return: boolean type, true value indicating the display is successful, and false indicating the display fails.

D.2.5.4.6 getStatus

Prototype: public int getStatus(int type)

Description: Getting the indication status according to the specified type.

Parameter: type – Int type, indicating the indication type. For the value, see the "indication type" constant field definition of the org.ngb.interact.FrontPanel class.

Return: Int type, indicating the indication status.

– If the type parameter specifies a valid indication type, its actual status is returned (that is, the value STATUS_ON or STATUS_OFF);
– If the type parameter specifies an invalid indication type, STATUS_UNKNOWN is returned.

D.2.5.4.7 setStatus

Prototype: public boolean setStatus(int type, int value)

Description: Setting the indication status.

Parameter: type-int type, indicating the indication type, for the value, please refer to the "indication type" constant field definition of the org.ngb.interact.FrontPanel class;

value – Int type, indicating the indication status. For the value, please refer to the "indication status" constant field definition of the org.ngb.interact.FrontPanel class.

Return: boolean type, true value indicating that the setting is successful, and false value indicating that the setting fails.

D.2.5.4.8 getMaxChars

Prototype: public int getMaxChars()

Description: Getting the number of display characters supported by the front panel.

Parameter: None.

Return: Int type, indicating the number of display characters supported by the front panel.

D.2.6 Class org.ngb.interact.NgbInputManager

Prototype: public class org.ngb.interact.NgbInputManager

Description: Input/output control manager, is used to listen and receive events such as keystrokes, mouse, and inject control.

D.2.6.1 Method

D.2.6.1.1 getInstance

Prototype: public static org.ngb.interact.NgbInputManager getInstance()

Description: Getting the only instance of the input/output control management class implemented by the system.

Parameter: None.

Return: An org.ngb.interact.NgbInputManager object, indicating the input/output control management singleton implemented by the system.

D.2.6.1.2 addKeyEventListener

Prototype: public void addKeyEventListener(org.ngb.interact.NgbKeyListener listener)

Description: Register the specified key event listener to the system.

Parameter: listener – An org.ngb.interact.NgbKeyListener object, the key event listener, used for notification of key events.

Return: None.

D.2.6.1.3 removeKeyEventListener

Prototype: public void removeKeyEventListener(org.ngb.interact.NgbKeyListener listener)

Description: Remove the specified key event listener from the system.

Parameter: listener – An org.ngb.interact.NgbKeyListener object, the key event listener, used for notification of key events.

Return: None.

D.2.6.1.4 injectKeyEvent

Prototype: public boolean injectKeyEvent(org.ngb.interact.KeyEvent event)

Description: Forcedly injecting a key press event into the system.

Parameter: event – An org.ngb.interact.KeyEvent object, key event.

Return: boolean type, indicating the result of injecting key events, true indicating the injection is successful, false indicating the injection is failed.

D.2.6.1.5 addMouseEventListener

Prototype: public void addMouseEventListener(org.ngb.interact.NgbMouseListener listener)

Description: Register the specified mouse event listener into the system.

Parameter: listener – An org.ngb.interact.NgbMouseListener object, mouse event listener, used for notification of mouse events.

Return: None.

D.2.6.1.6 removeMouseEventListener

Prototype: public void removeMouseEventListener(org.ngb.interact.NgbMouseListener listener)

Description: Remove the specified mouse event listener from the system.

Parameter: listener-org.ngb.interact.NgbMouseListener object, mouse event listener, used for notification of mouse events.

Return: None.

D.2.6.1.7 injectMouseEvent

Prototype: public boolean injectMouseEvent(org.ngb.interact.MouseEvent event)

Description: Forcedly injecting a mouse event into the system.

Parameter: event – An org.ngb.interact.MouseEvent object, mouse event.

Return: Boolean type, injection mouse event result, true indicating the injection is successful, false indicating the injection is failed.

D.2.7 Class org.ngb.interact.NgbVoiceManager

Prototype: public class org.ngb.interact.NgbVoiceManager

Description: Voice-related control manager, used for control and realization of related functions such as voice recognition and voice broadcast.

D.2.7.1 Method

D.2.7.1.1 getInstance

Prototype: public static org.ngb.interact.NgbVoiceManager getInstance(org.tvos.content.Context context,org.tvos.content.ComponentName cn)

Description: Getting the only instance object of the voice-related control manager, which is used to control and implement related functions such as voice recognition and voice broadcast.

Parameter: context – An org.tvos.content.Context object, indicating the context of the current voice;

cn – An org.tvos.content.ComponentName object, indicating the voice service engine component to be used.

Return: An org.ngb.interact.NgbVoiceManager object, indicating a singleton of the voice-related control manager class.

D.2.7.1.2 startListening

Prototype: public void startListening(org.tvos.content.Intent intent)

Description: Start voice listening.

Parameter: intent – Intent object, related parameters of voice recognition.

Return: None.

D.2.7.1.3 stopListening

Prototype: public void stopListening()

Description: Stop voice listening.

Parameter: None.

Return: None.

D.2.7.1.4 cancel

Prototype: public void cancel()

Description: Cancel this voice recognition.

Parameter: None.

Return: None.

D.2.7.1.5 setVoiceListener

Prototype: public void setVoiceListener(org.ngb.interact.NgbVoiceListener listener)

Description: Setting up a voice listener.

Parameter: listener – An org.ngb.interact.NgbVoiceListener object, the voice listener.

Return: None.

D.2.7.1.6 release

Prototype: public void release()

Description: Release the voice manager and its occupied resources.

Parameter: None.

Return: None.

D.2.8 Event org.ngb.interact.NgbInputEvent

Prototype: public abstract class org.ngb.interact.NgbInputEvent

Description: Extended input event class, is the base class of input events such as keystrokes and mouses, and includes interfaces such as input event generation events and generation source types.

D.2.8.1 Method

D.2.8.1.1 getEventTime

Prototype: public abstract long getEventTime()

Description: Getting the time value of the input event.

Parameter: None.

Return: long type, indicating the time value generated by the input event. Based on the start-up time of the set-top box, it returns a value of milliseconds counted from the start-up time of the set-top box.

D.2.8.1.2 getSource

Prototype: public abstract int getSource()

Description: Getting the generation source of the input event.

Parameter: None.

Return: Int type, indicating the source of the input event. The possible values are INPUT_UNKNOWN, INPUT_FRONTPANEL, INPUT_KEYBOARD, INPUT_MOUSE, INPUT_REMOTECTRL, INPUT_USER. For details, please refer to the "User Input Message Source" constant field definition of the UserInput interface.

D.2.9 Event org.ngb.interact.KeyEvent

Prototype: public class org.ngb.interact.KeyEvent extends org.ngb.interact.NgbInputEvent

Description: Key event class, inherits the org.ngb.interact.NgbInputEvent class.

D.2.9.1 Method

D.2.9.1.1 getAction

Prototype: public int getAction()

Description: Getting the status of the key event.

Parameter: None.

Return: Int type, indicating the state of the key event, which can be KEY_TYPED, KEY_PRESSED, or KEY_RELEASED. For details, see the the constant field definition of "Keyboard Key Status" of the org.ngb.interact.UserInput interface.

D.2.9.1.2 getCode

Prototype: public int getCode()

Description: Getting the key value code of the key event.

Parameter: None.

Return: Int type, indicating the key value code of the key event. For the value, see the key message code constant field definition of the org.ngb.interact.UserInput interface.

D.2.10 Event org.ngb.interact.MouseEvent

Prototype: public class org.ngb.interact.MouseEvent extends org.ngb.interact.NgbInputEvent

Description: Mouse event class, inherits the org.ngb.interact.NgbInputEvent class.

D.2.10.1 Method

D.2.10.1.1 getAction

Prototype: public int getAction()

Description: Getting the status of the mouse event.

Parameter: None.

Return: Int type, indicating the state of the mouse event. The value can be MOUSE_PRESSED, MOUSE_RELEASED or MOUSE_MOVED. For details, see the constant field definition of "Mouse Button Status" of the UserInput interface.

D.2.10.1.2 getCode

Prototype: public int getCode()

Description: Getting the key value code of the mouse event.

Parameter: None.

Return: Int type, indicating the key value code of the mouse event. For the value, see the constant field definition of the "mouse key message code" of the org.ngb.interact.UserInput interface.

D.2.10.1.3 getButton

Prototype: public int getButton()

Description: When the mouse is pressed or released, the mouse button that sent the message is obtained.

Parameter: None.

Return: Int type, indicating the mouse button that sent the message.

– When the message subtype is MOUSE_RELEASED or MOUSE_PRESSED, the corresponding mouse button code (MOUSE_LBUTTON/MOUSE_RBUTTON/MOUSE_MBUTTON) will be returned;
– When the message subtype is MOUSE_MOVED, the MOUSE_NOBUTTON will be returned.

D.2.10.1.4 getX

Prototype: public int getX()

Description: Getting the position of the mouse focus on the X axis.

Parameter: None.

Return: Int type, indicating the position of the mouse focus on the X axis, in pixel.

D.2.10.1.5 getY

Prototype: public int getY()

Description: Getting the position of the mouse focus on the Y axis.

Parameter: None.

Return: Int type, indicating the position of the mouse focus on the Y axis, in pixel.

Annex E

JAVA-AV setting unit

(This annex forms an integral part of this Recommendation.)

E.1 Overview

This annex defines the JAVA interface related to AV settings.

E.2 AV setting module

The AV setting module provides classes and methods related to the setting of audio and video output parameters.

The summary of the AV setting module is shown in Table E.1.

Table E.1 – Summary of AV setting module

Class	
AudioSetting	Various parameter settings of the audio input/output unit.
VideoSetting	Various parameter settings of the video input/output unit.

E.2.1 Class org.ngb.util.setting.AudioSetting

Prototype: public class org.ngb.util.setting.AudioSetting

Description: Various parameter settings of the audio input/output unit. The system should verify the permissions of the application program, and only privileged applications can call the methods provided by this class.

– Audio output parameter settings include volume, etc.

– The audio input parameter settings are reserved for future expansion.

Notes on the output volume setting:

– The actual volume of a broadcast program = global volume + the increase in value of the broadcast program relative to the global volume.

E.2.1.1 Constant field – channel type

E.2.1.1.1 CHANNEL_STEREO

Prototype: public static final int CHANNEL_STEREO = 0

Description: Channel type – Stereo.

E.2.1.1.2 CHANNEL_LEFT

Prototype: public static final int CHANNEL_LEFT = 1

Description: Channel type – Left channel.

E.2.1.1.3 CHANNEL_RIGHT

Prototype: public static final int CHANNEL_RIGHT = 2

Description: Channel type – Right channel.

E.2.1.1.4 CHANNEL_MIXED_MONO

Prototype: public static final int CHANNEL_MIXED_MONO = 3

Description: Channel type – Mixed sound.

E.2.1.2 Method

E.2.1.2.1 getOutputInterfaceList

Prototype: public static java.lang.String[] getOutputInterfaceList()

Description: Getting a list of all available audio output ports of the receiving terminal.

Parameter: None.

Return: An array of java.lang.String objects, indicating the names of all available audio output ports of the receiving terminal, such as "RCA", "S/PDIF", "HDMI", etc. If there is no audio output port, the length of the returned array is 0.

E.2.1.2.2 getOutputInterfaceStatus

Prototype: public static boolean getOutputInterfaceStatus(java.lang.String port)

Description: Getting the enabling status of the audio output port.

Parameter: port – A java.lang.String object, indicating the name of the audio output port.

NOTE – The audio output port name is obtained by the getOutputInterfaceList() method.

Return: boolean type, indicating the enabling status of the audio output port, true value indicating that the audio output port is allowed to output, and false value indicating that the audio output port is forbidden to output.

E.2.1.2.3 disableOutputInterface

Prototype: public static boolean disableOutputInterface(java.lang.String port)

Description: Disable output of the audio output port.

Parameter: port – A java.lang.String object, indicating the name of the audio output interface.

NOTE – The audio output port name is obtained by the getOutputInterfaceList() method.

Return: Boolean type, true value indicating success, false indicating failure.

E.2.1.2.4 enableOutputInterface

Prototype: public static boolean enableOutputInterface(java.lang.String port)

Description: Allow output of the audio output port.

Parameter: port – A java.lang.String object, indicating the name of the audio output port.

NOTE – The audio output port name is obtained by the getOutputInterfaceList() method.

Return: Boolean type, true value indicating success, false indicating failure.

E.2.1.2.5 getOutputVolume

Prototype: public static int getOutputVolume()

Description: Getting the global audio output volume.

Parameter: None.

Return: Int type, indicating the size of the output volume, the value range being 0-100, 0 indicating mute, and 100 indicating maximum volume.

E.2.1.2.6 setOutputVolume

Prototype: public static boolean setOutputVolume(int volume)

Description: Setting the global audio output volume.

NOTE – The actual output volume of a broadcast program = global output volume + the increase of the broadcast program relative to the global output volume.

Parameter: volume-int type, indicating the size of the output volume, the value range being 0-100, 0 indicating mute, and 100 indicating maximum volume.

Return: boolean type, indicating the setting result, true value indicating that the setting is successful, and false value indicating that the setting fails.

E.2.1.2.7 getOutputChannelMode

Prototype: public static int getOutputChannelMode()

Description: Getting the output channel type.

Parameter: None.

Return: Int type, indicating the output channel type. For the value, see the constant field definition of the "channel type" of the AudioSetting class.

E.2.1.2.8 setOutputChannelMode

Prototype: public static boolean setOutputChannelMode(int type)

Description: Setting the output channel.

Parameter: type – Int type, indicating the channel type to be set. For the value, please refer to the constant field definition of the "channel type" of the AudioSetting class.

Return: boolean type, indicates the setting result, true value indicating that the setting is successful, and false value indicating that the setting fails.

E.2.1.2.9 getOutputSPDIFMode

Prototype: public static int getOutputSPDIFMode()

Description: Getting the S/PDIF output interface data format (compressed or PCM format).

Parameter: None.

Return: Int type, indicating the data format of the S/PDIF output interface, and the value is:

– 0-indicating the PCM format;

– 1-indicating the compression format.

E.2.1.2.10 setOutputSPDIFMode

Prototype: public boolean static setOutputSPDIFMode(int mode)

Description: Setting the S/PDIF output interface data format (compressed or PCM format).

Parameter: mode – Int type, indicating the data format of the S/PDIF output interface, and the value is:

– 0-indicating the PCM format, that is, compressed audio is decoded by the receiving terminal;

– 1-indicating the compression format, that is, compressed audio is decoded by an external decoding device.

Return: boolean type, indicating the setting result, true value indicating that the setting is successful, and false value indicating that the setting fails.

E.2.1.2.11 isMute

Prototype: public static boolean isMute()

Description: Determine whether the audio output is muted.

Parameter: None.

Return: boolean type, true value indicating it is in the mute state, and false indicating it is in the sound state.

E.2.1.2.12 mute

Prototype: public static boolean mute()

Description: Mute.

Parameter: None.

Return: boolean type, indicating the setting result, true value indicating that the mute is successful, and false value indicating that the mute fails.

E.2.1.2.13 unMute

Prototype: public static boolean unMute()

Description: unmute.

Parameter: None.

Return: boolean type, indicating the result of the cancellation, true value indicating that the cancellation is successful, and false value indicating that the cancellation fails.

E.2.1.2.14 getOutputHDMIFMode

Prototype: public static int getOutputHDMIFMode()

Description: Getting the HDMI output interface data format.

Parameter: None.

Return: Int type, indicating the data format of the HDMI output interface.

0-indicating off;

1-Auto-negotiation;

2-LPCM, outputting the decoded signal;

3-RAW, outputting the original signal.

E.2.1.2.15 setOutputHDMIMode

Prototype: public static boolean setOutputHDMIMode(int mode)

Description: Setting the HDMI output interface data format (compressed or PCM format).

Parameter: mode – int type, indicating HDMI output interface data format, the value is:

0-indicating off;

1-Auto-negotiation;

2-LPCM, outputting the decoded signal;

3-RAW, outputting the original signal.

Return: boolean type, indicating the setting result, true value indicating that the setting is successful, and false value indicating that the setting fails.

E.2.2 Class org.ngb.util.setting.VideoSetting

Prototype: public class org.ngb.util.setting.VideoSetting

Description: Various parameter settings of the video input/output unit. The system should verify the permissions of the application, and only authorized applications can call the methods provided by this class.

– Video output parameter settings include standard (resolution, field frequency, amplitude-to-type ratio), brightness, contrast, transparency, etc.;
– Video input parameter settings are reserved for future expansion.

E.2.3 Constant field – match method

E.2.3.1.1 MATCH_METHOD_LETTER_BOX

Prototype: public static final int MATCH_METHOD_LETTER_BOX = 1

Description: Video window match method-letter box (letter_box).

E.2.3.1.2 MATCH_METHOD_PAN_SCAN

Prototype: public static final int MATCH_METHOD_PAN_SCAN = 2

Description: Video window match method-pan scan (pan_scan).

E.2.3.1.3 MATCH_METHOD_COMBINED

Prototype: public static final int MATCH_METHOD_COMBINED = 3

Description: Video window match method-combined method (combined).

E.2.3.1.4 MATCH_METHOD_IGNORE

Prototype: public static final int MATCH_METHOD_IGNORE = 4

Description: Video window match method-ignore method (ignore).

E.2.3.2 Constant field – output channel

E.2.3.2.1 VOUT_SD

Prototype: public static final int VOUT_SD = 1

Description: SD output channel.

E.2.3.2.2 VOUT_HD

Prototype: public static final int VOUT_HD = 2

Description: HD output channel.

E.2.3.3 Constant field – output standard

E.2.3.3.1 VOUT_STANDARD_UNKNOWN

Prototype: public static final int VOUT_STANDARD_UNKNOWN = 0

Description: Video output standard: unknown.

E.2.3.3.2 VOUT_STANDARD_NTSC_J

Prototype: public static final int VOUT_STANDARD_NTSC_J = 101

Description: Video output standard: SD-NTSC-J/3.5795MHz color subcarrier.

E.2.3.3.3 VOUT_STANDARD_NTSC_M

Prototype: public static final int VOUT_STANDARD_NTSC_M = 102

Description: Video output standard: SD-NTSC-M/3.5795MHz color subcarrier.

E.2.3.3.4 VOUT_STANDARD_NTSC_443

Prototype: public static final int VOUT_STANDARD_NTSC_443 = 103

Description: Video output standard: SD-NTSC-443/4.4336MHz color subcarrier.

E.2.3.3.5 VOUT_STANDARD_PAL_B

Prototype: public static final int VOUT_STANDARD_PAL_B = 211

Description: Video output standard: SD-PAL-B (Australia).

E.2.3.3.6 VOUT_STANDARD_PAL_B1

Prototype: public static final int VOUT_STANDARD_PAL_B1 = 212

Description: Video output standard: SD-PAL-B1 (Hungary).

E.2.3.3.7 VOUT_STANDARD_PAL_D

Prototype: public static final int VOUT_STANDARD_PAL_D = 213

Description: Video output standard: SD-PAL-D (Mainland China).

E.2.3.3.8 VOUT_STANDARD_PAL_D1

Prototype: public static final int VOUT_STANDARD_PAL_D1 = 214

Description: Video output standard: SD-PAL-D1 (Poland).

E.2.3.3.9 VOUT_STANDARD_PAL_G

Prototype: public static final int VOUT_STANDARD_PAL_G = 215

Description: Video output standard: SD-PAL-G (Europe).

E.2.3.3.10 VOUT_STANDARD_PAL_H

Prototype: public static final int VOUT_STANDARD_PAL_H = 216

Description: Video output standard: SD-PAL-H (Europe).

E.2.3.3.11 VOUT_STANDARD_PAL_I

Prototype: public static final int VOUT_STANDARD_PAL_I = 217

Description: Video output standard: SD-PAL-I (UK, Hong Kong, Macau).

E.2.3.3.12 VOUT_STANDARD_PAL_K

Prototype: public static final int VOUT_STANDARD_PAL_K = 218

Description: Video output standard: SD-PAL-K (Europe).

E.2.3.3.13 VOUT_STANDARD_PAL_M

Prototype: public static final int VOUT_STANDARD_PAL_M = 220

Description: Video output standard: SD-PAL-M (Brazil).

E.2.3.3.14 VOUT_STANDARD_PAL_N

Prototype: public static final int VOUT_STANDARD_PAL_N = 221

Description: Video output standard: SD-PAL-N (Jamaica, Uruguay).

E.2.3.3.15 VOUT_STANDARD_PAL_NC

Prototype: public static final int VOUT_STANDARD_PAL_NC = 222

Description: Video output standard: SD-PAL-NC (Argentina).

E.2.3.3.16 VOUT_STANDARD_SECAM_B

Prototype: public static final int VOUT_STANDARD_SECAM_B = 311

Description: Video output standard: SD-SECAM-B.

E.2.3.3.17 VOUT_STANDARD_SECAM_D

Prototype: public static final int VOUT_STANDARD_SECAM_D = 312

Description: Video output standard: SD-SECAM-D.

E.2.3.3.18 VOUT_STANDARD_SECAM_G

Prototype: public static final int VOUT_STANDARD_SECAM_G = 313

Description: Video output standard: SD-SECAM-G.

E.2.3.3.19 VOUT_STANDARD_SECAM_I

Prototype: public static final int VOUT_STANDARD_SECAM_I = 314

Description: Video output standard: SD-SECAM-I.

E.2.3.3.20 VOUT_STANDARD_SECAM_K

Prototype: public static final int VOUT_STANDARD_SECAM_K = 315

Description: Video output standard: SD-SECAM-K.

E.2.3.3.21 VOUT_STANDARD_SMPTE274_1080I_50

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080I_50 = 27400

Description: Video output standard: HD-SMPTE274/1920x1080I/50HZ/1125 lines.

E.2.3.3.22 VOUT_STANDARD_SMPTE274_1080I_59_94

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080I_59_94 = 27401

Description: Video output standard: HD-SMPTE274/1920x1080I/59.94HZ/1125 lines.

E.2.3.3.23 VOUT_STANDARD_SMPTE274_1080I_60

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080I_60 = 27402

Description: Video output standard: HD-SMPTE274/1920x1080I/60HZ/1125 lines.

E.2.3.3.24 VOUT_STANDARD_SMPTE274_1080P_23_98

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080P_23_98 = 27410

Description: Video output standard: HD-SMPTE274/1920x1080P/23.98HZ/1125 lines.

E.2.3.3.25 VOUT_STANDARD_SMPTE274_1080P_24

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080P_24 = 27411

Description: Video output standard: HD-SMPTE274/1920x1080P/24HZ/1125 lines.

E.2.3.3.26 VOUT_STANDARD_SMPTE274_1080P_25

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080P_25 = 27412

Description: Video output standard: HD-SMPTE274/1920x1080P/25HZ/1125 lines.

E.2.3.3.27 VOUT_STANDARD_SMPTE274_1080P_29_97

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080P_29_97 = 27413

Description: Video output standard: HD-SMPTE274/1920x1080P/29.97HZ/1125 lines.

E.2.3.3.28 VOUT_STANDARD_SMPTE274_1080P_30

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080P_30 = 27414

Description: Video output standard: HD-SMPTE274/1920x1080P/30HZ/1125 lines.

E.2.3.3.29 VOUT_STANDARD_SMPTE274_1080P_50

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080P_50 = 27415

Description: Video output standard: HD-SMPTE274/1920x1080P/50HZ/1125 lines.

E.2.3.3.30 VOUT_STANDARD_SMPTE274_1080P_59_94

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080P_59_94 = 27416

Description: Video output standard: HD-SMPTE274/1920x1080P/59.94HZ/1125 lines.

E.2.3.3.31 VOUT_STANDARD_SMPTE274_1080P_60

Prototype: public static final int VOUT_STANDARD_SMPTE274_1080P_60 = 27417

Description: Video output standard: HD-SMPTE274/1920x1080P/60HZ/1125 lines.

E.2.3.3.32 VOUT_STANDARD_SMPTE295_1080I_50

Prototype: public static final int VOUT_STANDARD_SMPTE295_1080I_50 = 29500

Description: Video output standard: HD-SMPTE295/1920x1080I/50HZ/1250 lines.

E.2.3.3.33 VOUT_STANDARD_SMPTE295_1080P_50

Prototype: public static final int VOUT_STANDARD_SMPTE295_1080P_50 = 29510

Description: Video output standard: HD-SMPTE295/1920x1080P/50HZ/1250 lines.

E.2.3.3.34 VOUT_STANDARD_SMPTE296_720P_23_98

Prototype: public static final int VOUT_STANDARD_SMPTE296_720P_23_98 = 29610

Description: Video output standard: HD-SMPTE296/1280x720P/23.98HZ/750 lines.

E.2.3.3.35 VOUT_STANDARD_SMPTE296_720P_24

Prototype: public static final int VOUT_STANDARD_SMPTE296_720P_24 = 29611

Description: Video output standard: HD-SMPTE296/1280x720P/24HZ/750 lines.

E.2.3.3.36 VOUT_STANDARD_SMPTE296_720P_25

Prototype: public static final int VOUT_STANDARD_SMPTE296_720P_25 = 29612

Description: Video output standard: HD-SMPTE296/1280x720P/25HZ/750 lines.

E.2.3.3.37 VOUT_STANDARD_SMPTE296_720P_29_97

Prototype: public static final int VOUT_STANDARD_SMPTE296_720P_29_97 = 29613

Description: Video output standard: HD-SMPTE296/1280x720P/29.97HZ/750 lines.

E.2.3.3.38 VOUT_STANDARD_SMPTE296_720P_30

Prototype: public static final int VOUT_STANDARD_SMPTE296_720P_30 = 29614

Description: Video output standard: HD-SMPTE296/1280x720P/30HZ/750 lines.

E.2.3.3.39 VOUT_STANDARD_SMPTE296_720P_50

Prototype: public static final int VOUT_STANDARD_SMPTE296_720P_50 = 29615

Description: Video output standard: HD-SMPTE296/1280x720P/50HZ/750 lines.

E.2.3.3.40 VOUT_STANDARD_SMPTE296_720P_59_94

Prototype: public static final int VOUT_STANDARD_SMPTE296_720P_59_94 = 29616

Description: Video output standard: HD-SMPTE296/1280x720P/59.94HZ/750 lines.

E.2.3.3.41 VOUT_STANDARD_SMPTE296_720P_60

Prototype: public static final int VOUT_STANDARD_SMPTE296_720P_60 = 29617

Description: Video output standard: HD-SMPTE296/1280x720P/60HZ/750 lines.

E.2.3.3.42 VOUT_STANDARD_2160P_24

Prototype: public static final int VOUT_STANDARD_2160P_24 = 29618

Description: Video output standard: 4K HD/3840x2160P/24.

E.2.3.3.43 VOUT_STANDARD_2160P_25

Prototype: public static final int VOUT_STANDARD_2160P_25 = 29619

Description: Video output standard: 4K HD/3840x2160P/25.

E.2.3.3.44 VOUT_STANDARD_2160P_30

Prototype: public static final int VOUT_STANDARD_2160P_30 = 29620

Description: Video output standard: 4K HD/3840x2160P/30.

E.2.3.3.45 VOUT_STANDARD_2160P_50

Prototype: public static final int VOUT_STANDARD_2160P_24 = 29621

Description: Video output standard: 4K HD/3840x2160P/50.

E.2.3.3.46 VOUT_STANDARD_2160P_60

Prototype: public static final int VOUT_STANDARD_2160P_60 = 29622

Description: Video output standard: 4K HD/3840x2160P/60.

E.2.3.3.47 VOUT_STANDARD_4096P_24

Prototype: public static final int VOUT_STANDARD_4096P_24 = 29623

Description: Video output standard: 4K HD/3840x2160P/24.

E.2.3.4 Method

E.2.3.4.1 getOutputInterfaceList

Prototype: public static java.lang.String[] getOutputInterfaceList()

Description: Getting a list of all available video output ports of a receiving terminal.

Parameter: None.

Return: A java.lang.String object array, indicating the name of all available video output interfaces of the receiving terminal, such as "CVBS", "YUV", "HDMI-0", "HDMI-1", "DVO", etc. If there is no video output port, the length of the returned array is 0.

E.2.3.4.2 getOutputInterfaceStatus

Prototype: public static boolean getOutputInterfaceStatus(java.lang.String port)

Description: Getting the enabling status of the video output port.

Parameter: port – A java.lang.String object, indicating the name of the video output port.

NOTE – The name of the video output port is obtained by the getOutputInterfaceList() method.

Return: boolean type, indicating the enabling status of the video output port, true value indicating that the video output interface is allowed to output, and false value indicating that the video output interface is forbidden to output.

E.2.3.4.3 disableOutputInterface

Prototype: public static boolean disableOutputInterface(java.lang.String port)

Description: Disable video output port output.

Parameter: port – A java.lang.String object, indicating the name of the video output port.

NOTE – The name of the video output port is obtained by the getOutputInterfaceList() method.

Return: boolean type, true value indicating that the prohibition is successful, and false value indicating that the prohibition fails.

E.2.3.4.4 enableOutputInterface

Prototype: public static void enableOutputInterface(java.lang.String port)

Description: Enable video output port to output.

Parameter: port – A java.lang.String object, indicating the name of the video output port.

NOTE – The name of the video output port is obtained by the getOutputInterfaceList() method.

Return: boolean type, true value indicating that success is allowed, and false value indicating that failure is allowed.

E.2.3.4.5 getOutputMatchMethod

Prototype: public static int getOutputMatchMethod()

Description: Getting a match method of the video and the display window. The specific effect needs to be determined according to the aspect ratio of the program being played, the match method, and whether the display device has an adaptive function.

Parameter: None.

Return: Int type, indicating the match method of the video output port. For the value, see the constant field definition of the "match method" of the VideoSetting class.

E.2.3.4.6 getOutputBrightness

Prototype: public static int getOutputBrightness()

Description: Getting the brightness of the video output.

Parameter: None.

Return: Int type, indicating the brightness of the video output. The value ranges from small to large, from 0 to 100, with 0 indicating the darkest and 100 indicating the brightest.

E.2.3.4.7 getOutputContrast

Prototype: public static int getOutputContrast()

Description: Getting the contrast of the video output.

Parameter: None.

Return: Int type, indicating the contrast of the video output, the value range being 0-100 from small to large.

E.2.3.4.8 getOutputSaturation

Prototype: public static int getOutputSaturation()

Description: Getting a saturation (chroma) of the video output.

Parameter: None.

Return: Int type, indicating the saturation (chroma) of the video output, the value range being from 0 to 100, from small to large.

E.2.3.4.9 getOutputStandard

Prototype: public static int getOutputStandard(int device)

Description: Getting a video output standard.

Parameter: device – Int type, indicating a video output unit, which can be VOUT_SD or VOUT_HD. For details, refer to the constant field definition of "output channel" of VideoSetting class.

Return: Int type, indicating the video output standard. For the value, see the constant field definition of "output format" of VideoSetting class.

E.2.3.4.10 getOutputStandards

Prototype: public static int[] getOutputStandards(int device)

Description: Getting all supported standards of video output.

Parameter: device – Int type, indicating a video output unit, the value of which can be VOUT_SD or VOUT_HD. For details, refer to the constant field definition of "output channel" of VideoSetting class.

Return: int type array, indicating all supported standards of video output. For the values, please refer to the constant field definition of "output standard" of org.ngb.util.setting.VideoSetting class.

E.2.3.4.11 getOutputTransparency

Prototype: public static int getOutputTransparency()

Description: Getting a transparency of the video output.

Parameter: None.

Return: int type, indicating transparency, the value range being 0-100, 0 indicating completely opaque, and 100 indicating completely transparent.

E.2.3.4.12 setOutputMatchMethod

Prototype: public static void setOutputMatchMethod(int matchMethod)

Description: Setting the adaptation method of the screen. The specific effect needs to be determined according to the aspect ratio of the program being played, the screen adaptation mode set by the user, and whether the display device has an adaptive function.

Parameter: matchMethod – int type, a match method of the video and the display window. For the value, please refer to the constant field definition of "match method" of org.ngb.util.setting.VideoSetting class.

Return: None.

E.2.3.4.13 setOutputBrightness

Prototype: public static void setOutputBrightness(int value)

Description: Setting a brightness of video output. This setting cannot be applied to a single video output unit and is effective for all video output units at the same time.

Parameter: value – Int type, indicating the brightness of the video output, the value range being 0-100, 0 indicating the darkest, and 100 indicating the brightest.

Return: None.

E.2.3.4.14 setOutputContrast

Prototype: public static void setOutContrast(int value)

Description: Contrast of the video output. This setting cannot be applied to a single video output unit and is effective for all video output units at the same time.

Parameter: value – Int type, indicating the contrast of the video output, and the value range being 0-100.

Return: None.

E.2.3.4.15 setOutputSaturation

Prototype: public static void setOutputSaturation(int value)

Description: Setting the saturation (chroma) of the video output. This setting cannot be applied to a single video output unit and is effective for all video output units at the same time.

Parameter: value – Int type, indicating a saturation (chroma), and the value range being 0-100.

Return: None.

E.2.3.4.16 setOutputStandard

Prototype: public static int setOutputStandard(int device, int standard)

throws unsupportedOperationException

Description: Setting the video output standard. The SD output unit and HD output unit need to be set separately.

Parameter: device – Int type, indicating video output unit, and the value being VOUT_SD or VOUT_HD. For details, please refer to constant field definition of "ouput channel" of org.ngb.util.setting.VideoSetting class;

standard – Int type, indicating a video output standard. A SD output unit can only select the SD standard, and An HD output unit can only select the HD standard. For the value, please refer to the constant field definition of "output standard" of org.ngb.util.setting.VideoSetting class.

Return: Int type, indicating a video output standard before setting. For the value, please refer to the constant field definition of "output standard" of org.ngb.util.setting.VideoSetting class.

Exception: unsupportedOperationException – If this operation is not supported, this exception is thrown.

E.2.3.4.17 setOutputTransparency

Prototype: public static void setOutputTransparency(int value)

Description: Setting the transparency of the video output. This method cannot be set for a single video output unit and is effective for all video output units.

Parameter: value – Int type, indicating transparency, the value range being 0-100, 0 indicating completely opaque, and 100 indicating completely transparent.

Return: None.

E.2.3.4.18 getOutputAspectRatio

Prototype: public static int getOutputAspectRatio(int device)

Description: Getting an aspect ratio of video output.

Parameter: device – Int type, indicating a video output unit, the value is VOUT_SD or VOUT_HD, see the constant field definition of "output channel" of org.ngb.util.setting.VideoSetting class for details;

Return: int type, 0 indicating 16:9; 1 indicating 4:3.

E.2.3.4.19 setOutputAspectRatio

Prototype: public static boolean setOutputAspectRatio(int device, int mode)

Description: Setting the aspect ratio of the video output.

Parameter: device – Int type, indicating the video output unit, the value is VOUT_SD or VOUT_HD, see the constant field definition of "output channel" of org.ngb.util.setting.VideoSetting class for details;

mode – int type, 0 indicating 16:9; 1 indicating 4:3.

Return: boolean type, indicating the execution result of the video output aspect ratio setting, true value indicating that the execution is successful, and false value indicating failure.

E.2.3.4.20 GetColorSpaceMode

Prototype: public static int GetColorSpaceMode()

Description: Getting a color space mode.

Parameter: None.

Return: int type, 0-RGB444, 1-YCBCR422, 2-YCBCR444, 3-YCBCR420.

E.2.3.4.21 GetDeepColorMode

Prototype: public static int GetDeepColorMode()

Description: Getting a color depth mode.

Parameter: None.

Return: int type, 0-COLOR_24BIT, 1-COLOR_30BIT, 2-COLOR_36BIT, 3-COLOR_DEEP_OFF.

E.2.3.4.22 SetColorSpaceAndDeepColor

Prototype: public static boolean SetColorSpaceAndDeepColor(int colorSpace, int deepColor)

Description: Setting the color space and color depth mode.

Parameter: colorSpace – int type, 0-RGB444, 1-YCBCR422, 2-YCBCR444, 3-YCBCR420

deepColor – int type, 0-COLOR_24BIT, 1-COLOR_30BIT, 2-COLOR_36BIT, 3-COLOR_DEEP_OFF

Return: boolean type, true indicating the execution is successful, and false value indicating failure.

E.2.3.4.23 GetHDRType

Prototype: public static int GetHDRType()

Description: Getting HDR mode.

Parameter: None.

Return: int type, 0-HDRTYPE_SDR, 1-HDRTYPE_DOLBY, 2-HDRTYPE_HDR10, 3-C HDRTYPE_AUTO.

E.2.3.4.24 SetHDRType

Prototype: public static boolean SetHDRType(int type)

Description: Setting HDR mode.

Parameter: type – int type, 0-HDRTYPE_SDR, 1-HDRTYPE_DOLBY, 2-HDRTYPE_HDR10, 3-C HDRTYPE_AUTO

Return: boolean type, true value indicating the setting is successful, and false indicating the setting fails.

E.2.3.4.25 GetStereoOutMode

Prototype: public static int GetStereoOutMode()

Description: Getting 3D output mode.

Parameter: None.

Return: int type, 0-3D_NONE, 1-3D_FRAME_PACKING, 2-3D_SIDE_BY_SIDE_HALF, 3-3D_TOP_AND_BOTTOM, 4-3D_FIELD_ALTERNATIVE, 5-3D_LINE_ALTERNATIVE, 6-3D_SIDE_BY_SIDE_FULL, 7-3D_L_DEPTH, 8-3D_L_DEPTH_GRAPHICS_GRAPHICS_DEPTH.

E.2.3.4.26 SetStereoOutMode

Prototype: public static boolean SetStereoOutMode(int mode,int fps)

Description: Setting 3D output mode.

Parameter: mode – int type, 0-3D_NONE, 1-3D_FRAME_PACKING, 2-3D_SIDE_BY_SIDE_HALF, 3-3D_TOP_AND_BOTTOM, 4-3D_FIELD_ALTERNATIVE, 5-3D_LINE_ALTERNATIVE, 6-3D_SIDE_BY_SIDE_FULL, 7-3D_L_DEPTH, 8-3D_L_DEPTH_GRAPHICS_GRAPHICS_DEPTH;

fps – int type, video frame rate, 23, 24, 25, 30, 50, 59, 60;

Return: boolean type, true value indicating the setting is successful, false indicating the setting fails.

E.2.3.4.27 GetRightEyeFirst

Prototype: public static int GetRightEyeFirst()

Description: Getting which eye of the 3D output signal is to come out first.

Parameter: None.

Return: int type, 0-LEFT_EYE_FIRST, 1-RIGHT_EYE_FIRST.

E.2.3.4.28 SetRightEyeFirst

Prototype: public static boolean SetRightEyeFirst(int Outpriority)

Description: Setting which eye of the 3D output signal is to come out first.

Parameter: Outpriority – int type, 0-LEFT_EYE_FIRST, 1-RIGHT_EYE_FIRST.

Return: boolean type, true value indicating the setting is successful, false indicating the setting fails.

E.2.3.4.29 GetStereoDepth

Prototype: public static int GetStereoDepth()

Description: Getting 3D picture depth adjustment information.

Parameter: None.

Return: int type, 0-10.

E.2.3.4.30 SetStereoDepth

Prototype: public static boolean SetStereoDepth(int depth)

Description: Setting 3D picture depth adjustment information.

Parameter: depth – int type, 0-10.

Return: boolean type, true value indicating the setting is successful, false indicating the setting fails.

E.2.3.4.31 getPictureMode

Prototype: public static int getPictureMode()

Description: Getting a picture mode.

Parameter: None.

Return: int type, 0 standard, 1 dynamic, 2 soft, 4 users, 5 gorgeous, 6 natural, 7 sports.

E.2.3.4.32 setPictureMode

Prototype: public static boolean setPictureMode(int mode)

Description: Setting a picture mode.

Parameter: mode – int type, 0 standard, 1 dynamic, 2 soft, 4 users, 5 gorgeous, 6 natural, 7 sports.

Return: boolean type, true value indicating that the setting is successful, and false value indicating that the setting fails.

E.2.3.4.33 getDisplayHue

Prototype: public static int getDisplayHue()

Description: Getting the picture mode.

Parameter: None.

Return: int type, the value range being 0-100, indicating the hue adjustment value.

E.2.3.4.34 setDisplayHue

Prototype: public static boolean setDisplayHue(int hue)

Description: Setting a picture mode.

Parameter: hue – int type, the value range being 0-100, indicating the hue adjustment value.

Return: boolean type, true value indicating the setting is successful, false indicating the setting fails.

E.2.3.4.35 SaveDisplayFmt

Prototype: public static boolean SaveDisplayFmt()

Description: Saving the video output format permanently.

Parameter: None.

Return: boolean type, true value indicating success, false indicating failure.

E.2.3.4.36 setOptimalFormatEnable

Prototype: public static boolean setOptimalFormatEnable(int enabled)

Description: Setting an auto-optimized video output format enabling status.

Parameter: enabled – int type, 0-indicating not enabled; 1-indicating enabled.

Return: boolean type, true value indicating success, false indicating failure.

E.2.3.4.37 getOptimalFormatEnable

Prototype: public static int getOptimalFormatEnable()

Description: Getting the auto-optimized video output format enabling status.

Parameter: None.

Return: int type, 0-indicating not enabled; 1-indicating enabled.

Annex F

JAVA-Media processing unit

(This annex forms an integral part of this Recommendation.)

F.1 Overview

This annex defines a JAVA interface related to media playback.

F.2 Media processing module

The media processing module defines interface classes that provide all media functions and related audio and video information classes. The media processing module is used to support media playback functions including DVB live broadcast, VOD on-demand, IP live broadcast, IP on-demand and local broadcast.

The most basic classes defined by the media processing module are:

– Media player class (MediaPlayer);

– Track information class such as audio, video, and subtitles (TrackInfo);

– Audio and video stream information class (MediaFormat).

The summary of the media processing module is shown in Table F.1.

Table F.1 – Summary of media processing module

Classes	
MediaPlayer	Provide interfaces for all media functions. Support includes DVB live broadcast, VOD on demand, IP live broadcast, IP on demand and local broadcast scenes.
TrackInfo	Describe track information such as audio, video, and subtitles.
MediaFormat	Use HashMap to store information such as audio and video streams.

F.2.1 Class org.tvos.media.MediaPlayer

Prototype: public class org.tvos.media.MediaPlayer

Description: It directly assumes all functions and provides interfaces for all media functions. Support includes DVB live broadcast, VOD on demand, IP live broadcast, IP on demand and local broadcast functions.

F.2.1.1 Constant field – cause of error

F.2.1.1.1 MEDIA_ERROR_UNKNOWN

Prototype: public static final int MEDIA_ERROR_UNKNOWN = 1

Description: Cause of error – unknown.

F.2.1.1.2 MEDIA_ERROR_SERVER_DIED

Prototype: public static final int MEDIA_ERROR_SERVER_DIED = 100

Description: Cause of error – media server died.

F.2.1.1.3 MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK

Prototype: public static final int MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK = 200

Description: Cause of error – Progressive playback illegal.

F.2.1.1.4 TVOS_MEDIA_ERROR_START_FAILD

Prototype:TVOS_MEDIA_ERROR_START_FAILD = 1000

Description: Cause of error – Playback failed.

F.2.1.1.5 TVOS_MEDIA_ERROR_SETPACE_FAILED

Prototype: public static final int TVOS_MEDIA_ERROR_SETPACE_FAILED = 1001

Description: Cause of error – Setting of double speed playback failed.

F.2.1.1.6 TVOS_MEDIA_ERROR_SEEK_FAILD

Prototype: public static final int TVOS_MEDIA_ERROR_SEEK_FAILD = 1002

Description: Cause of error – SEEK failed.

F.2.1.1.7 TVOS_MEDIA_ERROR_PAUSE_FAILD

Prototype: public static final int TVOS_MEDIA_ERROR_PAUSE_FAILD = 1003

Description: Cause of error – Pause failed.

F.2.1.1.8 TVOS_MEDIA_ERROR_RESUME_FAILD

Prototype: public static final int TVOS_MEDIA_ERROR_RESUME_FAILD = 1004

Description: Cause of error – Resume failed.

F.2.1.1.9 TVOS_MEDIA_ERROR_STOP_FAILD

Prototype: public static final int TVOS_MEDIA_ERROR_STOP_FAILD = 1005

Description: Cause of error – Stop failed.

Description: Cause of error – URL is valid.

F.2.1.1.10 TVOS_MEDIA_ERROR_URL_INVALID

Prototype: public static final int TVOS_MEDIA_ERROR_URL_INVALID = 1006

Description: Cause of error – URL is invalid.

F.2.1.1.11 TVOS_MEDIA_ERROR_RESOURCE_UNAVAILABLE

Prototype: public static final int TVOS_MEDIA_ERROR_RESOURCE_UNAVAILABLE = 1007

Description: Cause of error – Playback resource is unavailable.

F.2.1.1.12 TVOS_MEDIA_ERROR_AUDIO_DECODE_ERROR

Prototype: public static final int TVOS_MEDIA_ERROR_AUDIO_DECODE_ERROR = 1008

Description: Cause of error – Audio decoding failed.

F.2.1.1.13 TVOS_MEDIA_ERROR_VIDEO_DECODE_ERROR

Prototype: public static final int TVOS_MEDIA_ERROR_VIDEO_DECODE_ERROR = 1009

Description: Cause of error – Video decoding failed.

F.2.1.1.14 TVOS_MEDIA_ERROR_UNSUPPORT_VIDEO_DEC

Prototype: public static final int TVOS_MEDIA_ERROR_UNSUPPORT_VIDEO_DEC = 1010

Description: Cause of error – Unsupport video format decoding.

F.2.1.1.15 TVOS_MEDIA_ERROR_UNSUPPORT_AUDIO_DEC

Prototype: public static final int TVOS_MEDIA_ERROR_UNSUPPORT_AUDIO_DEC = 1011

Description: Cause of error – Unsupport video format decoding.

F.2.1.1.16 TVOS_MEDIA_ERROR_CONNECT_FAILED

Prototype: public static final int TVOS_MEDIA_ERROR_CONNECT_FAILED = 1012

Description: Cause of error – Failed to connect to the server, failed to establish a session or server return timeout.

F.2.1.1.17 TVOS_MEDIA_ERROR_VOD_SEARCH_FAILED

Prototype: public static final int TVOS_MEDIA_ERROR_VOD_SEARCH_FAILED = 1300

Description: Cause of error – Failed to search data during IPQAM playback.

F.2.1.1.18 TVOS_MEDIA_ERROR_VOD_OUT_OF_RANGE

Prototype: public static final int TVOS_MEDIA_ERROR_VOD_OUT_OF_RANGE = 1023

Description: Cause of error – The incoming time is out of the valid range.

F.2.1.2 Constant field – media information

F.2.1.2.1 MEDIA_INFO_UNKNOWN

Prototype: public static final int MEDIA_INFO_UNKNOWN = 1

Description: Media information – unknown.

F.2.1.2.2 MEDIA_INFO_STARTED_AS_NEXT

Prototype: public static final int MEDIA_INFO_STARTED_AS_NEXT = 2

Description: Media information – The next one starts to play.

F.2.1.2.3 MEDIA_INFO_RENDERING_START

Prototype: public static final int MEDIA_INFO_RENDERING_START = 3

Description: Media information – First frame begins to render.

F.2.1.2.4 MEDIA_INFO_VIDEO_TRACK_LAGGING

Prototype: public static final int MEDIA_INFO_VIDEO_TRACK_LAGGING = 700

Description: Media information – Video coding is complicated and the decoding performance is insufficient.

F.2.1.2.5 MEDIA_INFO_BUFFERING_START

Prototype: public static final int MEDIA_INFO_BUFFERING_START = 701

Description: Media information – Media begins to buffer.

F.2.1.2.6 MEDIA_INFO_BUFFERING_END

Prototype: public static final int MEDIA_INFO_BUFFERING_END = 702

Description: Media information – Media buffering ends.

F.2.1.2.7 MEDIA_INFO_NETWORK_BANDWIDTH

Prototype: public static final int MEDIA_INFO_NETWORK_BANDWIDTH = 703

Description: Media information – Network bandwidth.

F.2.1.2.8 MEDIA_INFO_BAD_INTERLEAVING

Prototype: public static final int MEDIA_INFO_BAD_INTERLEAVING = 800

Description: Media information – Media playback stalls.

F.2.1.2.9 MEDIA_INFO_NOT_SEEKABLE

Prototype: public static final int MEDIA_INFO_NOT_SEEKABLE = 801

Description: Media information – Media playback does not support SEEK.

F.2.1.2.10 MEDIA_INFO_METADATA_UPDATE

Prototype: public static final int MEDIA_INFO_METADATA_UPDATE = 802

Description: Media information – Media meta information updates.

F.2.1.2.11 MEDIA_INFO_EXTERNAL_METADATA_UPDATE

Prototype: public static final int MEDIA_INFO_EXTERNAL_METADATA_UPDATE = 803

Description: Media information – External metadata updates.

F.2.1.2.12 MEDIA_INFO_TIMED_TEXT_ERROR

Prototype: public static final int MEDIA_INFO_TIMED_TEXT_ERROR = 900

Description: Media information – Sync-subtitles error.

F.2.1.2.13 MEDIA_INFO_UNSUPPORTED_SUBTITLE

Prototype: public static final int MEDIA_INFO_UNSUPPORTED_SUBTITLE = 901

Description: Media information – Unsupported subtitle.

F.2.1.2.14 MEDIA_INFO_SUBTITLE_TIMED_OUT

Prototype: public static final int MEDIA_INFO_SUBTITLE_TIMED_OUT = 902

Description: Media information – Subtitle timeout.

F.2.1.2.15 TVOS_MEDIA_INFO_START_SUCCESS

Prototype: public static final int TVOS_MEDIA_INFO_START_SUCCESS = 1000

Description: Media information – Playback is successful.

F.2.1.2.16 TVOS_MEDIA_INFO_TUNE_LOCK_SUCCES

Prototype: public static final int TVOS_MEDIA_INFO_TUNE_LOCK_SUCCES = 1001

Description: Media information – Tuner frequency lock successfully or signal recovery.

F.2.1.2.17 TVOS_MEDIA_INFO_TUNE_LOCK_LOST

Prototype: public static final int TVOS_MEDIA_INFO_TUNE_LOCK_LOST = 1002

Description: Media information – Tuner signal is lost.

F.2.1.2.18 TVOS_MEDIA_INFO_NO_STREAM

Prototype: public static final int TVOS_MEDIA_INFO_NO_STREAM = 1003

Description: Media information – There is no stream information (or sudden stop of the stream).

F.2.1.2.19 TVOS_MEDIA_INFO_STREAM_RECOVER

Prototype: public static final int TVOS_MEDIA_INFO_STREAM_RECOVER = 1004

Description: Media information – Stream recover.

F.2.1.2.20 TVOS_MEDIA_INFO_AUDIO_DECODE_SUCCESS

Prototype: public static final int TVOS_MEDIA_INFO_AUDIO_DECODE_SUCCESS = 1005

Description: Media information – Audio decoding is successful.

F.2.1.2.21 TVOS_MEDIA_INFO_VIDEO_DECODE_SUCCESS

Prototype: public static final int TVOS_MEDIA_INFO_VIDEO_DECODE_SUCCESS = 1006

Description: Media information – Video decoding is successful.

F.2.1.2.22 TVOS_MEDIA_INFO_PACE_CHANGE

Prototype: public static final int TVOS_MEDIA_INFO_PACE_CHANGE = 1007

Description: Media information – Speed changes.

F.2.1.2.23 TVOS_MEDIA_INFO_DVB_CA_READY

Prototype: public static final int TVOS_MEDIA_INFO_DVB_CA_READY = 1100

Description: Media information – DVB CA is ready.

F.2.1.2.24 TVOS_MEDIA_INFO_DVB_CA_NOT_READY

Prototype: public static final int TVOS_MEDIA_INFO_DVB_CA_NOT_READY = 1101

Description: Media information – DVB CA is not ready.

F.2.1.2.25 TVOS_MEDIA_INFO_VOD_END_OF_STREAM

Prototype: public static final int TVOS_MEDIA_INFO_VOD_END_OF_STREAM = 1300

Description: Media information – Time-shifted channel or VOD movie is played to the end of the on-demand media stream.

F.2.1.2.26 TVOS_MEDIA_INFO_VOD_BEGIN_OF_STREAM

Prototype: public static final int TVOS_MEDIA_INFO_VOD_BEGIN_OF_STREAM = 1301

Description: Media information – Time-shifted channel or VOD movie is played to the beginning of the on-demand media stream.

F.2.1.3 Method

F.2.1.3.1 MediaPlayer

Prototype: public MediaPlayer()

Description: Construction method.

Parameter: None.

F.2.1.3.2 setDataSource

Prototype: void setDataSource(java.lang.String path)

Description: Setting the path of the resource. The path format played by digital television (DTV) contains three ways to match three scenarios, and the constraints are as follows.

Scenario 1: After the App searches for the program, it is played through the three elements of the program, the format is dvb://onid.tsid.sid. For multi-tuner scenes, it is expanded to dvb://onid.tsid.sid?tunerid=xxx.

Scenario 2: App does not need to search for programs, and plays by specifying frequency points and other program information to meet quick start. The format is dvbelement://frequency.symbolrate.modulation.serviceid.pmtpid.pcrpid.videotype.videopid.audiotype.audiopid. The frequency is measured in kilohertz, and all parameters are decimal, such as dvbelement://131000.6875.64.3.4.1.2.3.1.2. For multi-tuner scenes, it is expanded to dvbelement://frequency.symbolrate.modulation.serviceid.pmtpid.pcrpid.videotype.videopid.audiotype.audiopid?tunerid=xxx.

Scenario 3: App does not need to search for programs, and plays by specifying frequency points and other program information, to satisfy scrambling and multi-track playback when not searching for programs. The format is dvbelement://frequency.symbolrate.modulation.serviceid.-1. -1.-1.-1.-1.-1.-1. In this scene, the media engine parses pmtpid from the code stream according to serviceid, and parses pcrpid, videotype, videopid, audiotype, and audiopid according to pmtpid. If pmtpid exists and is valid, it does not search for pmtpid, and parses pcrpid, videotype, videopid, audiotype, and audiopid according to pmtpid. The frequency is measured in kilohertz, all numbers are decimal, and the invalid parameter is -1, such as dvbelement://131000.6875.64.3.-1.-1.-1.-1.-1.-1 or dvbelement://131000.6875.64.3.4.-1.-1.-1.-1.-1. For multi-tuner scenes, it is expanded to dvbelement://frequency.symbolrate.modulation.serviceid.-1.-1.-1.-1.-1.-1?tunerid=xxx.

Parameter: path – A java.lang.String object, indicating the resource path.

Return: None.

F.2.1.3.3 prepareAsync

Prototype: void prepareAsync()

Description: Prepare the player, asynchronously.

Parameter: None.

Return: None.

F.2.1.3.4 prepare

Prototype: void prepare()

Description: Prepare the player for playback, synchronously.

Parameter: None.

Return: None.

F.2.1.3.5 start

Prototype: void start()

Description: Start or resume playing.

Parameter: None.

Return: None.

F.2.1.3.6 pause

Prototype: void pause()

Description: Pause playing.

Parameter: None.

Return: None.

F.2.1.3.7 getDuration

Prototype: int getDuration()

Description: Getting a duration of a file.

Parameter: None.

Return: Int type, indicating the duration of the file.

F.2.1.3.8 getCurrentPosition

Prototype: int getCurrentPosition()

Description: Getting a current playing time.

Parameter: None.

Return: Int type, indicating the current playing position.

F.2.1.3.9 setVolume

Prototype: void setVolume(float volume)

Description: Setting a volume of this player.

Parameter: volume – float type, indicating a numerical value of the volume.

Return: None.

F.2.1.3.10 seekTo

Prototype: void seekTo(int pos)

Description: Seek to a specified position.

Parameter: pos – Int type, indicating the position to which it seeks.

Return: None.

F.2.1.3.11 reset

Prototype: void reset()

Description: Reset to the initial state of the player.

Parameter: None.

Return: None.

F.2.1.3.12 release

Prototype: void release()

Description: Release the resources associated with the player.

Parameter: None.

Return: None.

F.2.1.3.13 setOnInfoListener

Prototype: void setOnInfoListener(org.tvos.media.MediaPlayer.OnInfoListener listener)

Description: Register a callback to be called when information/warning is available.

Parameter: listener – An org.tvos.media.MediaPlayer.OnInfoListener, indicating an information listener.

Return: None.

F.2.1.3.14 setOnErrorListener

Prototype: void setOnErrorListener(org.tvos.media.MediaPlayer.OnErrorListener listener)

Description: Register a callback function to be called when an error occurs during asynchronous operation.

Parameter: listener – An org.tvos.media.MediaPlayer.OnErrorListener, indicating an error listener.

Return: None.

F.2.1.3.15 setOnCompletionListener

Prototype: void setOnCompletionListener(org.tvos.media.MediaPlayer.OnCompletionListener listener)

Description: Register a callback function to be called when the playing of the media source is finished.

Parameter: listener – An org.tvos.media.MediaPlayer.OnCompletionListener, indicating a playing end listener.

Return: None.

F.2.1.3.16 setOnPreparedListener

Prototype: void setOnPreparedListener(org.tvos.media.MediaPlayer.OnPreparedListener listener)

Description: Register the callback function to be called when the media source is ready to play.

Parameter: listener – An org.tvos.media.MediaPlayer.OnPreparedListener, indicating the listener when the media source is ready to play.

Return: None.

F.2.1.3.17 setOnBufferingUpdateListener

Prototype: public void setOnBufferingUpdateListener(org.tvos.media.MediaPlayer.OnBufferingUpdateListener listener)

Description: Register a callback function, which is called when the state of the network stream buffer changes.

Parameter: listener – An org.tvos.media.MediaPlayer.OnBufferingUpdateListener, indicating a status update listener of the buffer.

Return: None.

F.2.1.3.18 setDisplay

Prototype: void setDisplay(org.tvos.view.SurfaceHolder sh)

Description: Setting SurfaceHolder for media video playing.

Parameter: sh – An org.tvos.view.SurfaceHolder, indicating SurfaceHolder that needs to be set.

Return: None.

F.2.1.3.19 setAudioStreamType

Prototype: void setAudioStreamType(int streamtype)

Description: Setting audio stream type.

Parameter: streamtype – Int type, indicating the audio stream type.

Return: None.

F.2.1.3.20 setStopMode

Prototype: boolean setStopMode(int mode)

Description: Setting static frame mode. Set static frame mode: mode is 0 for black screen, and 1 for static frame. The setting mode can be called only once, and the subsequent cutting stations will keep this mode.

Parameter: mode – Int type, indicating the static frame mode of the player.

Return: boolean type, indicating whether the setting is successful, returning true if it succeeds, and returning false if it fails.

F.2.1.3.21 getStopMode

Prototype: int getStopMode()

Description: Getting the static frame mode of the player. Set static frame mode: mode is 0 for black screen, 1 for static frame.

Parameter: None.

Return: Int type, indicating a stop mode of the player.

F.2.1.3.22 getStartTime

Prototype: long getStartTime()

Description: Increase the start time of acquiring time-shifted (or watch back) programs.

Parameter: None.

Return: long type, indicating the start time of the time-shifted (or watch back) program.

F.2.1.3.23 getPace

Prototype: int getPace()

Description: Getting a speed in fast forward or fast backward.

Parameter: None.

Return: Int type, indicating the speed in fast forward or fast backward.

F.2.1.3.24 setPace

Prototype: void setPace(int pace)

Description: Setting a speed in fast forward or fast backward.

Parameter: pace – int type, indicating the speed in fast forward or fast backward.

Return: None.

F.2.1.3.25 setClip

Prototype: void setClip(org.tvos.graphics.Rect rect)

Description: Setting a clipping area.

Parameter: rect – Rect, indicating the clipping area.

Return: None.

F.2.1.3.26 getClip

Prototype: org.tvos.graphics.Rect getClip()

Description: Getting a clipping area.

Parameter: None.

Return: org.tvos.graphics.Rect type, indicating the clipping area.

F.2.1.3.27 clearVideo

Prototype: int clearVideo()

Description: Stop the stream and clear the screen. Usage scenario introduction: After the head-end stream is stopped, the engine will detect the stop of the stream, and will report the interruption message – TVOS_MEDIA_INFO_NO_STREAM. After the app receives the message, it needs to adjust the interface to clear the screen into a black screen.

Parameter: None.

Return: int type, returning 0 if the screen is cleared successfully, and -1 if it fails.

F.2.1.3.28 setVideoDisplay

Prototype: boolean setVideoDisplay(int mode)

Description: Setting video output.

Parameter: mode – Int type, indicating whether the video display is off or on, and the value is as follows.

TVOS_VIDEO_DISPLAY_CLOSE = 0: off;

TVOS_VIDEO_DISPLAY_OPEN = 1: on.

Return: boolean type, indicating whether the setting is successful, returning true if it succeeds, and false if it fails.

F.2.1.3.29 getVideoDisplay

Prototype: int getVideoDisplay()

Description: Getting video output mode.

Parameter: None.

Return: Int type, indicating whether the video display is off or on. The value is as follows.

TVOS_VIDEO_DISPLAY_CLOSE = 0: off;

TVOS_VIDEO_DISPLAY_OPEN = 1: on.

F.2.2 Class org.tvos.media.TrackInfo

Prototype: public class TrackInfo

Description: Describe track information such as audio, video, and subtitles.

F.2.2.1 Constant field – media type

F.2.2.1.1 MEDIA_TRACK_TYPE_UNKNOWN

Prototype: public static final int MEDIA_TRACK_TYPE_UNKNOWN = 0

Description: media type – unknown.

F.2.2.1.2 MEDIA_TRACK_TYPE_VIDEO

Prototype: public static final int MEDIA_TRACK_TYPE_VIDEO = 1

Description: media type – video type.

F.2.2.1.3 MEDIA_TRACK_TYPE_AUDIO

Prototype: public static final int MEDIA_TRACK_TYPE_AUDIO = 2

Description: media type – audio type.

F.2.2.1.4 MEDIA_TRACK_TYPE_TIMEDTEXT

Prototype: public static final int MEDIA_TRACK_TYPE_TIMEDTEXT = 3

Description: media type – Timed text type.

F.2.2.1.5 MEDIA_TRACK_TYPE_SUBTITLE

Prototype: public static final int MEDIA_TRACK_TYPE_SUBTITLE = 4

Description: media type – Subtitle/title type.

F.2.2.2 Method

F.2.2.2.1 getTrackInfo

Prototype: org.tvos.media.TrackInfo[] getTrackInfo()

Description: Getting all the operable audio stream information of the program currently being played.

1 Returning all track information here, including audio/video/subtitles/synchronized subtitles, etc.

2 Setting the audio track, you need to get the audio track part from the track.

Parameter: None.

Return: An org.tvos.media.TrackInfo array, indicating an array of track information.

F.2.2.2.2 selectTrack

Prototype: void selectTrack(int index)

Description: Choose a track.

Parameter: index – Int type, indicating the index of the track.

Return: None.

F.2.2.2.3 getSelectedTrack

Prototype: int getSelectedTrack(int trackType)

Description: Getting an index of the audio, video or subtitle track currently selected for playback.

Parameter: trackType – Int type, indicating the type of track. For details, see the constant field of the TrackInfo class.

Return: Int type, indicating the index of the audio, video or subtitle track currently selected for playback.

F.2.3 Class org.tvos.media.MediaFormat

Prototype: public class MediaFormat

Description: Use HashMap to store information such as audio and video streams.

F.2.3.1 Constant field – type description

F.2.3.1.1 MIMETYPE_VIDEO_VP8

Prototype: public static final java.lang.String MIMETYPE_VIDEO_VP8 = "video/x-vnd.on2.vp8"

Description: Type description – indicating MIME video type VP8.

F.2.3.1.2 MIMETYPE_VIDEO_VP9

Prototype: public static final java.lang.String MIMETYPE_VIDEO_VP9 = "video/x-vnd.on2.vp9"

Description: Type description – indicating MIME video type VP9.

F.2.3.1.3 MIMETYPE_VIDEO_AVC

Prototype: public static final java.lang.String MIMETYPE_VIDEO_AVC = "video/avc"

Description: Type description – indicating MIME video type AVC.

F.2.3.1.4 MIMETYPE_VIDEO_HEVC

Prototype: public static final java.lang.String MIMETYPE_VIDEO_HEVC = "video/hevc"

Description: Type description – indicating MIME video type HEVC.

F.2.3.1.5 MIMETYPE_VIDEO_MPEG4

Prototype: public static final java.lang.String MIMETYPE_VIDEO_MPEG4 = "video/mp4v-es"

Description: Type description – indicating MIME video type MPEG4.

F.2.3.1.6 MIMETYPE_VIDEO_H263

Prototype: public static final java.lang.String MIMETYPE_VIDEO_H263 = "video/3gpp"

Description: Type description – indicating MIME video type H263.

F.2.3.1.7 MIMETYPE_VIDEO_MPEG2

Prototype: public static final java.lang.String MIMETYPE_VIDEO_MPEG2 = "video/mpeg2"

Description: Type description – indicating MIME video type MPEG2.

F.2.3.1.8 MIMETYPE_VIDEO_RAW

Prototype: public static final java.lang.String MIMETYPE_VIDEO_RAW = "video/raw"

Description: Type description – indicating MIME video type RAW.

F.2.3.1.9 MIMETYPE_AUDIO_AMR_NB

Prototype: public static final java.lang.String MIMETYPE_AUDIO_AMR_NB = "audio/3gpp"

Description: Type description – indicating MIME audio type AMR_NB.

F.2.3.1.10 MIMETYPE_AUDIO_AMR_WB

Prototype: public static final java.lang.String MIMETYPE_AUDIO_AMR_WB = "audio/amr-wb"

Description: Type description – indicating MIME audio type AMR_WB.

F.2.3.1.11 MIMETYPE_AUDIO_MPEG

Prototype: public static final java.lang.String MIMETYPE_AUDIO_MPEG = "audio/mpeg"

Description: Type description – indicating MIME audio type MPEG.

F.2.3.1.12 MIMETYPE_AUDIO_AAC

Prototype: public static final java.lang.String MIMETYPE_AUDIO_AAC = "audio/mp4a-latm"

Description: Type description – indicating MIME audio type AAC.

F.2.3.1.13 MIMETYPE_AUDIO_QCELP

Prototype: public static final java.lang.String MIMETYPE_AUDIO_QCELP = "audio/qcelp"

Description: Type description – indicating MIME audio type QCELP.

F.2.3.1.14 MIMETYPE_AUDIO_VORBIS

Prototype: public static final java.lang.String MIMETYPE_AUDIO_VORBIS = "audio/vorbis"

Description: Type description – indicating MIME audio type VORBIS.

F.2.3.1.15 MIMETYPE_AUDIO_OPUS

Prototype: public static final java.lang.String MIMETYPE_AUDIO_OPUS = "audio/opus"

Description: Type description – indicating MIME audio type OPUS.

F.2.3.1.16 MIMETYPE_AUDIO_G711_ALAW

Prototype: public static final java.lang.String MIMETYPE_AUDIO_G711_ALAW = "audio/g711-alaw"

Description: Type description – indicating MIME audio type G711_ALAW.

F.2.3.1.17 MIMETYPE_AUDIO_G711_MLAW

Prototype: public static final java.lang.String MIMETYPE_AUDIO_G711_MLAW = "audio/g711-mlaw"

Description: Type description – indicating MIME audio type G711_MLAW.

F.2.3.1.18 MIMETYPE_AUDIO_RAW

Prototype: public static final java.lang.String MIMETYPE_AUDIO_RAW = "audio/raw"

Description: Type description – indicating MIME audio type.

F.2.3.1.19 MIMETYPE_AUDIO_FLAC

Prototype: public static final java.lang.String MIMETYPE_AUDIO_FLAC = "audio/flac"

Description: Type description – indicating MIME audio type FLAC.

F.2.3.1.20 MIMETYPE_AUDIO_MSGSM

Prototype: public static final java.lang.String MIMETYPE_AUDIO_MSGSM = "audio/gsm"

Description: Type description – indicating MIME audio type MSGSM.

F.2.3.1.21 MIMETYPE_AUDIO_AC3

Prototype: public static final java.lang.String MIMETYPE_AUDIO_AC3 = "audio/ac3"

Description: Type description – indicating MIME audio type AC3.

F.2.3.1.22 MIMETYPE_TEXT_VTT

Prototype: public static final java.lang.String MIMETYPE_TEXT_VTT = "text/vtt"

Description: Type description – indicating MIME text type VTT.

F.2.3.1.23 MIMETYPE_TEXT_CEA_608

Prototype: public static final java.lang.String MIMETYPE_TEXT_CEA_608 = "text/cea-608"

Description: Type description – indicating MIME text type CEA_608.

F.2.3.2 Constant field – property description

F.2.3.2.1 KEY_MIME

Prototype: public static final java.lang.String KEY_MIME = "mime"

Description: Property description – indicating the type of format.

F.2.3.2.2 KEY_LANGUAGE

Prototype: public static final java.lang.String KEY_LANGUAGE = "language"

Description: Property description – indicating language of content.

F.2.3.2.3 KEY_SAMPLE_RATE

Prototype: public static final java.lang.String KEY_SAMPLE_RATE = "sample-rate"

Description: Property description – indicating sampling rate of audio format.

F.2.3.2.4 KEY_CHANNEL_COUNT

Prototype: public static final java.lang.String KEY_CHANNEL_COUNT = "channel-count"

Description: Property description – indicating the number of channels in the audio format.

F.2.3.2.5 KEY_WIDTH

Prototype: public static final java.lang.String KEY_WIDTH = "width"

Description: Property description – indicating width of the content in the video format.

F.2.3.2.6 KEY_HEIGHT

Prototype: public static final java.lang.String KEY_HEIGHT = "height"

Description: Property description – indicating the height of the content of the video format.

F.2.3.2.7 KEY_MAX_WIDTH

Prototype: public static final java.lang.String KEY_MAX_WIDTH = "max-width"

Description: Property description – indicating the maximum expected width of the content in the video decoder format to prevent the resolution of the video content from changing. The associated value is an integer.

F.2.3.2.8 KEY_MAX_HEIGHT

Prototype: public static final java.lang.String KEY_MAX_HEIGHT = "max-height"

Description: Property description – indicating the maximum expected height decode format of the video content, if the video content resolution changes. The associated value is an integer.

F.2.3.2.9 KEY_MAX_INPUT_SIZE

Prototype: public static final java.lang.String KEY_MAX_INPUT_SIZE = "max-input-size"

Description: Property description – indicating the maximum size of the mediaformat byte described by the data in a buffer. The associated value is an integer.

F.2.3.2.10 KEY_BIT_RATE

Prototype: public static final java.lang.String KEY_BIT_RATE = "bitrate"

Description: Property description – indicating a bit rate in bits per second. The associated value is an integer.

F.2.3.2.11 KEY_STREAM_PID

Prototype: public static final java.lang.String KEY_STREAM_PID = "stream-pid"

Description: Property description – indicating PID of audio and video ES. The associated value is shaping.

F.2.3.2.12 KEY_COLOR_FORMAT

Prototype: public static final java.lang.String KEY_COLOR_FORMAT = "color-format"

Description: Property description – indicating color format of the content in the video format.

F.2.3.2.13 KEY_CAPTURE_RATE

Prototype: public static final java.lang.String KEY_CAPTURE_RATE = "capture-rate"

Description: Property description – indicating a capture rate of the video format in frames per second.

F.2.3.2.14 KEY_I_FRAME_INTERVAL

Prototype: public static final java.lang.String KEY_I_FRAME_INTERVAL = "i-frame-interval"

Description: Property description – indicating the frequency of I frames between I frames in a few seconds.

F.2.3.2.15 KEY_TEMPORAL_LAYERING

Prototype: public static final java.lang.String KEY_TEMPORAL_LAYERING = "ts-schema"

Description: Property description – indicating a temporal hierarchical mode.

F.2.3.2.16 KEY_REPEAT_PREVIOUS_FRAME_AFTER

Prototype: public static final java.lang.String KEY_REPEAT_PREVIOUS_FRAME_AFTER = "repeat-previous-frame-after"

Description: Property description – indicating that it is only applied when the video encoder is configured in "Surface Input" mode. The associated value is a long type value and gives the time in microseconds. After that, if no new frame is available, the frame previously submitted to the encoder will be repeated (once).

F.2.3.2.17 KEY_PUSH_BLANK_BUFFERS_ON_STOP

Prototype: public static final java.lang.String KEY_PUSH_BLANK_BUFFERS_ON_STOP = "push-blank-buffers-on-shutdown"

Description: Property description – If specified when configuring the video decoder to a surface, it will make the decoder output "blank", that is, the black frame to the surface when it stops clearing the previously displayed content. The relevant value is an integer with a value of 1.

F.2.3.2.18 KEY_DURATION

Prototype: public static final java.lang.String KEY_DURATION = "durationUs"

Description: Property description – indicating the duration (microseconds).

F.2.3.2.19 KEY_IS_ADTS

Prototype: public static final java.lang.String KEY_IS_ADTS = "is-adts"

Description: Property description – Optional, indicating that if you are decoding AAC audio content, set this key to 1 to indicate that each audio frame is prefixed with the ADTS header.

F.2.3.2.20 KEY_CHANNEL_MASK

Prototype: public static final java.lang.String KEY_CHANNEL_MASK = "channel-mask"

Description: Property description – Optional, indicating mask assigned by the audio channel.

F.2.3.2.21 KEY_AAC_PROFILE

Prototype: public static final java.lang.String KEY_AAC_PROFILE = "aac-profile"

Description: Property description – Optional, indicating that if the content is AAC audio, specify the required configuration file.

F.2.3.2.22 KEY_AAC_SBR_MODE

Prototype: public static final java.lang.String KEY_AAC_SBR_MODE = "aac-sbr-mode"

Description: Property description – Optional, indicating that if the content is AAC audio, specify the required SBR mode.

F.2.3.2.23 KEY_AAC_MAX_OUTPUT_CHANNEL_COUNT

Prototype: public static final java.lang.String KEY_AAC_MAX_OUTPUT_CHANNEL_COUNT = "aac-max-output-channel_count"

Description: Property description – Optional, indicating that if the content is AAC audio, specify the maximum number of channels output by the decoder.

F.2.3.2.24 KEY_AAC_DRC_TARGET_REFERENCE_LEVEL

Prototype: public static final java.lang.String KEY_AAC_DRC_TARGET_REFERENCE_LEVEL = "aac-target-ref-level"

Description: Property description – Optional, indicating that if the content is AAC audio, specify the target reference level.

F.2.3.2.25 KEY_AAC_ENCODED_TARGET_LEVEL

Prototype: public static final java.lang.String KEY_AAC_ENCODED_TARGET_LEVEL = "aac-encoded-target-level"

Description: Property description – Optional, indicating that if the content is AAC audio, specify the target reference level used by the encoder.

F.2.3.2.26 KEY_AAC_DRC_BOOST_FACTOR

Prototype: public static final java.lang.String KEY_AAC_DRC_BOOST_FACTOR = "aac-drc-boost-level"

Description: Property description – Optional, indicating that if the content is AAC audio, specify a DRC enhancement factor.

F.2.3.2.27 KEY_AAC_DRC_ATTENUATION_FACTOR

Prototype: public static final java.lang.String KEY_AAC_DRC_ATTENUATION_FACTOR = "aac-drc-cut-level"

Description: Property description – Optional, indicating that if the content is AAC audio, specify a DRC attenuation factor.

F.2.3.2.28 KEY_AAC_DRC_HEAVY_COMPRESSION

Prototype: public static final java.lang.String KEY_AAC_DRC_HEAVY_COMPRESSION = "aac-drc-heavy-compression"

Description: Property description – Optional, indicating that if the content is AAC audio, specify whether to use larger compression.

F.2.3.2.29 KEY_FLAC_COMPRESSION_LEVEL

Prototype: public static final java.lang.String KEY_FLAC_COMPRESSION_LEVEL = "flac-compression-level"

Description: Property description – Optional, indicating that if the content is FLAC audio, specify the desired compression level.

F.2.3.2.30 KEY_COMPLEXITY

Prototype: public static final java.lang.String KEY_COMPLEXITY = "complexity"

Description: Property description – indicating coding complexity.

F.2.3.2.31 KEY_PROFILE

Prototype: public static final java.lang.String KEY_PROFILE = "profile"

Description: Property description – indicating a profile that the encoder needs to use.

F.2.3.2.32 KEY_BITRATE_MODE

Prototype: public static final java.lang.String KEY_BITRATE_MODE = "bitrate-mode"

Description: Property description – indicating a bit rate mode required by the encoder application.

F.2.3.2.33 KEY_AUDIO_SESSION_ID

Prototype: public static final java.lang.String KEY_AUDIO_SESSION_ID = "audio-session-id"

Description: Property description – indicating a session ID of the audio.

F.2.3.2.34 KEY_IS_AUTOSELECT

Prototype: public static final java.lang.String KEY_IS_AUTOSELECT = "is-autoselect"

Description: Property description – indicating the independent choice of track, if not specified, the default is TRUE.

F.2.3.2.35 KEY_IS_DEFAULT

Prototype: public static final java.lang.String KEY_IS_DEFAULT = "is-default"

Description: Property description – indicating the independent choice of track, if not specified, the default is false.

F.2.3.2.36 KEY_IS_FORCED_SUBTITLE

Prototype: public static final java.lang.String KEY_IS_FORCED_SUBTITLE = "is-forced-subtitle"

Description: Property description – indicating a forced field for subtitles. If it is forced subtitles. If not specified, the default is forced to be false.

F.2.3.3 Method

F.2.3.3.1 MediaFormat

Prototype: public MediaFormat()

Description: Constructed function.

Parameter: None.

Return: None.

F.2.3.3.2 containsKey

Prototype: public final boolean containsKey(java.lang.String name)

Description: Determine whether there is information whose key value is name.

Parameter: name – A java.lang.String object, indicating key value;

Return: boolean type, return true if there is information whose key value is name, otherwise it returns false.

F.2.3.3.3 getInteger

Prototype: public final int getInteger(java.lang.String name)

Description: Getting integer property value whose key value is name.

Parameter: name – A java.lang.String object, indicating key value;

Return: Int type, indicating the integer property value whose key value is name. If the key does not exist, or the corresponding property value is not int type, an exception is thrown.

F.2.3.3.4 getLong

Prototype: public final long getLong(java.lang.String name)

Description: Getting a long integer property value whose key value is name.

Parameter: name – A java.lang.String object, indicating key value;

Return: long type, indicating the long integer property value whose key value is name. If the key does not exist, or the corresponding property value is not int type, an exception is thrown.

F.2.3.3.5 getFloat

Prototype: public final float getFloat(java.lang.String name)

Description: Getting a floating-point property value whose key value is name.

Parameter: name – A java.lang.String object, indicating key value;

Return: float type, indicating floating-point property value whose key value is name. If the key does not exist, or the corresponding property value is not int type, an exception is thrown.

F.2.3.3.6 getString

Prototype: public final String getString(java.lang.String name)

Description: Getting a string property value whose key value is name.

Parameter: name – A java.lang.String object, indicating key value;

Return: A String object, indicating the string property value whose key value is name. If the key does not exist or the corresponding property value is not a String object, an exception is thrown.

F.2.3.3.7 getByteBuffer

Prototype: public final ByteBuffer getByteBuffer(java.lang.String name)

Description: Getting a string property value whose key value is name.

Parameter: name – A java.lang.String object, indicating key value;

Return: A java.nio.ByteBuffer object, indicating the string property value whose key value is name. If the key does not exist or the corresponding property value is not a ByteBuffer object, an exception is thrown.

Annex G

System management unit

(This annex forms an integral part of this Recommendation.)

G.1 Overview

This annex defines JAVA interfaces related to system management.

G.2 System management module

The system management module provides classes and methods such as configuration parameter access, software and hardware configuration information acquisition, peripheral management and system operation.

The summary of the system management module is shown in Table G.1.

Table G.1 – Summary of system management module

Interfaces	
PeripheralType	Peripheral type constant definition interface supported by TVOS.
Peripheral	A Peripheral description interface, providing methods for obtaining the name, status, type, and ID of the peripheral.
PeripheralListener	Peripheral event listener, implemented by the application program.
Classes	
PeripheralManager	Peripheral manager is the entry class of the peripheral management module.
DataConfig	A Configuration data access class, providing a method to access the configuration data stored in the receiving terminal NVM.
HardwareInfo	A Hardware information description class, providing a method to obtain the hardware parameter information of the receiving terminal.
SoftwareInfo	A Software information description class, providing a method to obtain the software parameter information of the receiving terminal.
SysTool	A System tool class, providing methods for system standby, sleep and restart.
Events	
PeripheralEvent	Peripheral notification events.

G.2.1 Interface org.ngb.system.PeripheralType

Prototype: public interface org.ngb.system.PeripheralType

Description: Peripheral type constant definition interface supported by TVOS.

G.2.1.1 Constant field – peripheral type

G.2.1.1.1 PERIPHERAL_ALL

Prototype: pubic static final java.lang.String PERIPHERAL_ALL = "all"

Description: Peripheral type – all.

G.2.1.1.2 PERIPHERAL_MOUSE

Prototype: pubic static final java.lang.String PERIPHERAL_MOUSE = "mouse"

Description: Peripheral type – Mouse.

G.2.1.1.3 PERIPHERAL_PC_KEYBOARD

Prototype: pubic static final java.lang.String PERIPHERAL_PC_KEYBOARD = "keyboard"

Description: Peripheral type – PC keyboard.

G.2.2 Interface org.ngb.system.Peripheral

Prototype: public interface org.ngb.system.Peripheral

Description: A Peripheral description interface, providing methods for obtaining the name, status, type, and ID of the peripheral.

G.2.2.1 Method

G.2.2.1.1 getType

Prototype: pubic java.lang.String getType()

Description: Getting a peripheral type.

Return: A java.lang.String object, indicating the peripheral type. For the value, see the constant field definition of "peripheral type" of PeripheralType interface.

G.2.2.1.2 getID

Prototype: pubic long getID()

Description: Getting an unique identifier of the peripheral.

Parameter: None.

Return: long type, indicating a globally unique identifier of the peripheral.

G.2.2.1.3 getName

Prototype: pubic java.lang.String getName()

Description: Getting the name of the peripheral.

Parameter: None.

Return: A java.lang.String object, indicating the name of the peripheral.

G.2.2.1.4 getStatus

Prototype: pubic int getStatus()

Description: Getting peripheral status.

Parameter: None.

Return: Int type, indicating the status of the peripheral, and the value being related to the specific peripheral type.

G.2.3 Interface org.ngb.system.PeripheralListener

Prototype: public interface org.ngb.system.PeripheralListener

Description: Peripheral event listener, implemented by the application program.

G.2.3.1 Method

G.2.3.1.1 processPeripheralEvent

Prototype: pubic void processPeripheralEvent(org.ngb.system.PeripheralEvent event)

Description: Peripheral message processing method.

Parameter: event – An org.ngb.system.PeripheralEvent object, indicating the peripheral event message.

G.2.4 Class org.ngb.system.PeripheralManager

Prototype: public class org.ngb.system.PeripheralManager

Description: A Peripheral manager, providing a method of peripheral management, is the entry class of the peripheral management module.

G.2.4.1 Method

G.2.4.1.1 getInstance

Prototype: public static org.ngb.system.PeripheralManager getInstance()

Description: Getting the only instance of the org.ngb.system.PeripheralManager class implemented by the system.

Parameter: None.

Return: A PeripheralManager object, indicating the only instance of the org.ngb.system.PeripheralManager class implemented by the system.

G.2.4.1.2 addPeripheralEventListener

Prototype: public void addPeripheralEventListener(org.ngb.system.PeripheralListener listener)

Description: Register a peripheral event listener.

Parameter: listener – An org.ngb.system.PeripheralListener object, indicating the peripheral event listener to be registered.

Return: None.

G.2.4.1.3 removePeripheralListener

Prototype: public void removePeripheralListener(org.ngb.system.PeripheralListener listener)

Description: Unregister a peripheral event listener.

Parameter: listener – An org.ngb.system.PeripheralListener object, indicating the peripheral event listener to be unregistered.

Return: None.

G.2.4.1.4 getAllPeripheralsByType

Prototype: public org.ngb.system.Peripheral[] getAllPeripheralsByType(java.lang.String strType)

Description: Getting all devices of the specified type.

Parameter: strType – A java.lang.String object, indicating the peripheral type. For the value, see the constant field definition of "Peripheral Type" of the org.ngb.system.PeripheralType interface.

Return: org.ngb.system.Peripheral objects array, indicating all devices conforming to the specified type. If there are no eligible peripherals, the length of the returned array is 0.

G.2.4.1.5 getPeripheralsByID

Prototype: public org.ngb.system.Peripheral getPeripheralsByID(long id)

Description: Getting peripherals based on the globally unique ID.

Parameter: id – long type, indicating the globally unique ID of the peripheral.

Return: An org.ngb.system.Peripheral object, indicating the peripherals that meet the conditions. If there are no eligible peripherals, null is returned.

G.2.4.1.6 uninstallPeripheralByID

Prototype: public boolean uninstallPeripheralByID(long id)

Description: Uninstall peripherals based on the globally unique ID.

Parameter: id – long type, indicating the globally unique ID of the peripheral.

Return: boolean type, true value indicating that the uninstallation was successful, and false value indicating that the uninstallation failed.

G.2.5 Class org.ngb.system.DataConfig

Prototype: public class org.ngb.system.DataConfig

Description: Configuration data access class, providing a method to access a system data table stored in the receiving terminal NVM.

A system data table is used to store global configuration information set by the system. It is accessed in the way of "key name + key value". The key value is stored in JSON format in RAM and NVM, and the key name and key value are disclosed in JSON format. An authorized application can set/read system parameters, but cannot delete data items. When the application reads the data, it should be parsed according to the application scenario corresponding to the key name. The system should verify the authority of the application, and only privileged applications can access the system data table. JSON format definition of the key name and key value of the system data table is shown in Table G.2.

The access to the user data table adopts the method provided by the standard Properties interface.

Table G.2 – Key name and key value table of system data table

Key Name	Key Value	
DVBMainFrequencyInfo	Use	Describe DVB main frequency point information

	JSON format	[{
		"deliveryType":1,
		"deliveryParams":[
		{"frequency":626000, "symbolRate":6875, "modulation":3},
		{"frequency":634000, "symbolRate":6875, "modulation":3},
		{"frequency":642000, "symbolRate":6875, "modulation":3}
		]
		},
		...
		{
		"deliveryType":10,
		"deliveryParams":[
		{"frequency":12020, "symbolRate":28800, "polarization":3}
		]
		},
		{
		"deliveryType":12,
		"deliveryParams":[
		{"frequency":642000}
		]
		}]
		NOTE – The JSON format of the key value of the DVBMainFrequencyInfo key can describe multiple transmission systems, and each transmission system can describe multiple main frequency point information. In this example, the above JSON string describes the DVB-C, ABS-SS, and DTMB transmission systems, where DVB-C

Table G.2 – Key name and key value table of system data table

Key Name	Key Value		
		transmission system contains 3 main frequency point information, and ABS-SS transmission system contains 1 main frequency point information. DTMB transmission system includes 1 main frequency point information.	
DVBMainFrequencyInfo	Description	deliveryType	Int type, indicating the type of the delivery system. For the values, refer to the constant field definition of the DeliverySystemType interface and the DvbDeliverySystemType interface "Delivery System Type".
		Frequency	Int type, indicating the tuning frequency, and is measured in a unit related to the value of the deliveryType field: – If deliveryType=1, it indicates DVB-C delivery system, in kHz; – If deliveryType=10, it indicates ABS-SS delivery system, in MHz; – If deliveryType=12, it indicates DTMB delivery system, in kHz; – If deliveryType is another value, this key is meaningless.
		symbolRate	Int type, indicating the symbol rate, in ksymbol/s. – If deliveryType=12, it indicates DTMB delivery system, this key is meaningless.
		modulation	Int type, indicating a modulation mode, and the value is related to the value of the deliveryType field: – If deliveryType=1, it indicates DVB-C delivery system. For the value, please refer to the constant field definition of "modulation mode" of DvbcTunningParameters class; – If deliveryType is another value, this key is meaningless.
		Polarization	Int type, indicating polarization mode, the value is related to the value of the deliveryType field: – If deliveryType=10, it indicates ABS-SS delivery system. For the value, please refer to the "Polarization Mode" constant field definition of AbsssTunningParameters class; – If deliveryType is another value, this key is meaningless.
EPGSetting	Use	Describe EPG search setting information.	
	JSON Format	{"search_start_date":0, "search_days":7, "program_event_maxcount":255}	
	Description	search_start_date	int type, indicating the start date for searching the program table. The default is 0, indicating that the current day will be the start date of the search, 1 indicating the

Table G.2 – Key name and key value table of system data table

Key Name	Key Value		
			second day, 2 indicating the third day,..., and so on.
		search_days	Int type, indicating how many consecutive days the program table will search.
		program_event_max_count	Int type, indicating the maximum number of EPG searching program events. If the value is -1, it indicates unlimited.
AudioSetting	Use	Describe audio settings information.	

	JSON Format	{
		"enableGlobalVolume":0,
		"outputVolume":50,
		"spdifMode":"PCM",
		"soundMode":0
		}	

AudioSetting	Description	enableGlobalVolume	Int type, indicating whether to enable unified volume control, value: – 0-indicating that the application allows users to individually set the volume of each channel; – 1-indicating that the volume of all live TV, audio broadcast, near video on demand (NVOD), and mosaic is unified as the outputVolume value.
		outputVolume	Int type, indicating that the receiving terminal outputs analog volume, the value range being 0-100, 0 indicating mute, 100 indicating maximum volume, and taking effect immediately after setting.
		spdifMode	Character string, indicating the signal format of the SPDIF output interface of the receiving terminal. The value is: – "ORIGINAL" – indicating original code output (not decoded); – "PCM" – indicating PCM format output (decoded); – "default" – indicating that the original code or PCM output is determined according to the stream_type field value of the PMT table; – "OFF" – indicating that SPDIF output is turned off.
		soundMode	Int type, indicating a channel mode. The value is: – 0-indicating stereo sound; – 1-indicating a single left channel, that is, there is no sound in the right channel; – 2-indicating a single right channel, that is, there is no sound in the left channel; – 3-indicating mixing.
VideoSetting	Use	Describe video settings information.	

	JSON format	{
		"videoStandard":27400,
		"OSDAlpha":0
		}	

Table G.2 – Key name and key value table of system data table

Key Name	Key Value		
	Description	videoStandard	Int type, indicating the video output format. For the value, see the constant field definition of "output format" of VideoSetting class.
		OSDAlpha	Int type, indicating the transparency of the OSD layer, the value range being 0-100, 0 indicating completely opaque, and 100 indicating completely transparent.
UserPreference	Use	Describe user preference setting information.	

	JSON Format	{
		"audioLang":"zho",
		"osdLang":"eng"
		}	

	Description	audioLang	A character string, indicating the audio language preferred by the user. Three-letter language code follows GB/T 4880.2-2000 standard.
		osdLang	A character string, indicating the user's preferred interface language. Three-letter language code follows GB/T 4880.2-2000 standard.
Portal	Use	Describe portal server address information.	

	JSON format	{
		"address":"http://ngb.com",
		"port":8080
		}	

	Description	address	A character string, indicating a portal server address.
		port	Int type, indicating an access port of the portal server.
NTP	Use	Describe network time protocol (NTP) server address information.	

	JSON format	{
		"address":"http://ntp.com",
		"port":8080
		}	

	Description	address	A character string, indicating NTP server address.
		port	Int type, indicating an access port of the NTP server.
VODChannel	Use	Describe VOD server address information.	
	JSON format	{ "MD5":"5A8B6493", "VODParams":[{ "frequency":634000, "symbolRate":6875, "modulation":3, "QAMName":1234 }, { "frequency":642000, "symbolRate":6875, "modulation":3, "QAMName":4321	

Table G.2 – Key name and key value table of system data table

Key Name	Key Value		

		},
		{
		"frequency":650000,
		"symbolRate":6875,
		"modulation":3,
		"QAMName":8888
		}
		]
		}
		NOTE – Only applicable to cable digital TV.	

VODChannel	Description	MD5	A character string, indicating MD5 code of configuration file.
		frequency	Int type, indicating the frequency of IPQAM frequency point, in kHz.
		symbolRate	Int type, indicating a symbol rate of IPQAM frequency point, in ksymbol/s.
		modulation	Int type, indicating IPQAM frequency point modulation mode. For the value, please refer to the constant field definition of "modulation mode" in the DvbcTunningParameters class.
		QAMName	Int type, indicating VOD area code.
UserInfo	Use	Describe end user information.	

	JSON format	{
		"account":"882012459",
		"customerGroup":"group1",
		"adminPassword":"12345"
		}	

	Description	account	A character string, indicating a device account (UserID, user account information synchronized by BOSS/SMS).
		customerGroup	A character string, indicating a user group where the terminal is located.
		adminPassword	A character string, indicating an administrator password, used to enter the system setting interface and watch locked channels.
Autodeployer	Use	Describe application automatic deployment information.	
	JSON format	{ "mode":"auto-ip", "ocPath":[{ "deliveryType":1, "deliveryParams":[{"frequency":626000, "symbolRate":6875, "modulation":3}, {"frequency":634000, "symbolRate":6875, "modulation":3}, {"frequency":642000, "symbolRate":6875, "modulation":3}] }, ...],	

Table G.2 – Key name and key value table of system data table

<table>
<tr><th>Key Name</th><th colspan="3">Key Value</th></tr>
<tr><td rowspan="3"></td><td></td><td colspan="2">"ipPath":[
{
"udpPath":"192.168.1.12",
"udpPort":8080,
"downloadTimeOut":60
},
...
]
}
NOTE – A one-way broadcast channel supports multiple delivery systems and multiple frequency points to issue XML signaling files; and a two-way broadband channel supports multiple UDP servers to issue XML signaling files.</td></tr>
<tr><td rowspan="2">Description</td><td>mode</td><td>A character string indicating the method of obtaining the XML signaling file for automatic deployment. The value is:
– "ip" – means to obtain the XML signaling file from the two-way broadband channel;
– "oc" – means to obtain the XML signaling file from the one-way broadcast channel;
– "auto-ip" – means self-adaptation, preferentially obtain XML signaling files from the two-way broadband channel;
– "auto-oc" – means self-adaptation, preferentially obtain XML signaling file from unidirectional broadcast channel.</td></tr>
<tr><td>ocPath</td><td>JSON object type, indicating OC download path of the xml signaling file that is automatically deployed.
– deliveryType：Int type, indicating the type of delivery system, same as the explanation of the DVBMainFrequencyInfo key;
– deliveryParams：JSON object, same as the explanation of the DVBMainFrequencyInfo key.</td></tr>
<tr><td>Autodeployer</td><td>Description</td><td>ipPath</td><td>JSON object type, indicating ip download path of the xml signaling file for automatic deployment.
– udpPath：A character string, indicating UDP server address;
– udpPort：Int type, indicating UDP service port;
– downloadTimeOut: Int type, indicating the application download timeout time, in seconds.</td></tr>
</table>

G.2.5.1 Method

G.2.5.1.1 getInstance

Prototype: public static DataConfig getInstance()

Description: Getting the only instance of the DataConfig class implemented by the system.

Parameter: None.

Return: DataConfig class singleton.

G.2.5.1.2 getProperty

Prototype: public java.lang.String getProperty(java.lang.String strItem)

Description: Getting the value of a data item in the system data table (read from memory).

Parameter: strItem – A java.lang.String object, indicating the key name of the data item.

Return: A java.lang.String object, indicating the key value of the data item, which is described by a JSON character string; if the data item does not exist, it returns null.

G.2.5.1.3 setProperty

Prototype: public int setProperty(java.lang.String strItem, java.lang.String strValue)

Description: Setting the value of a data item in the system data table (written to memory).

Parameter: strItem – A java.lang.String object, indicating the key name of the data item;

strValue – A java.lang.String object, indicating the key value of the data item, which is described by JSON character string.

Return: Int type, indicating the modification result, and the values are as follows:

– If the modification is successful, the return value is greater than 0;

– If the content modification fails due to unknown reasons, 0 will be returned;

– If the data item does not exist, – 1 is returned.

G.2.5.1.4 restoreDefault

Prototype: public boolean restoreDefault()

Description: Restore the system data table in NVM to the factory setting state, and update the system data table in memory synchronously.

Return: boolean type, true value indicating that restoration of the factory settings succeeds, and false value indicating that the restoration of the factory settings fails.

G.2.5.1.5 restoreFromNvm

Prototype: public boolean restoreFromNvm()

Description: Read the system data table in NVM into memory and overwrite the current data in memory.

Parameter: None.

Return: boolean type, indicating the read result, true value indicating read success, and false value indicating read failure.

G.2.5.1.6 saveToNvm

Prototype: public boolean saveToNvm()

Description: Write the system data table in memory into NVM, and overwrite the system data table in NVM.

Parameter: None.

Return: boolean type, indicating the write result, true value indicating the write success, and false value indicating the write failure.

G.2.6 Class org.ngb.system.HardwareInfo

Prototype: public class org.ngb.system.HardwareInfo

Description: Hardware configuration information class, providing a method to obtain the hardware configuration parameter information of the receiving terminal.

G.2.6.1 Constant field – hardware configuration item

G.2.6.1.1 FLASH_SIZE

Prototype: public static final java.lang.String FLASH_SIZE = "flash_size"

Description: A size of the receiving terminal's flash memory, in MB.

G.2.6.1.2 RAM_SIZE

Prototype: public static final java.lang.String RAM_SIZE = "ram_size"

Description: A size of the receiving terminal's flash memory, in MB.

G.2.6.1.3 RAM_TYPE

Prototype: public static final java.lang.String RAM_TYPE = "ram_type"

Description: The type of the receiving terminal's flash memory, the value is "SDRAM", "DDR", etc.

G.2.6.1.4 SOC_MODEL

Prototype: public static final java.lang.String SOC_MODEL = "soc_model"

Description: Model of main chip of receiving terminal.

G.2.6.1.5 SOC_FREQUENCY

Prototype: public static final java.lang.String SOC_FREQUENCY = "soc_frequency"

Description: Operating frequency of main chip of receiving terminal, in MHz.

G.2.6.1.6 SOC_PROVIDER

Prototype: public static final java.lang.String SOC_PROVIDER = "soc_provider"

Description: Name of provider of main chip of receiving terminal.

G.2.6.1.7 DEFINITION_TYPE

Prototype: public static final java.lang.String DEFINITION_TYPE = "definition_type"

Description: Definition type of receiving terminal, which can take the values "HD" and "SD".

G.2.6.1.8 HW_VERSION

Prototype: public static final java.lang.String HW_VERSION = "hw_version"

Description: Hardware version number of the receiving terminal.

G.2.6.1.9 STB_BRAND

Prototype: public static final java.lang.String STB_BRAND = "stb_brand"

Description: Brand name of the receiving terminal.

G.2.6.1.10 STB_MODEL

Prototype: public static final java.lang.String STB_MODEL = "stb_model"

Description: Model of the receiving terminal.

G.2.6.1.11 STB_PROVIDER

Prototype: public static final java.lang.String STB_PROVIDER = "stb_provider"

Description: Name of the provider of the receiving terminal.

G.2.6.1.12 STB_SERIAL_NUMBER

Prototype: public static final java.lang.String STB_SERIAL_NUMBER = "stb_serial_number"

Description: Serial number of the receiving terminal.

G.2.6.1.13 TRANSPORT_TYPE

Prototype: public static final java.lang.String TRANSPORT_TYPE = "transport_type"

Description: Transmission mode type of the receiving terminal, which can be a combination of "DVB-C", "DVB-S", and "DVB-T".

G.2.6.2 Method

G.2.6.2.1 getProperty

Prototype: public static java.lang.String getProperty(java.lang.String key)

throws IllegalArgumentException

Description: Getting hardware configuration information of the receiving terminal.

Parameter: key – A java.lang.String object, indicating keyword of hardware configuration property. For the value, refer to the constant field definition of "Hardware Configuration Item" of the HardwareInfo class.

Return: A java.lang.String object, indicating hardware configuration parameters.

Exception: IllegalArgumentException –If the hardware configuration item keyword is illegal, this exception is thrown.

G.2.7 Class org.ngb.system.SoftwareInfo

Prototype: public class org.ngb.system.SoftwareInfo

Description: A Software configuration information class, providing a method to get a software configuration parameter information of the receiving terminal.

G.2.7.1 Constant field – software configuration item

G.2.7.1.1 CA_NAME

Prototype: public static final java.lang.String CA_NAME = "ca_name"

Description: Name of CA module.

G.2.7.1.2 CA_PROVIDER

Prototype: public static final java.lang.String CA_PROVIDER = "ca_provider"

Description: Name of the provider of the CA module.

G.2.7.1.3 CA_VERSION

Prototype: public static final java.lang.String CA_VERSION = "ca_version"

Description: Version number of CA module.

G.2.7.1.4 DRVIER_VERSION

Prototype: public static final java.lang.String DRIVER_VERSION = "driver_version"

Description: Driver version number of receiving terminal.

G.2.7.1.5 LOADER_NAME

Prototype: public static final java.lang.String LOADER_NAME = "loader_name"

Description: Name of software update module (loader).

G.2.7.1.6 LOADER_PROVIDER

Prototype: public static final java.lang.String LOADER_PROVIDER = "loader_provider"

Description: Name of the provider of the software update module (loader).

G.2.7.1.7 LOADER_SIZE

Prototype: public static final java.lang.String LOADER_SIZE = "loader_size"

Description: Size of the software update module (loader), in KB.

G.2.7.1.8 LOADER_VERSION

Prototype: public static final java.lang.String LOADER_VERSION = "loader_version"

Description: Version of the software update module (loader).

G.2.7.1.9 MW_COPYRIGHT

Prototype: public static final java.lang.String MW_COPYRIGHT = "mw_copyright"

Description: Copyright information of the system software.

G.2.7.1.10 MW_NAME

Prototype: public static final java.lang.String MW_NAME = "mw_name"

Description: Name of the system software.

G.2.7.1.11 MW_NVM_SIZE

Prototype: public static final java.lang.String MW_NVM_SIZE = "mw_nvm_size"

Description: Flash memory space occupied by the system software, in KB.

G.2.7.1.12 MW_PLATFORM_LEVEL

Prototype: public static final java.lang.String MW_PLATFORM_LEVEL = "mw_platform_level"

Description: Platform level supported by the system software.

G.2.7.1.13 MW_PLATFORM_PROFILE

Prototype: public static final java.lang.String MW_PLATFORM_PROFILE = "mw_platform_profile"

Description: Platform grade supported by the system software.

G.2.7.1.14 MW_PROVIDER

Prototype: public static final java.lang.String MW_PROVIDER = "mw_provider"

Description: Name of the provider of the system software.

G.2.7.1.15 MW_RAM_SIZE

Prototype: public static final java.lang.String MW_RAM_SIZE = "mw_ram_size"

Description: Memory space occupied by the system software, in KB.

G.2.7.1.16 MW_RELEASE_DATE

Prototype: public static final java.lang.String MW_RELEASE_DATE = "mw_release_date"

Description: Release date of the system software.

G.2.7.1.17 MW_VERSION

Prototype: public static final java.lang.String MW_VERSION = "mw_version"

Description: Version number of the system software.

G.2.7.1.18 OS_NAME

Prototype: public static final java.lang.String OS_NAME = "os_name"

Description: Name of the operating system software.

G.2.7.1.19 OS_PROVIDER

Prototype: public static final java.lang.String OS_PROVIDER = "os_provider"

Description: Name of the provider of the operating system software.

G.2.7.1.20 OS_VERSION

Prototype: public static final java.lang.String OS_VERSION = "os_version"

Description: Version number of the operating system software.

G.2.7.2 Method

G.2.7.2.1 getProperty

Prototype: public static java.lang.String getProperty(java.lang.String key)

throws IllegalArgumentException

Description: Getting the software configuration information of the receiving terminal.

Parameter: key – A java.lang.String object, indicating the keyword of the software configuration item. For the value, please refer to the constant field definition of "Software Configuration Item" in the SoftwareInfo class.

Return: A java.lang.String object, indicating software configuration parameters.

Exception: IllegalArgumentException – If the software configuration item keyword is illegal, this exception is thrown.

G.2.8 Class org.ngb.system.SysTool

Prototype: public class org.ngb.system.SysTool

Description: System tool class, providing methods for system standby, sleep, and restart. The system should verify the permissions of the application, and only authorized applications can call the methods provided by this class.

– Standby: CPU of the receiving terminal is still working to run some background programs, such as push downloads, software upgrade monitoring, etc., while turning off all audio and video output. The receiving terminal looks as if has stopped working;
– Sleep: CPU of the receiving terminal stops working, completely cuts off the power of the CPU and the main board, and monitors the activation command sent by the remote control by an external single-chip or other means, and then starts the switching power supply to supply power to the CPU and the main board to realize remote boot.

The relevant methods of this class should be implemented according to the actual capabilities of the receiver.

G.2.8.1 Method

G.2.8.1.1 getInstance

Prototype: public static org.ngb.system.SysTool getInstance()

Description: Getting the only instance of the org.ngb.system.SysTool class implemented by the system.

Parameter: None.

Return: An org.ngb.system.SysTool object, indicating the only instance of the org.ngb.system.SysTool class implemented by the system.

G.2.8.1.2 getStandByStatus

Prototype: public boolean getStandByStatus()

Description: Getting the standby state.

Parameter: None.

Return: boolean type, true value indicating entering standby state, false indicating exiting standby state (that is, to enter working state).

G.2.8.1.3 reboot

Prototype: public void reboot()

Description: Restart the receiver.

Parameter: None.

Return: None.

G.2.8.1.4 sleep

Prototype: public void sleep()

Description: Control the receiver to enter the sleep state, and the CPU will power off and stop working.

Parameter: None.

Return: None.

G.2.8.1.5 standBy

Prototype: public void standBy()

Description: Control the receiver to enter the standby state. The CPU is still running.

Parameter: None.

Return: None.

G.2.8.1.6 wakeUp

Prototype: public void wakeUp()

Description: Wake up the receiver to enter the working state.

Parameter: None.

Return: None.

G.2.9 Event org.ngb.system.PeripheralEvent

Prototype: public abstract class org.ngb.system.PeripheralEvent

Description: Peripheral notification events.

G.2.9.1 Constant field – peripheral event type

G.2.9.1.1 TYPE_FOUND

Prototype: public static final int TYPE_FOUND = 0

Description: A peripheral is found, its information can be obtained, but the peripheral cannot be accessed.

G.2.9.1.2 TYPE_READY

Prototype: public static final int TYPE_READY = 1

Description: The peripheral is ready for access.

G.2.9.1.3 TYPE_ERROR

Prototype: public static final int TYPE_ERROR = 2

Description: The peripheral has an unrecoverable error and cannot be accessed.

G.2.9.1.4 TYPE_PLUGOUT

Prototype: public static final int TYPE_PLUGOUT = 3

Description: The peripheral has been removed.

G.2.9.2 Method

G.2.9.2.1 getPeripheral

Prototype: public org.ngb.system.Peripheral getPeripheral()

Description: Getting the peripheral that issued this event.

Parameter: None.

Return: An org.ngb.system.Peripheral object, indicating the peripheral that issued this event message.

G.2.9.2.2 getType

Prototype: public int getType()

Description: Getting the peripheral event type.

Parameter: None.

Return: Int type, indicating the peripheral event type. For the value, please refer to the "Peripheral Event Type" constant field definition of the org.ngb.system.PeripheralEvent class.

G.3 OTA upgrade module

OTA upgrade module provides classes and methods for OTA software upgrade detection and processing operations.

The summary of the OTA upgrade module is shown in Table G.3.

Table G.3 – Summary of OTA upgrade module

Interface	
OTAEventListener	OTA event listener is implemented by the application program.
Class	
OTAManager	OTA manager is the entry class of the OTA function module.
Event	
OTAEvent	OTA event.

G.3.1 Interface org.ngb.system.OTAEventListener

Prototype: public interface org.ngb.system.OTAEventListener

Description: OTA event listener is implemented by the application program.

G.3.1.1 Method

G.3.1.1.1 processEvent

Prototype: void processEvent(org.ngb.system.OTAEvent event)

Description: OTA upgrade message processing method.

Parameter: event – An org.ngb.system.OTAEvent object, indicating an OTA upgrade prompt message.

Return: None.

G.3.2 Class org.ngb.system.OTAManager

Prototype: public class org.ngb.system.OTAManager

Description: OTA manager. The system should verify the permissions of the application, and only privileged applications can call the methods provided by this class.

G.3.2.1 Method

G.3.2.1.1 getInstance

Prototype: public static org.ngb.system.OTAManager getInstance()

Description: Getting the only instance of the org.ngb.system.OTAManager class implemented by the system.

Parameter: None.

Return: An OTAManager object, indicating the only instance of the org.ngb.system.OTAManager class implemented by the system.

G.3.2.1.2 checkOTA

Prototype: public boolean checkOTA()

Description: Determine whether to deploy a new OTA upgrade on the front end. This method is mainly used to manually detect OTA upgrade information.

Return: boolean type, indicating the determination result, true value indicating that there is an OTA upgrade, and false value indicating that there is no OTA upgrade.

G.3.2.1.3 getOTAName

Prototype: public java.lang.String getOTAName()

Description: Getting the name of the OTA upgrade event, which is different from the name of the OTA provider and refers to the text description of the OTA upgrade event.

Parameter: None.

Return: A java.lang.String object, indicating the name of the OTA upgrade event. If the upgrade event name is not provided on the front end, null will be returned.

G.3.2.1.4 startOTA

Prototype: public boolean startOTA()

Description: An Asynchronous method, starting to upgrade, and returning immediately upon being called.

Parameter: None.

Return: boolean type, true value indicating that the OTA upgrade is started successfully, and false value indicating that the OTA upgrade has failed.

G.3.2.1.5 addOTAEventListener

Prototype: public void addOTAEventListener(org.ngb.system.OTAEventListener listener)

Description: Register an OTA event listener.

Parameter: listener – An org.ngb.system.OTAEventListener object, indicating the OTA event listener to be registered.

Return: None.

G.3.2.1.6 removeOTAEventListener

Prototype: public void removeOTAEventListener(org.ngb.system.OTAEventListener listener)

Description: Unregister an OTA event listener.

Parameter: listener – An org.ngb.system.OTAEventListener object, indicating the OTA event listener to be unregistered.

Return: None.

G.3.3 Event org.ngb.system.OTAEvent

Prototype: public class org.ngb.system.OTAEvent extends EventObject

Description: OTA event.

G.3.3.1 Constant field – upgrade type

G.3.3.1.1 OTA_FORCE

Prototype: public static final int OTA_FORCE = 0

Description: Mandatory upgrade. A new OTA upgrade package was deployed on the front end of the operation to notify the receiving terminal to perform OTA upgrade. After receiving the message, the application does not prompt the user, and directly calls the startOTA() method of the OTAManager class to directly force the OTA upgrade.

G.3.3.1.2 OTA_NORMAL

Prototype: public static final int OTA_NORMAL = 1

Description: Upgrade normally. A new OTA upgrade package was deployed on the front end of the operation to notify the receiving terminal to perform OTA upgrade. The application prompts the user after receiving the message, and calls the startOTA() method of the OTAManager class to upgrade upon acknowledgement of the user.

G.3.3.2 Method

G.3.3.2.1 getType

Prototype: public int getType()

Description: Getting OTA upgrade event type.

Parameter: None.

Return: Int type, indicating the type of the OTA upgrade event. The value can be OTA_FORCE or OTA_NORMAL. For details, see the constant field definition of "upgrade type"in the OTAEvent class.

G.4 Storage management module

The storage management module provides classes and methods for storage devices and storage device partition access.

The summary of the storage management module is shown in Table G.4.

Table G.4 – Summary of storage management module

Interface	
Storage	Describe storage device information, such as name, size, idle state, partition, etc.
StorageEventListener	Storage event listener, implemented by the application.
StoragePartition	Describe the partition information of the storage device, such as name, size, idle state, access path, partition type, etc.
Class	
StorageManager	Provide methods for managing storage devices and storage device partitions.
Event	
StorageEvent	Storage events related to storage devices.

G.4.1 Interface org.ngb.system.Storage

Prototype: public interface org.ngb.system.Storage

Description: Storage device description interface, providing methods for obtaining serial numbers, partitions, etc.

G.4.1.1 Method

G.4.1.1.1 getAllPartitions

Prototype: org.ngb.system.StoragePartition[] getAllPartitions()

Description: Getting all partition objects of the storage device.

Parameter: None.

Return: An org.ngb.system.StoragePartition object array, indicating all partitions of the storage device. If there is none, the length of the returned array is 0.

G.4.1.1.2 getSerialNumber

Prototype: java.lang.String getSerialNumber()

Description: Getting serial number of the storage device.

Parameter: None.

Return: A java.lan.String object, indicating the serial number of the storage device.

G.4.2 Interface org.ngb.system.StorageEventListener

Prototype: public interface org.ngb.system.StorageEventListener

Description: A Storage event listener, implemented by the application.

G.4.2.1 Method

G.4.2.1.1 processStorageEvent

Prototype: void processStorageEvent(org.ngb.system.StorageEvent event)

Description: Storage event processing method.

Parameter: event – An org.ngb.system.StorageEvent object, indicating a storage event.

Return: None.

G.4.3 Interface org.ngb.system.StoragePartition

Prototype: public interface org.ngb.system.StoragePartition

Description: A Storage partition description interface, providing methods for obtaining the name, size, idle state, and access path of the storage device partition.

G.4.3.1 Method

G.4.3.1.1 getID

Prototype: long getID()

Description: Getting the globally unique ID of the storage device partition.

Parameter: None.

Return: long type, indicating the globally unique ID of the storage device partition.

G.4.3.1.2 getName

Prototype: java.lang.String getName()

Description: Getting the partition name of the storage device.

Parameter: None.

Return: A java.lang.String object, indicating the partition name of the storage device.

G.4.3.1.3 getPath

Prototype: java.lang.String getPath()

Description: Getting access path of the storage device partition.

Parameter: None.

Return: A java.lang.String object, indicating the access path of the storage device partition.

G.4.3.1.4 getStatus

Prototype: java.lang.String getStatus()

Description: Getting a partition status of the storage device.

Parameter: None.

Return: A java.lang.String object, indicating the partition status of the storage device, such as "good", "unformatted", etc.

G.4.3.1.5 getFreeSize

Prototype: long getFreeSize()

Description: Getting a free storage space size of the storage device partition.

Parameter: None.

Return: long type, indicating the free storage space size of the storage device partition, in KB.

G.4.3.1.6 getTotalSize

Prototype: long getTotalSize()

Description: Getting a total storage space size of the storage device partition.

Parameter: None.

Return: long type, indicating the total storage space size of the storage device partition, in KB.

G.4.4 Class org.ngb.system.StorageManager

Prototype: public class org.ngb.system.StorageManager

Description: The storage device manager provides a method for managing storage devices and storage device partitions, and is the entry class of the storage device management function module. The system should verify the permissions of the application, and only authorized applications can call the methods provided by this class.

G.4.4.1 Method

G.4.4.1.1 getInstance

Prototype: public static org.ngb.system.StorageManager getInstance()

Description: Getting the only instance of the storage device manager implemented by the system.

Parameter: None.

Return: org.ngb.system.StorageManager object singleton.

G.4.4.1.2 addStorageEventListener

Prototype: public void addStorageEventListener(org.ngb.system.StorageEventListener listener)

Description: Register a storage event listener.

Parameter: listener – An org.ngb.system.StorageEventListener object, indicating the storage event listener to be registered.

Return: None.

G.4.4.1.3 removeStorageEventListener

Prototype: public void removeStorageEventListener(org.ngb.system.StorageEventListener listener)

Description: Unregister a storage event listener.

Parameter: listener – An org.ngb.system.StorageEventListener object, indicating the storage event listener to be unregistered.

G.4.4.1.4 getAllStorages

Prototype: public org.ngb.system.Storage[] getAllStorages()

Description: Getting all storage device objects.

Return: An org.ngb.system.Storage object array, indicating all storage device objects.

G.4.4.1.5 uninstallStorage

Prototype: public boolean uninstallStorage(org.ngb.system.Storage storage)

Description: Uninstall storage device.

Parameter: storage – An org.ngb.system.Storage object, indicating storage device information.

Return: boolean type, indicating the uninstallation result, true value indicating that the uninstallation was successful, and false value indicating that the uninstallation failed.

G.4.5 Event org.ngb.system.StorageEvent

Prototype: public class org.ngb.system.StorageEvent

Description: Storage events related to storage devices.

G.4.5.1 Constant field – message type

G.4.5.1.1 TYPE_PARTITION_FOUND

Prototype: public static final int TYPE_PARTITION_FOUND = 22

Description: Device message type-partition found.

G.4.5.1.2 TYPE_PARTITION_MOUNTED

Prototype: public static final int TYPE_PARTITION_MOUNTED = 23

Description: Device message type-succeeded to mount the partition.

G.4.5.1.3 TYPE_PARTITION_MOUNT_FAILED

Prototype: public static final int TYPE_PARTITION_MOUNT_FAILED = 24

Description: Device message type-failed to mount the partition.

G.4.5.1.4 TYPE_PARTITION_UNINSTALL

Prototype: public static final int TYPE_PARTITION_UNINSTALL = 25

Description: Device message type-partition uninstall message.

G.4.5.1.5 TYPE_INSUFFICIENT_SPACE

Prototype: public static final int TYPE_INSUFFICIENT_SPACE = 32

Description: A Storage event-there is not enough space.

G.4.5.2 Method

G.4.5.2.1 getType

Prototype: public int getType()

Description: Getting the storage event message type.

Parameter: None.

Return: Int type, indicating the message type of the storage event. For the value, see the constant field definition of the "message type" of the StorageEvent class.

G.4.5.2.2 getStorage

Prototype: public org.ngb.system.Storage getStorage()

Description: Getting the storage device object that sent this event.

Parameter: None.

Return: An org.ngb.system.Storage object, indicating the storage device object that sent this event.

G.4.5.2.3 getStoragePartition

Prototype: public org.ngb.system.StoragePartition getStoragePartition()

Description: Getting the storage device partition object that sent this event.

Parameter: None.

Return: An org.ngb.system.StoragePartition object, indicating the storage device partition object that sent this event.

Annex H

JAVA-application engine unit

(This annex forms an integral part of this Recommendation.)

H.1 Overview

This annex defines the application engine JAVA interface, including channel scan module, electronic program guide module and information search module.

H.2 Channel scan module

The channel scan module provides classes and methods related to channel scan.

The summary of channel scan module is shown in Table H.1.

Table H.1 – Overview of the channel scan module

Interface	
ChannelScanListener	The listener for the event of the search process is implemented by the application program.
Class	
ChannelScanEngine	Search engine, the entry class of the channel scan function unit.
Event	
ChannelScanEvent	A channel scan event, base class.
ChannelScanFailureEvent	A channel scan failure event, inheriting the ChannelScanEvent class.
ChannelScanFinishEvent	A channel scan end event, inheriting the ChannelScanEvent class.
ChannelScanNITSuccessEvent	A channel scan successfully parses NIT event, inheriting the ChannelScanEvent class.
ChannelScanSuccessEvent	A channel scan success event, inheriting the ChannelScanEvent class.

Channel scan method definition:

– Manual search – According to the set tuning and demodulation parameters, search for broadcast and TV program channels within a single frequency point;

– Automatic search – According to the starting channel tuning and demodulation parameters specified by the operator, search for NIT, and then automatically perform tuning and demodulation according to the instructions of NIT, and search for broadcast and TV program channels throughout the network; the operator may specify multiple starting frequency points, the automatic search is completed upon the NIT being found at any one frequency point among a plurality of starting frequency points that the operator may specify;

– Zone search – According to China's digital TV channel assignment table, search for broadcasting and TV program channels frequency point by frequency point within a designated frequency range.

H.2.1 Interafce org.ngb.toolkit.channelscan.ChannelScanListener

Prototype: public interface org.ngb.toolkit.channelscan.ChannelScanListener extends java.util.EventListener

Description: A listener for the event of the channel scan process, implemented by an application program.

H.2.1.1 Method

H.2.1.1.1 processEvent

Prototype: void processEvent(org.ngb.toolkit.channelscan.ChannelScanEvent event)

Description: Channel scan event handling method.

Parameter: event – An org.ngb.toolkit.channelscan.ChannelScanEvent object, indicating a channel scan event. The application further determines a prototype of the event object through the instanceof method.

Return: None.

H.2.2 Class org.ngb.toolkit.channelscan.ChannelScanEngine

Prototype: public class org.ngb.toolkit.channelscan.ChannelScanEngine

Description: Search engine, an entry class of a channel scan function unit.

H.2.2.1 Constant field – zone search method

H.2.2.1.1 CHANNELSCAN_TYPE_MANUAL

Prototype: public static final int CHANNELSCAN_TYPE_MANUAL = 0

Description: A Channel scan method – manual search.

H.2.2.1.2 CHANNELSCAN_TYPE_NIT

Prototype: public static final int CHANNELSCAN_TYPE_NIT = 1

Description: A Channel scan method – automatic search.

H.2.2.1.3 CHANNELSCAN_TYPE_ZONE

Prototype: public static final int CHANNELSCAN_TYPE_ZONE = 2

Description: A Channel scan method – zone search.

H.2.2.1.4 CHANNELSCAN_TYPE_JSON

Prototype: public static final int CHANNELSCAN_TYPE_JSON = 3

Description: A Channel scan method – JSON search. This type has a separate search entry function startScanExt.

H.2.2.2 Method

H.2.2.2.1 createInstance

Prototype: public static org.ngb.toolkit.channelscan.ChannelScanEngine createInstance(int tunerId)throws org.davic.mpeg.ResourceException

Description: Creating a zone search engine object.

Parameter: tunerId – Int type, indicating tunerid corresponding to the search engine to be created. If this parameter is not filled in, the default tunerid is 0.

Return: An org.ngb.toolkit.channelscan.ChannelScanEngine object, indicating a zone search engine.

Exception: org.davic.mpeg.ResourceException – If there are insufficient underlying resources, this exception is thrown.

H.2.2.2.2 addChannelScanListener

Prototype: public void addChannelScanListener (org.ngb.toolkit.channelscan.ChannelScanListener listener)

Description: Register a search process status listener.

Parameter: listener – An org.ngb.toolkit.channelscan.ChannelScanListener object, indicating a search process listener object to be registered.

Return: None.

H.2.2.2.3 removeChannelScanListener

Prototype: public void removeChannelScanListener(org.ngb.toolkit.channelscan.

org.ngb.toolkit.channelscan.ChannelScanListener listener)

Description: Unregister a search process status listener.

Parameter: listener – An org.ngb.toolkit.channelscan.ChannelScanListener object, indicating a search process listener object to be unregistered.

Return: None.

H.2.2.2.4 startScan

Prototype: public void startScan(int type, org.ngb.broadcast.dvb.tuner.TuningParameters[] params)

throws java.lang.IllegalArgumentException,org.davic.mpeg.ResourceException

Description: An Asynchronous method, starting channel scan. The search results are sent to the application through the search event org.ngb.toolkit.channelscan.ChannelScanEvent.

– When NIT table is searched, send org.ngb.toolkit.channelscan.ChannelScanNITSuccessEvent;
– When a frequency point is searched, send org.ngb.toolkit.channelscan.ChannelScanSuccessEvent;
– When the search is over, send org.ngb.toolkit.channelscan.ChannelScanFinishEvent;
– When the search process fails due to various reasons, send org.ngb.toolkit.channelscan.ChannelScanFailureEvent. The application can get the specific reason of the failure through ChannelScanFailureEvent.getReason().

Parameter: type – type-Int type, indicating a search type, which can be CHANNELSCAN_TYPE_MANUAL, CHANNELSCAN_TYPE_NIT or CHANNELSCAN_TYPE_ZONE, see the channel scan method constant field definition for details;

params-org.ngb.broadcast.dvb.tuner.TuningParameters object array, indicating search tuning demodulation parameters. The length of the array is related to the search type:

– If type = CHANNELSCAN_TYPE_MANUAL, the length of the array is 1;
– If type = CHANNELSCAN_TYPE_NIT, the length of the array is equal to the number of starting frequency points deployed by the operator;
– If type = CHANNELSCAN_TYPE_ZONE, the length of the array is 2, params[0] indicating the tuning parameters for starting the search frequency point, and params[1] indicating the tuning parameters for ending the search frequency point.

Return: None.

Exception: java.lang.IllegalArgumentException – If the parameter is invalid, this exception is thrown;

org.davic.mpeg.ResourceException – If there are insufficient underlying resources, this exception is thrown.

H.2.2.2.5 startScanExt

Prototype: public void startScanExt(org.ngb.broadcast.dvb.tuner.TuningParameters params,int pid, int tableid)

throws java.lang.IllegalArgumentException, org.davic.mpeg.ResourceException

Description: An Asynchronous method, starting JSON format program information search. Corresponding search type: CHANNELSCAN_TYPE_JSON, the result is sent to the application through the event org.ngb.toolkit.channelscan.ChannelScanEvent.

– When the search is over, send org.ngb.toolkit.channelscan.ChannelScanFinishEvent;
– When the search process fails due to various reasons, send org.ngb.toolkit.channelscan.ChannelScanFailureEvent. The application can get the specific reason of the failure through org.ngb.toolkit.channelscan.ChannelScanFailureEvent.getReason().
– When the search is started and needs to be forcedly canceled, the cancel() function is called in accordance with other search types.

Parameter: params – An org.ngb.broadcast.dvb.tuner.TuningParameters object, indicating tuning and demodulation parameters of the frequency point where the program information data is located.

pid – PID of TS packet where the program information data is located.

tableid – tableid assigned by the program information data.

Return: None.

Exception: java.lang.IllegalArgumentException – If the parameter is invalid, this exception is thrown;

org.davic.mpeg.ResourceException – If there are insufficient underlying resources, this exception is thrown.

H.2.2.2.6 cancel

Prototype: public void cancel()

Description: An Asynchronous method, canceling channel scan. Before the application captures a search failure (org.ngb.toolkit.channelscan.ChannelScanFailureEvent) or a search finish (org.ngb.toolkit.channelscan.ChannelScanFinishEvent) event, you can cancel the channel scan by calling this method. After successfully canceling the search, the org.ngb.toolkit.channelscan.ChannelScanFinishEvent event will be sent to the application. If the channel scan is not started, calling this method will not perform any action.

Parameter: None.

Return: None.

H.2.2.2.7 release

Prototype: public void release()

Description: Instructing the system to release resources used by the search engine. If the search engine is running, you must cancel the search before calling this method.

Parameter: None.

Return: None.

H.2.2.2.8 saveScanResult

Prototype: public boolean saveScanResult()

Description: Saving the results of this search to NVM.

Return: boolean type, true value indicating that the save succeeded, and false value indicating that the save failed.

H.2.2.2.9 saveServicesInfo

Prototype: public boolean saveServicesInfo(java.lang.String jsonSIInfo)

Description: The program information data in JSON format is transferred to the DTV component for analysis and storage.

Since it is only data analysis and does not consume too much time, the synchronous mode is adopted.

Parameter: jsonSIInfo – Apply the PSI/SI information in JSON format obtained from the operator's front-end IP.

Return: boolean type, true value indicating the parsing and saving is successful, false indicating the data format parsing fails.

H.2.3 Event org.ngb.toolkit.channelscan.ChannelScanEvent

Prototype: public abstract class org.ngb.toolkit.channelscan.ChannelScanEvent

extends java.util.EventObject

Description: Base class for channel scan events.

H.2.4 Event org.ngb.toolkit.channelscan.ChannelScanFailureEvent

Prototype: public class org.ngb.toolkit.channelscan.ChannelScanFailureEvent

extends org.ngb.toolkit.channelscan.ChannelScanEvent

Description: A Channel scan process failure event, the application can get the specific reason for the search failure through the getReason() method.

H.2.4.1 Constant field – channel scan failure reason

H.2.4.1.1 REASON_UNKOWN

Prototype: public static int REASON_UNKOWN = 0

Description: channel scan failure reason–unknown reason.

H.2.4.1.2 REASON_TUNE_LOCK_FAILED

Prototype: public static int REASON_TUNE_LOCK_FAILED = 1

Description: channel scan failure reason–Frequency lock failed.

H.2.4.1.3 REASON_NIT_SEARCH_FAILED

Prototype: public static int REASON_NIT_SEARCH_FAILED = 2

Description: channel scan failure reason–NIT search failed.

H.2.4.1.4 REASON_BAT_SEARCH_FAILED

Prototype: public static int REASON_BAT_SEARCH_FAILED = 3

Description: channel scan failure reason–BAT search failed.

H.2.4.1.5 REASON_PAT_SEARCH_FAILED

Prototype: public static int REASON_PAT_SEARCH_FAILED = 4

Description: channel scan failure reason–PAT search failed.

H.2.4.1.6 REASON_PMT_SEARCH_FAILED

Prototype: public static int REASON_PMT_SEARCH_FAILED = 5

Description: channel scan failure reason–PMT search failed.

H.2.4.2 Method

H.2.4.2.1 getReason

Prototype: public int getReason()

Description: Getting the reason for the search failure.

Return: Int type, indicating the specific reason for the failure. For the value, please refer to the constant field definition of search failure reason of the org.ngb.toolkit.channelscan.ChannelScanFailureEvent class.

H.2.5 Event org.ngb.toolkit.channelscan.ChannelScanFinishEvent

Prototype: public class org.ngb.toolkit.channelscan.ChannelScanFinishEvent

extends org.ngb.toolkit.channelscan.ChannelScanEvent

Description: Search process finish event.

H.2.5.1 Method

H.2.5.1.1 getServiceCount

Prototype: public int getServiceCount()

Description: Getting the number of services found in the search.

Parameter: None.

Return: Int type, indicating the number of services.

H.2.5.1.2 getTransportStreamCount

Prototype: public int getTransportStreamCount()

Description: Getting the number of transport streams obtained by the search.

Parameter: None.

Return: Int type, indicating the number of transport streams.

H.2.6 Event org.ngb.toolkit.channelscan.ChannelScanNITSuccessEvent

Prototype: public class org.ngb.toolkit.channelscan.ChannelScanNITSuccessEvent

extends org.ngb.toolkit.channelscan.ChannelScanEvent

Description: channel scan success parse NIT event.

H.2.6.1 Method

H.2.6.1.1 getTransportStream

Prototype: public org.ngb.broadcast.dvb.si.SITransportStream[] getTransportStream()

Description: Getting the transport stream object described in the NIT table.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.TransportStream object array, indicating the transport stream object described in the NIT table.

H.2.7 Event org.ngb.toolkit.channelscan.ChannelScanSuccessEvent

Prototype: public class org.ngb.toolkit.channelscan.ChannelScanSuccessEvent

extends org.ngb.toolkit.channelscan.ChannelScanEvent

Description: Channel scan success event. The application can get the successfully found service object through the getResult() method.

H.2.7.1 Method

H.2.7.1.1 getResult

Prototype: public org.ngb.broadcast.dvb.si.SIService[] getResult()

Description: Getting search results.

Parameter: None.

Return: An org.ngb.broadcast.dvb.si.SIService object array, indicating search results.

H.3 Electronic Program Guide Module

The Electronic Program Guide (EPG) provides end users with a way to browse broadcasting service-related information, such as service name, program start and end time, content summary, etc., so that end users can quickly retrieve and access services.

The electronic program guide module provides classes and methods for obtaining EPG information. EPG information can be obtained using a cache (Cache) mechanism, or it can be temporarily loaded when needed. If a cache mechanism is adopted, the EPG information should be monitored in real time to ensure that the application can extract the latest EPG information.

The summary of the electronic program guide module is shown in Table H.2.

Table H.2 – Summary of Electronic Program Guide Module

Interface	
ProgramEvent	Describing a certain program event information.
ProgramEventFilter	Defining a filter interface used by the application to query program event information from the EPG module, which is implemented by the application layer.
ProgramService	Describing a certain program service information, which is a package of a group of program event information belonging to the same service.
ProgramServiceFilter	Defining a filter interface used by the application to query program service information from the EPG module, which is implemented by the application layer.
EPGUpdateListener	EPG information update event listener, implemented by the application program.
Class	
EPGManager	EPG manager, an entry class for obtaining EPG information.
Event	
EPGUpdateEvent	EPG information update event.

H.3.1 Interface org.ngb.toolkit.epg.ProgramEvent

Prototype: public interface org.ngb.toolkit.epg.ProgramEvent

Description: Program event information description class.

H.3.1.1 Method

H.3.1.1.1 getTitle

Prototype: public java.lang.String getTitle()

Description: Getting the name of the program event.

Parameter: None.

Return: A java.lang.String object, indicating the name of the program event.

H.3.1.1.2 getShortDescription

Prototype: public java.lang.String getShortDescription()

Description: Getting a brief introduction to the program event.

Parameter: None.

Return: A java.lang.String object, indicating the brief introduction to the program event.

H.3.1.1.3 getBeginDate

Prototype: public java.util.Date getBeginDate()

Description: Getting a start time of the program event.

Parameter: None.

Return: A java.util.Date object, indicating the start time of the program event.

H.3.1.1.4 getDuration

Prototype: public long getDuration()

Description: Getting the time length of the program event.

Parameter: None.

Return: long type, indicating the duration of the program event, in seconds.

H.3.1.1.5 getEndDate

Prototype: public java.util.Date getEndDate()

Description: Getting an end time of the program event.

Parameter: None.

Return: A java.util.Date object, indicating the end time of the program event.

H.3.1.1.6 getLanguageCode

Prototype: public java.lang.String getLanguageCode()

Description: Getting a coding language of the program event description information.

Parameter: None.

Return: A java.lang.String object, indicating the language of the program event description information, the three-letter language code follows GB/T 4880.2-2000.

H.3.1.1.7 getNibbles

Prototype: public byte[] getNibbles()

Description: Getting a content classification information of the program event.

Parameter: None.

Return: A byte type array, indicating the content classification information associated with the program event.

byte[0] – upper 4 bits indicate the second level of content classification (content_nibble_level_2); lower 4 bits indicate the first level of content classification (content_nibble_level_1);

byte[1] – upper 4 bits indicate the second level of user-defined classification (user_nibble_level_2); lower 4 bits indicate the first level of user-defined classification (user_nibble_level_1).

H.3.1.1.8 getCALockMode

Prototype: public boolean getCALockMode()

Description: Determine whether the program event requires CA authorization.

NOTE – How to identify whether a program event requires CA authorization is implemented by calling the CA related interface within the system. This specification does not make mandatory provisions for the implementation method.

Parameter: None.

Return: boolean type, true value indicating that the CA does not authorize the program, and false indicating that the CA authorizes the program.

H.3.1.1.9 getDvbLocator

Prototype: public org.davic.net.dvb.DvbNetworkBoundLocator getDvbLocator()

Description: Getting a locator of the program event.

Parameter: None.

Return: An org.davic.net.dvb.DvbNetworkBoundLocator object, indicating the locator of the program event.

H.3.1.1.10 getProgramService

Prototype: public org.ngb.toolkit.epg.ProgramService getProgramService()

Description: Getting a program service object to which this program event belongs.

Parameter: None.

Return: An org.ngb.toolkit.epg.ProgramService object, indicating the program service object to which this program event belongs.

H.3.2 Interface org.ngb.toolkit.epg.ProgramEventFilter

Prototype: public interface org.ngb.toolkit.epg.ProgramEventFilter

Description: An EPG program event query filter interface, implemented by the application layer.

H.3.2.1 Method

H.3.2.1.1 accept

Prototype: public boolean accept(org.ngb.toolkit.epg.ProgramEvent programEvent)

Description: Determining whether the program event meets the filtering conditions, the filtering behavior being implemented by the application itself.

Parameter: org.ngb.toolkit.epg.programEvent – An org.ngb.toolkit.epg.ProgramEvent object, indicating the program event to be filtered.

Return: boolean type, true value indicating that the program event specified by the parameter org.ngb.toolkit.epg.programEvent meets the filter condition, and false value indicating that the filter condition is not met.

H.3.3 Interface org.ngb.toolkit.epg.ProgramService

Prototype: public interface org.ngb.toolkit.epg.ProgramService

Description: Program service information, indicating a package of a group of program events (org.ngb.toolkit.epg.ProgramEvent) belonging to the same service. The program events included in the program service belong to the same service, and different program events are included due to

different query conditions. Program event information can be obtained through the methods provided by the org.ngb.toolkit.epg.ProgramService interface.

H.3.3.1 Constant field – service type

H.3.3.1.1 SERVICE_TYPE_RESERVED

Prototype: public static final short SERVICE_TYPE_RESERVED = 0

Description: service type – Reserved for use.

H.3.3.1.2 SERVICE_TYPE_DIGITAL_TELEVISION

Prototype: public static final short SERVICE_TYPE_DIGITAL_TELEVISION = 1

Description: service type – Digital TV Broadcasting Service.

H.3.3.1.3 SERVICE_TYPE_DIGITAL_RADIO_SOUND

Prototype: public static final short SERVICE_TYPE_DIGITAL_RADIO_SOUND = 2

Description: service type – Digital sound broadcasting service.

H.3.3.1.4 SERVICE_TYPE_TELETEXT

Prototype: public static final short SERVICE_TYPE_TELETEXT = 3

Description: service type – Teletext service.

H.3.3.1.5 SERVICE_TYPE_NVOD_REFERENCE

Prototype: public static final short SERVICE_TYPE_NVOD_REFERENCE = 4

Description: service type – NVOD reference business.

H.3.3.1.6 SERVICE_TYPE_NVOD_TIME_SHIFTED

Prototype: public static final short SERVICE_TYPE_NVOD_TIME_SHIFTED = 5

Description: service type – NVOD time shift service.

H.3.3.1.7 SERVICE_TYPE_MOSAIC

Prototype: public static final short SERVICE_TYPE_MOSAIC = 6

Description: service type – Mosaic service.

H.3.3.1.8 SERVICE_TYPE_PAL

Prototype: public static final short SERVICE_TYPE_PAL = 7

Description: service type – PAL encoded signal.

H.3.3.1.9 SERVICE_TYPE_SECAM

Prototype: public static final short SERVICE_TYPE_SECAM = 8

Description: service type – SECAM encoded signal.

H.3.3.1.10 SERVICE_TYPE_D_D2_MAC

Prototype: public static final short SERVICE_TYPE_D_D2_MAC = 9

Description: service type – D/D2-MAC.

H.3.3.1.11 SERVICE_TYPE__FM_RADIO

Prototype: public static final short SERVICE_TYPE_FM_RADIO = 10

Description: service type – FM wireless.

H.3.3.1.12 SERVICE_TYPE__NTSC

Prototype: public static final short SERVICE_TYPE_NTSC = 11

Description: service type – NTSC encoded signal.

H.3.3.1.13 SERVICE_TYPE_DATA_BROADCAST

Prototype: public static final short SERVICE_TYPE_DATA_BROADCAST = 12

Description: service type – Data broadcasting service.

H.3.3.1.14 SERVICE_TYPE_RESERVED_FOR_CI

Prototype: public static final short SERVICE_TYPE_RESERVED_FOR_CI = 13

Description: service type – reserved for Common Interface Usage.

H.3.3.1.15 SERVICE_TYPE_RCS_MAP

Prototype: public static final short SERVICE_TYPE_RCS_MAP = 14

Description: service type – RCS Map.

H.3.3.1.16 SERVICE_TYPE_RCS_FLS

Prototype: public static final short SERVICE_TYPE_RCS_FLS = 15

Description: service type – RCS FLS.

H.3.3.1.17 SERVICE_TYPE_DVB_MHP

Prototype: public static final short SERVICE_TYPE_DVB_MHP = 16

Description: service type – DVB MHP service.

H.3.3.2 Method

H.3.3.2.1 getDvbLocator

Prototype: public org.davic.net.dvb.DvbNetworkBoundLocator getDvbLocator()

Description: Getting the locator of the service corresponding to the program service object.

Parameter: None.

Return: An org.davic.net.dvb.DvbNetworkBoundLocator object, indicating a service locator.

H.3.3.2.2 getNetworkID

Prototype: public int getNetworkID()

Description: Getting the network ID of the network to which the program service object belongs.

Parameter: None.

Return: Int type, indicating the network ID of the network to which the program service object belongs.

H.3.3.2.3 getServiceName

Prototype: public java.lang.String getServiceName()

Description: Getting the name of the service corresponding to the program service object.

Parameter: None.

Return: A java.lang.String object, indicating the service name.

H.3.3.2.4 getServiceLogicNumber

Prototype: public int getServiceLogicNumber()

Description: Getting the logical channel number of the service corresponding to the program service object.

NOTE – The method of getting the logical channel number is determined by the system itself.

Parameter: None.

Return: Int type, indicating the service logic channel number.

H.3.3.2.5 getServiceType

Prototype: public int getServiceType()

Description: Getting the type of service corresponding to the program service object.

Parameter: None.

Return: Int type, indicating a service type. For the value, please refer to the constant field definition "Service Type" of the org.ngb.toolkit.epg.ProgramService interface.

H.3.3.2.6 getCAFreeMode

Prototype: public boolean getCAFreeMode()

Description: Getting whether the service corresponding to the program service object is scrambled.

Parameter: None.

Return: boolean type, indicating whether the service is scrambled, true value indicating scrambled, and false value indicating not scrambled.

H.3.3.2.7 getPresentProgramEvent

Prototype: public org.ngb.toolkit.epg.ProgramEvent getPresentProgramEvent()

Description: Getting a current program event of the service corresponding to the program service object.

Return: An org.ngb.toolkit.epg.ProgramEvent object, indicating the current program event.

H.3.3.2.8 getFollowingProgramEvent

Prototype: public ProgramEvent getFollowingProgramEvent()

Description: Getting a subsequent program event of the service corresponding to the program service object.

Return: An org.ngb.toolkit.epg.ProgramEvent object, indicating the subsequent program event.

H.3.3.2.9 getProgramEvents

Prototype: public org.ngb.toolkit.epg.ProgramEvent[] getProgramEvents

(ProgramEventFilter filter)

Description: Getting the program event (org.ngb.toolkit.epg.ProgramEvent) object contained in the program service object according to the specified filter.

Parameter: filter – An org.ngb.toolkit.epg.ProgramEventFilter object, indicating the program event filter. If the parameter is null, it means to get all program events.

Return: An org.ngb.toolkit.epg.ProgramEvent object array, indicating the program events contained in the program service object. If there is no program event object that meets the filter conditions, the length of the returned array is 0.

H.3.3.2.10 getEPGManager

Prototype: public org.ngb.toolkit.epg.EPGManager getEPGManger()

Description: Getting the org.ngb.toolkit.epg.EPGManager instance to which the program service object belongs.

Parameter: None.

Return: An org.ngb.toolkit.epg.EPGManager object, indicating the org.ngb.toolkit.epg.EPGManager instance to which the program service object belongs.

H.3.4 Interface org.ngb.toolkit.epg.ProgramServiceFilter

Prototype: public interface org.ngb.toolkit.epg.ProgramServiceFilter

Description: An EPG program service query filter interface, implemented by the application layer.

H.3.4.1 Method

H.3.4.1.1 accept

Prototype: public boolean accept(org.ngb.toolkit.epg.ProgramService programService)

Description: Determining whether the program service meets the filter conditions, the filtering behavior being implemented by the application.

Parameter: An org.ngb.toolkit.epg.programService – An org.ngb.toolkit.epg.ProgramService object, indicating the program service object to be filtered.

Return: A boolean type, indicating the determination result, true value indicating that the filter condition is met, and false indicating that the filter condition is not met.

H.3.5 Interface org.ngb.toolkit.epg.EPGUpdateListener

Prototype: public interface org.ngb.toolkit.epg.EPGUpdateListener extends java.util.EventListener

Description: EPG information update event listener, implemented by the application program.

H.3.5.1 Method

H.3.5.1.1 onUpdate

Prototype: public void onUpdate(org.ngb.toolkit.epg.EPGUpdateEvent event)

Description: EPG information update processing method.

Parameter: event – An org.ngb.toolkit.epg.EPGUpdateEvent object, indicating EPG information update event.

Return: None.

H.3.6 Class org.ngb.toolkit.epg.EPGManager

Prototype: public class org.ngb.toolkit.epg.EPGManager

Description: EPG Information Manager, providing a method to obtain EPG information, and being the entry class of the EPG unit.

H.3.6.1 Method

H.3.6.1.1 getEPGManager

Prototype: public static org.ngb.toolkit.epg.EPGManager[] getEPGManager()

Description: Getting an instance of the EPG manager. The receiving terminal may have multiple broadcast network interfaces (such as simultaneous connection to a wired network and a national

standard terrestrial wireless network), and each network interface corresponds to an EPG manager instance. If there are multiple networks, this method will return multiple EPG manager instances.

Return: An EPGManager object array, indicating an instance of the EPG manager. In a system with only one network interface, the length of the array is 1.

NOTE – When implementing the system, the following special scenarios need to be considered:

– Scenario 1 – The same physical network interface is connected to multiple broadcast networks. For example, the ground wireless Tuner may be connected to the wireless networks of different operators. In this scenario, there is only one instance of org.ngb.toolkit.epg.EPGManager;

– Scenario 2 – Multiple physical network interfaces are connected to the same broadcast network. For example, a receiver with PVR function has two or more Tuners connected to the same network at the same time. In this scenario, it corresponds to multiple EPGManager instances.

H.3.6.1.2 getNetworkIDs

Prototype: public java.lang.Integer[] getNetworkIDs()

Description: Getting the network IDs of the networks corresponding to this org.ngb.toolkit.epg.EPGManager instance.

Parameter: None.

Return: An integer object array, indicating network IDs of the networks corresponding to this org.ngb.toolkit.epg.EPGManager instance.

NOTE – In the scenario where the same physical network interface is connected to multiple broadcast networks, an array will be returned.

H.3.6.1.3 getNetworkName

Prototype: public java.lang.String getNetworkName(int networkId)

Description: Getting the name of the network corresponding to this EPGManager instance specified by the parameter networkId.

Parameter: networkId – Int type, indicating the network identifier.

Return: A java.lang.String object, indicating the name of the network corresponding to the org.ngb.toolkit.epg.EPGManager instance specified by the parameter networkId.

H.3.6.1.4 addUpdateListener

Prototype: public void addUpdateListener(org.ngb.toolkit.epg.EPGUpdateListener listener)

Description: Register an EPG update event listener.

Parameter: listener – An org.ngb.toolkit.epg.EPGUpdateListener object, indicating the EPG update event listener to be registered.

Return: None.

H.3.6.1.5 removeUpdateListener

Prototype: public void removeUpdateListener(org.ngb.toolkit.epg.EPGUpdateListener listener)

Description: Unregister an EPG update event listener.

Parameter: listener – An org.ngb.toolkit.epg.EPGUpdateListener object, indicating the EPG update event listener to be unregistered.

Return: None.

H.3.6.1.6 getService

Prototype: public org.ngb.toolkit.epg.ProgramService getService(int networkId, int logicNumber)

throw Java.security.InvalidParameterException

Description: Getting the program service (org.ngb.toolkit.epg.ProgramService) object according to the service logic channel number.

Parameter: networkId – Int type, indicating the network identifier of the network to which the broadcast service belongs;

logicNumber – Int type, indicating the logical channel number of the broadcast service.

Return: An org.ngb.toolkit.epg.ProgramService object, indicating a program service.

Exception: java.security.InvalidParameterException – If the org.ngb.toolkit.epg.ProgramService object specified by the parameters networkId and logicNumber does not exist, this exception is thrown.

H.3.6.1.7 getService

Prototype: public org.ngb.toolkit.epg.ProgramService getService

(org.davic.net.dvb.DvbNetworkBoundLocator serviceLocator)

throw java.security.InvalidParameterException

Description: Getting the program service (org.ngb.toolkit.epg.ProgramService) object according to the DVB locator.

Parameter: serviceLocator – An org.davic.net.dvb.DvbNetworkBoundLocator object, indicating a service locator.

Return: An org.ngb.toolkit.epg.ProgramService object, indicating a program service.

Exception: A java.security.InvalidParameterException –If the parameter serviceLocator is invalid, this exception is thrown.

H.3.6.1.8 getServices

Prototype: public org.ngb.toolkit.epg.ProgramService[] getServices(ProgramServiceFilter filter)

throw java.security.InvalidParameterException

Description: Getting the program service (org.ngb.toolkit.epg.ProgramService) object according to the specified filter.

Parameter: filter – An org.ngb.toolkit.epg.ProgramServiceFilter object, indicating the program service filter.

Return: An org.ngb.toolkit.epg.ProgramService object array, indicating the program service that meets the filter conditions. If there is no org.ngb.toolkit.epg.ProgramService object that meets the filter conditions, the length of the returned array is 0.

Exception: A java.security.InvalidParameterException –If the parameter filter is null, this exception is thrown.

H.3.6.1.9 getPresentFollowingEvent

Prototype: public ProgramService getPresentFollowingEvent(DvbNetworkBoundLocator locator)

throws java.security.InvalidParameterException

Description: A Synchronous method, getting PF information of the specified service, do not search Schedule. The return value is org.ngb.toolkit.epg.ProgramServic object, and the NVOD reference service may have multiple PFs. The method of the object org.ngb.toolkit.epg.getProgramEvents should return the PF information of the reference service in chronological order, namely: P[0]F[0]P[1]F[1]… P[n]F[n].

– If EPG PF information that meets the conditions is successfully found, the org.ngb.toolkit.epg.ProgramService object will be returned. Through the methods of the org.ngb.toolkit.epg.ProgramService object, the program events (org.ngb.toolkit. epg.ProgramEvent) object can be further obtained;
– If the service specified by the parameter locator is in the current transport stream associated with this EPGManager instance, although whether the program event is obtained from the stream or the cache is determined by the system itself, it should be ensured that the program event obtained by this interface is the latest;
– If the service specified by the parameter locator is not in the current transport stream associated with this EPGManager instance, the program event should be obtained from the cache.

Parameter: locator – An org.davic.net.dvb.DvbNetworkBoundLocator object, indicating the locator of the service;

Return: org.ngb.toolkit.epg.ProgramService object.

Exception: A java.security.InvalidParameterException–If the locator parameter cannot locate a service, this exception is thrown.

H.3.6.1.10 getProgramService

Prototype: public org.ngb.toolkit.epg.ProgramService getProgramService

(org.davic.net.dvb.DvbNetworkBoundLocator locator,

java.util.Date beginDate, java.util.Date endDate)

throws java.security.InvalidParameterException, InvalidPeriodException

Description: A Synchronous method, getting the program service (org.ngb.toolkit.epg.ProgramService) object. The start and end time of the program event contained in the program service object is determined by the parameters beginDate and endDate.

– If EPG information that meets the conditions is successfully found, the org.ngb.toolkit.epg.ProgramService object will be returned. Through the methods of the org.ngb.toolkit.epg.ProgramService object, the program event (org.ngb.toolkit.epg.ProgramEvent) object can be further obtained;
– If the service specified by the parameter locator is in the current transport stream associated with this org.ngb.toolkit.epg.EPGManager instance, although whether the program event is obtained from the stream or the cache is determined by the system itself, it should be ensured that the program event obtained by this interface is the latest;
– If the service specified by the parameter locator is not in the current transport stream associated with this org.ngb.toolkit.epg.EPGManager instance, the program event should be obtained from the cache.

Parameter: locator – An org.davic.net.dvb.DvbNetworkBoundLocator object, indicating the locator of the service;

beginDate – A java.util.Date object, indicating the start time of the program event contained in the program service object;

endDate – A java.util.Date object, indicating the end time of the program event contained in the program service object.

Return: org.ngb.toolkit.epg.ProgramService object.

Exception: A java.security.InvalidParameterException– If the locator parameter cannot locate a service, this exception is thrown.

An org.ngb.broadcast.dvb.si.InvalidPeriodException – If the specified start and end time is invalid, this exception is thrown.

H.3.6.1.11 getProgramService

Prototype: public org.ngb.toolkit.epg.ProgramService getProgramService(

org.davic.net.dvb.DvbNetworkBoundLocator locator,java.util.Date beginDate, int count, boolean isForward)

throws java.security.InvalidParameterException

Description: A Synchronous method, getting the program service (org.ngb.toolkit.epg.ProgramService) object. The start and end time of the program event included in the program service object is determined by the parameters beginDate, count and isForward.

– If EPG information that meets the conditions is successfully found, the org.ngb.toolkit.epg.ProgramService object will be returned. Through the methods of the org.ngb.toolkit.epg.ProgramService object, the program event (org.ngb.toolkit.epg.ProgramEvent) object can be further obtained;
– If the service specified is in the current transport stream associated with this org.ngb.toolkit.epg.EPGManager instance, although whether the program event is obtained from the stream or the cache is determined by the implementer itself, it should be ensured that the program event obtained by this interface is the latest;
– If the service specified is not in the current transport stream associated with this org.ngb.toolkit.epg.EPGManager instance, the program event should be obtained from the cache.

Parameter: locator – An org.davic.net.dvb.DvbNetworkBoundLocator object, indicating the locator of the designated business;

beginDate – A java.util.Date object, indicating the start time of the program event;

count – Int type, indicating the number of program events to be searched;

isForward – boolean type, true value indicating that the program events are obtained backwards from the specified time; false indicating that the program events are obtained forwards from the specified time.

Return: org.ngb.toolkit.epg.ProgramService object.

Exception: A java.security.InvalidParameterException–If the locator parameter cannot locate a service, this exception is thrown.

H.3.6.1.12 getProgramServices

Prototype: public org.ngb.toolkit.epg.ProgramService getProgramServices

(org.ngb.toolkit.epg.ProgramServiceFilter filter)

Description: A Synchronous method, getting the program service object according to the filter. The program service object will contain all available program event objects.

– If EPG information that meets the conditions is successfully found, the org.ngb.toolkit.epg.ProgramService object array will be returned, from which the searched program event (org.ngb.toolkit.epg.ProgramEvent) object can be obtained;
– If the service specified is in the current transport stream associated with this org.ngb.toolkit.epg.EPGManager instance, although whether the program event is obtained from the stream or the cache is determined by the implementer itself, it should be ensured that the program event obtained by this interface is the latest;

– If the service specified is not in the current transport stream associated with this org.ngb.toolkit.epg.EPGManager instance, the program event should be obtained from the cache.

Parameter: filter – An org.ngb.toolkit.epg.ProgramServiceFilter object, a filter object implemented by the application.

Return: An org.ngb.toolkit.epg.ProgramService object.

H.3.6.1.13 getProgramServices

Prototype: public org.ngb.toolkit.epg.ProgramService getProgramServices

(org.ngb.toolkit.epg.ProgramServiceFilter filter,

java.util.Date beginDate, java.util.Date endDate)

throws org.ngb.broadcast.dvb.si.InvalidPeriodException

Description: An Asynchronous method, getting the program service (org.ngb.toolkit.epg.ProgramService) object according to the filter. The start and end time of the program event included in the program service object is determined by the parameters beginDate and endDate.

– If EPG information that meets the conditions is successfully found, the org.ngb.toolkit.epg.ProgramService object array will be returned, from which the searched program event (org.ngb.toolkit.epg.ProgramEvent) object can be obtained;
– If the service specified is in the current transport stream associated with this org.ngb.toolkit.epg.EPGManager instance, although whether the program event is obtained from the stream or the cache is determined by the implementer itself, it should be ensured that the program event obtained by this interface is the latest;
– If the service specified is not in the current transport stream associated with this org.ngb.toolkit.epg.EPGManager instance, the program event should be obtained from the cache.

Parameter: filter – An org.ngb.toolkit.epg.ProgramServiceFilter object, a filter object implemented by the application;

beginDate – A java.util.Date object, indicating the start time of the program event;

endDate – A java.util.Date object, indicating the end time of the program event.

Return: An org.ngb.toolkit.epg.ProgramService object.

Exception: An org.ngb.broadcast.dvb.si.InvalidPeriodException – If the specified time range is invalid, this exception is thrown.

H.3.7 Event org.ngb.toolkit.epg.EPGUpdateEvent

Prototype: public class org.ngb.toolkit.epg.EPGUpdateEvent extends java.util.EventObject

Description: An EPG information change event.

H.3.7.1 Method

H.3.7.1.1 getResult

Prototype: public org.ngb.toolkit.epg.ProgramService[] getResult()

Description: Getting the updated program service object.

Parameter: None.

Return: An org.ngb.toolkit.epg.ProgramService object array, indicating the updated program service.

H.3.7.1.2 EPGUpdateEvent

Prototype: protected EPGUpdateEvent(Object object, org.ngb.toolkit.epg.ProgramService[] result)

Description: Construction method, creating an EPGUpdateEvent object. Not exposed to the application layer.

Parameter: None.

Return: An org.ngb.toolkit.epg.EPGUpdateEvent object, indicating the org.ngb.toolkit.epg.EPGUpdateEvent instance to which the program service object belongs.

H.4 Information search module

The information search module provides classes and methods related to global search and matching search. The terms "global search" and "matching search" are defined as follows:

Global search – Search for content such as SI, PVR, etc. according to the search conditions set by the user, and return meaningful search results to improve the user experience.

Matching search – According to the user's input, a character string that can be matched in the current data source is given to shorten the time for the user to enter a querying keyword and reduce the difficulty for the user to enter the querying keyword.

The information search module is divided into the following components:

Search Manager (org.ngb.toolkit.search.SearchManager) – An Entry class of the information search module;

Global search session (org.ngb.toolkit.search.GlobalSearchSession) – A session associated with the global search process;

Matching search session (org.ngb.toolkit.search.AutoCompleteSearchSession) – A session associated with the ongoing matching search process.

The summary of information search module is shown in Table H.3.

Table H.3 – Summary of information search module

Interface	
AutoCompleteSearchListener	Matching search event listener, implemented by the application.
AutoCompleteSearchResultItem	It describes matching search results and provides methods to obtain various information about matching search results.
AutoCompleteSearchResultList	It describes a matching search result list object, and provides a method to access matching search result items.
AutoCompleteSearchSession	It describes a matching search session and provides a matching search session control method.
GlobalSearchListener	A Global search event listener, implemented by the application.
GlobalSearchResultItem	It describes a global search result object, and provides a method to access information relevant to the search result.
GlobalSearchResultList	It describes a global search result list object, and provides a method to get global search result items.
GlobalSearchSession	It describes a global search session and provides a global search session control method.
RetrieveDirection	It defines a search result search direction constant.
SearchContentType	Content type constant definition.
SearchCriteriaFlags	Filter condition tag definition.

Table H.3 – Summary of information search module

SearchFields	It provides an interface for setting search fields.
SearchHistoryItem	It describes a search history record and provides a method to get various information about the search history record.
SearchHistoryList	It describes a search history record list and provides a function of traversing the search history record list.
SearchStatus	It defines a search status constant.
SourceType	Search for data source constant definitions.
Class	
AutoCompleteSearchFilter	It describes an automatic matching search filter, provides methods for setting and obtaining matching search filter conditions.
GlobalSearchFilter	It describes a global search filter, provides methods for setting and obtaining global search filter conditions.
SearchManager	A search manager of global search and matching search, which is the entry class of the search function module.
SortCriteria	It defines a sorting constant and a sorting method.
Exception	
SearchException	It describes an exception when an error occurs in the global search.

H.4.1 Interface org.ngb.toolkit.search.AutoCompleteSearchListener

Prototype: public interface org.ngb.toolkit.search.AutoCompleteSearchListener

Description: A Matching search event listener, implemented by the application. Matching search events notified by the search engine to the application include:

– Start of search session onAutoCompleteSearchStart(AutoCompleteSearchSession, int);

– End of search session onAutoCompleteSearchStop(AutoCompleteSearchSession, int);

– Destruction of search session onAutoCompleteSearchDestroy(AutoCompleteSearchSession, int);

– Error of search session onAutoCompleteSearchError(AutoCompleteSearchSession, int).

H.4.1.1 Method

H.4.1.1.1 onAutoCompleteSearchStart

Prototype: void onAutoCompleteSearchStart

(org.ngb.toolkit.search.AutoCompleteSearchSession searchSession, int status)

Description: Call the startSearch() method of the AutoCompleteSearchSession object to complete the callback method after matching search.

Parameter: searchSession – An org.ngb.toolkit.search.AutoCompleteSearchSession object, indicating the matching search session instance that issued this event;

status – Int type, indicating the current matching search status. The value can be SearchStatus.COMPLETED.

Return: None.

H.4.1.1.2 onAutoCompleteSearchStop

Prototype: void onAutoCompleteSearchStop(org.ngb.toolkit.search.AutoCompleteSearchSession searchSession, int status)

Description: Call the stopSearch() method of the org.ngb.toolkit.search.AutoCompleteSearchSession object to stop the callback method after the matching search session.

Parameter: searchSession – An org.ngb.toolkit.search.AutoCompleteSearchSession object, indicating the matching search session instance that issued this event;

status – Int type, indicating the current matching search status, and the value can be SearchStatus.STOP_SUCCESS.

Return: None.

H.4.1.1.3 onAutoCompleteSearchDestroy

Prototype: void onAutoCompleteSearchDestroy

(org.ngb.toolkit.search.AutoCompleteSearchSession searchSession, int status)

Description: Call the dispose() method of the AutoCompleteSearchSession object to destroy the callback method after matching search session.

Parameter: searchSession – An org.ngb.toolkit.search.AutoCompleteSearchSession object, indicating the matching search session instance that issued this event;

status – Int type, indicating the current matching search status, and the value can be SearchStatus.DISPOSE_SUCCESS.

Return: None.

H.4.1.1.4 onAutoCompleteSearchError

Prototype: void onAutoCompleteSearchError

(org.ngb.toolkit.search.AutoCompleteSearchSession searchSession, int status)

Description: Callback method when there is an error in the matching search session.

Parameter: searchSession – An org.ngb.toolkit.search.AutoCompleteSearchSession object, indicating the matching search session instance that issued this event;

status – Int type, indicating the current matching search status, the possible values are:

– SearchStatus.FAILED;
– SearchStatus.TIMEOUT;
– SearchStatus.STOP_FAILED;
– SearchStatus.DISPOSE_FAILED.

Return: None.

H.4.2 Interface org.ngb.toolkit.search.AutoCompleteSearchResultItem

Prototype: public interface org.ngb.toolkit.search.AutoCompleteSearchResultItem

Description: It describes a matching search result and provides a method to obtain various information about the matching search result.

H.4.2.1 Method

H.4.2.1.1 getSource

Prototype: int getSource()

Description: Getting the data source that matches the search result.

Return: Int type, indicating the data source that matches the search result. For the value, please refer to the constant field definition of "Search Data Source" of org.ngb.toolkit.search.SourceType interface.

H.4.2.1.2 getString

Prototype: java.lang.String getString()

Description: Getting a character string that matches the keyword entered by the user.

Parameter: None.

Return: A java.lang.String object, indicating the character string that matches the keyword entered by the user.

H.4.3 Interface org.ngb.toolkit.search.AutoCompleteSearchResultList

Prototype: public interface org.ngb.toolkit.search.AutoCompleteSearchResultList extends java.util.ListIterator

Description: It describes a matching search result list object, and provides a method to access the matching search result items.

NOTE – AutoCompleteSearchResultList is a collection of autocompletesearchresultitem objects.

H.4.4 Interface org.ngb.toolkit.search.AutoCompleteSearchSession

Prototype: public interface org.ngb.toolkit.search.AutoCompleteSearchSession

Description: A Matching search session interface, providing matching search session control method.

– The application calls the startsearch () method to start the matching search session;
– The application calls the stopsearch () method to stop matching search session;
– The application calls dispose () method to destroy the matching search session;
– The application calls the getsearchresultlist () method to get the matching search results;
– The search engine calls the callback method provided by the org.ngb.toolkit.search.autocompletesearchlistener object to inform the application of matching search status.

```
// Application gets the AutoCompleteSearchFilter object and fill it with default parameters.
AutoCompleteSearchFilter acFilter = AutoCompleteSearchFilter.getAutoCompleteSearchFilter();
acFilter.setMaxResults(10);
acFilter.setSource(SourceType.ALL);
acFilter.setTimeLimit(250);
acFilter.setSearchField(SearchFields.TITLE);
acFilter.setSearchLanguage("zho");   //Search Chinese
SearchManager searchManager = SearchManager.getInstance();
// Get the auto complete search object
AutoCompleteSearchSession acSearch = searchManager.getAutoCompleteSearchSession(acFilter, this);
// User entered some characters and waits for the Auto complete List to be displayed to them.
// EPG start the auto complete search
acSearch.startSearch(searchString);
// Engine start the search with middleware and then wait for notification when search is success.
// Once available, Engine notifies EPG with the search status.
// listener.onSearchStart(acSearch, SearchStatus.COMPLETED);
```

```
// EPG can now fetch the auto complete result list.
AutoCompleteSearchResultList acList = acSearch.getSearchResultList();
// The list is an iteration and hence EPG can iterate through the list fetching the elements.
// If at any instant of time, if EPG want to stop the on going search
// may be because user has entered another character before the previous search
// is completed or may be middleware is taking more time for search
acSearch.stopSearch();
// When the auto complete search object is no more in use, then
// Application should dispose() the search object to make sure that the search
// session is destroyed and all resources are freed properly.
acSearch.dispose();
```

H.4.4.1 Method

H.4.4.1.1 startSearch

Prototype: void startSearch(java.lang.String searchStr)

throws org.ngb.toolkit.search.SearchException, java.lang.IllegalArgumentException

Description: Start a new matching search request. Once the search is completed, the search engine will inform the application to get the search results. After starting the search, the application needs to wait for the searchstatus.completed notification from the engine. Only after receiving the notification can the application successfully obtain the results.

NOTE:

– If the timeout of autocompletesearchfilter is set to 0, the search will not stop until the stopsearch() method is explicitly called.

– Get the autocompletesearchresultlist object by calling getsearchresultlist() method, and terminate the previous search request by calling stopsearch() method. If the matching search session is no longer used, call dispose() method to close the search session and release resources.

– Only one matching search session can be active at the same time. During this period, the startsearch() and stopsearch() methods can be called repeatedly. If the matching search session is no longer needed, the dispose() method should be called to close the search session and release resources.

– The search results are stored in the same cache. Once a new search is started, the old autocompletesearchresultlist object can no longer be used to get the previous search results.

Parameter: searchStr – A java.lang.String object, indicating a character string entered by the user.

Return: None.

Exception: org.ngb.toolkit.search.SearchException – If the start session request operation fails, this exception is thrown;

java.lang.IllegalArgumentException – If the input parameter is illegal, this exception is thrown.

H.4.4.1.2 stopSearch

Prototype: void stopSearch() throws org.ngb.toolkit.search.SearchException

Description: Terminate the matching search request that was started before.

NOTE – This method only stops the matching search, however, the matching search session is still valid, and the corresponding resources are not released. This method should be called forcedly in the following scenarios:

– If the matching search session has started and the result is waiting to be returned, and the user has entered another character before the search engine returns the result, the stopsearch() method must be called to terminate the search;
– The timeout of the matching search filter is set to 0, which will wait for the search results. If there is no corresponding match results, the stopsearch() method must be called to terminate the search before starting another search.

Tip: if the matching search has been completed, you do not need to call this method.

Parameter: None.

Return: None.

Exception: org.ngb.toolkit.search.SearchException – This exception is thrown if the matching search is not started or if the stop matching search fails.

H.4.4.1.3 dispose

Prototype: void dispose() throws org.ngb.toolkit.search.SearchException

Description: Destroy matching search session objects that are no longer needed.

NOTE – After the matching search, once the session is no longer needed, the method must be called to release corresponding resources.

Parameter: None.

Return: None.

Exception: org.ngb.toolkit.search.SearchException – If the matching search is not started or the destroy matching search session fails, this exception is thrown.

H.4.4.1.4 getSearchResultList

Prototype: AutoCompleteSearchResultList getSearchResultList()

throws org.ngb.toolkit.search.SearchException

Description: Getting the results of the matching search.

NOTE – The autocompletesearchlistener object can only call this method after receiving the notification of SearchStatus.COMPLETED. The search results are stored in the same cache. Once a new matching search is initialized, the old AutoCompleteSearchResultList object can no longer be used to get the previous search results.

Return: An AutoCompleteSearchResultList object, indicating a list of matching search results.

Exception: org.ngb.toolkit.search.SearchException – If the org.ngb.toolkit.search.AutoCompleteSearchListener object does not receive the notification of SearchStatus.COMPLETED (that is, when the matching search is not completed), the method is called, and this exception is thrown.

H.4.5 Interface org.ngb.toolkit.search.GlobalSearchListener

Prototype: public interface org.ngb.toolkit.search.GlobalSearchListener

Description: Global search event listener, implemented by application. The matching search events notified by the search engine to the application include:

– Start of search session onGlobalSearchStart(GlobalSearchSession, int);
– End of search session onGlobalSearchStop(GlobalSearchSession, int);
– Close of search session onGlobalSearchDestroy(GlobalSearchSession, int);
– Error of search session onGlobalSearchError(GlobalSearchSession, int);
– Get local data onGlobalSearchRetrieval(GlobalSearchSession, int).

H.4.5.1 Method

H.4.5.1.1 onGlobalSearchStart

Prototype: void onGlobalSearchStart(org.ngb.toolkit.search.GlobalSearchSession searchSession, int status)

Description: Call startsearch() method of org.ngb.toolkit.search.GlobalSearchSession object to complete the callback method after global search.

Parameter: searchSession – An org.ngb.toolkit.search.GlobalSearchSession object, indicating the global search instance that issued this event;

status – Int type, indicating the current global search status, the value can be:

– SearchStatus.COMPLETED;

– SearchStatus.IN_PROGRESS;

– SearchStatus.INITIATED.

Return: None.

H.4.5.1.2 onGlobalSearchStop

Prototype: void onGlobalSearchStop(org.ngb.toolkit.search.GlobalSearchSession searchSession, int status)

Description: Call stopSearch() method of org.ngb.toolkit.search.globalsearchsession object to stop the callback method after global search.

Parameter: searchSession – An org.ngb.toolkit.search.GlobalSearchSession object, indicating the global search session instance that issued this event;

status – Int type, indicating the current global search status, the value can be SearchStatus.STOP_SUCCESS.

Return: None.

H.4.5.1.3 onGlobalSearchDestroy

Prototype: void onGlobalSearchDestroy(org.ngb.toolkit.search.GlobalSearchSession searchSession, int status)

Description: Call the dispose () method of the GlobalSearchSession object to destroy the callback method after the global search session.

Parameter: searchSession – An org.ngb.toolkit.search.GlobalSearchSession object, indicating the global search session instance that issued this event;

status – Int type, indicating the current global search status, the value can be SearchStatus.DISPOSE_SUCCESS.

Return: None.

H.4.5.1.4 onGlobalSearchError

Prototype: void onGlobalSearchError(org.ngb.toolkit.search.GlobalSearchSession searchSession, int status)

Description: Callback method for global search session error.

Parameter: searchSession – An org.ngb.toolkit.search.GlobalSearchSession object, indicating the global search session instance that issued this event;

status – Int type, indicating the global search status, the value can be:

– SearchStatus.FAILED;

– SearchStatus.INTERRUPTED;
– SearchStatus.TIMEOUT_STOP_FAILED;
– SearchStatus.TIMEOUT;
– SearchStatus.DISPOSE_FAILED;
– SearchStatus.RETRIEVAL_FAILED;
– SearchStatus.RETRIEVAL_INSUFFICIENT;
– SearchStatus.STOP_FAILED.

H.4.5.1.5 onGlobalSearchRetrieval

Prototype: void onGlobalSearchRetrieval(org.ngb.toolkit.search.GlobalSearchSession searchSession, int status)

Description: Callback method for global search to obtain partial search results.

Parameter: searchSession – An org.ngb.toolkit.search.GlobalSearchSession object, indicating the global search session instance that issued this event;

status – Int type, indicating the global search status, the value can be SearchStatus.RETRIEVAL_SUCCESS.

Return: None.

H.4.6 Interface org.ngb.toolkit.search.GlobalSearchResultItem

Prototype: public interface org.ngb.toolkit.search.GlobalSearchResultItem

Description: It describes a global search result and provides a method to access the information related to the search result.

H.4.6.1 Method

H.4.6.1.1 getContent

Prototype: java.lang.Object getContent()

Description: Getting the object associated with this search result.

Parameter: None.

Return: java.lang.Object object, the object may be of SIEvent type.

The application should further determine the type of the returned object through instanceof method.

H.4.7 Interface org.ngb.toolkit.search.GlobalSearchResultList

Prototype: public interface org.ngb.toolkit.search.GlobalSearchResultList extends java.util.ListIterator

Description: It describes the global search result list object, and provides a method to obtain global search result items.

NOTE – The org.ngb.toolkit.search.GlobalSearchResultList is a collection of org.ngb.toolkit.search.GlobalSearchResultItem.

H.4.8 Interface org.ngb.toolkit.search.GlobalSearchSession

Prototype: public interface org.ngb.toolkit.search.GlobalSearchSession

Description: Global search session interface, providing global search session control method.
– The application calls the startSearch () method to start the global search session;
– The application calls stopSearch () method to stop the global search session;

– The application calls dispose () method to destroy the global search session and release resources;
– The application calls retrievePage () method to navigate among the first page, the next page and the previous page;
– The application calls the getSearchResultList () method to obtain the search results;
– The search engine calls the method provided by the org.ngb.toolkit.search.GlobalSearchListener object to inform the application of the global search status.

NOTE – The application calls setPageSize() to set the page size. If the page size is not set, the default value DEFAULT_PAGE_SIZE will be used.

```
// If application want to filter the search results by providing advanced criteria,
 // like if they want to filter on broadcast events, which will be broadcasted
 // between 10 AM and 10 PM,
 // with genre ALL, with parental rating as 13 and star rating between 2-5
 // Application gets the GlobalSearchFilter object and fill it with required
 // parameters.
GlobalSearchFilter gsFilter = GlobalSearchFilter.getGlobalSearchFilter();
 gsFilter.setSource(BROADCAST);
 gsFilter.setMaxResults(5);
// Fills the advanced criteria
 gsFilter.setCategory(MAIN_CATEGORY_ALL, SUB_CATEGORY_ALL);
gsFilter.setTimeLimit(1500); // 1.5 secs
 gsFilter.setSearchField(TITLE); // search only in title.
// Sort criteria
 SortCriteria sortCriteria = new SortCriteria();
 sortCriteria.setOrder(SORT_ORDER_ASCENDING);
 sortCriteria.setType(SORT_TYPE_TITLE);
SearchManager searchManager = SearchManager.getInstance();
 // Get the global search object.
 GlobalSearchSession gsSearch = searchManager.getGlobalSearchSession(gsFilter, sortCriteria, this);
// start the global search
 gsSearch.startSearch(searchString);
// Engine start the search with middleware and then wait for notification when
 // search is success.
 // Once available, Engine notifies EPG with the search status.
 // listener.onGlobalSearchStart(gsSearch, SearchStatus.INITIATED);
 // Application can update the UI indicating the initiated status.
// Engine notifies the in progress notification.
```

```
 // listener.onGlobalSearchStart(gsSearch, SearchStatus.IN_PROGRESS);
// EPG can now fetch the call the retrieve API to fetch the results.
 gsSearch.setPageSize(6);
 gsSearch.retrievePage(FIRST_PAGE);
// Engine notifies the retrieve status.
 //listener.onGlobalSearchRetrieve(gsSearch, SearchStatus.RETRIEVAL_SUCCESS);
// EPG can now display the result list.
 GlobalSearchResultList gsList = gsSearch.getSearchResultList();
// The list is an iteration and hence EPG can iterate through the list fetching
 // the elements.
// If at any instant of time, if EPG want to stop the on going search
 // may be because user has pressed back key or explicitly stopped.
 gsSearch.stopSearch();
// When the global search object is no more in use, then
 // Application should dispose() the search object to make sure that the search
 // session is destroyed and all resources are freed properly.
 gsSearch.dispose();
```

H.4.8.1 Constant field – size of default service

H.4.8.1.1 DEFAULT_PAGE_SIZE

Prototype: public static final int DEFAULT_PAGE_SIZE = 6

Description: The default page size of global search results is the number of search result items contained in each page.

H.4.8.2 Method

H.4.8.2.1 startSearch

Prototype: void startSearch(java.lang.String searchStr)

throws org.ngb.toolkit.search.SearchException, java.lang.IllegalArgumentException

Description: Start a new global search request. Once the search is completed, the search engine will notify the application to get the search results. After starting the search, the application needs to wait for the notification from the search engine. Only after receiving the notification, the application can successfully obtain the results.

NOTE:

– If the timeout of GlobalSearchFilter is set to 0, the search will stop only when the application explicitly calls the stopSearch() method.

– Get the GlobalSearchResultList object by calling the getSearchResultList() method, and terminate the previous search request that is still in progress by calling the stopSearch() method. If the global search session is no longer used, you should call the dispose() method to release it.

– Only one search session is active at the same time. The search results are stored in the same cache. Once a new search is started, the old GlobalSearchResultList object can no longer be used to obtain the previous search results.

Parameter: searchStr – A java.lang.String object, indicating the character string entered by the user.

Return: None.

Exception: org.ngb.toolkit.search.SearchException – If the operation of starting session request fails, this exception is thrown;

java.lang.IllegalArgumentException – If the input parameter is illegal, this exception is thrown.

H.4.8.2.2 stopSearch

Prototype: void stopSearch() throws org.ngb.toolkit.search.SearchException

Description: Terminate the global search request that has been started before.

NOTE – This operation only terminates the global search, however, the global search session is still valid, and the corresponding resources are not released.

Parameter: None.

Return: None.

Exception: org.ngb.toolkit.search.SearchException – If the search has not been started or the search fails to stop, this exception is thrown.

H.4.8.2.3 dispose

Prototype: void dispose() throws org.ngb.toolkit.search.SearchException

Description: Destroy global search session objects that are no longer needed.

NOTE – After the global search is completed, once the session is no longer needed, this method must be called to release the corresponding resources.

Parameter: None.

Return: None.

Exception: org.ngb.toolkit.search.SearchException – If the search is not started or the destroy operation fails, this exception is thrown.

H.4.8.2.4 getPageSize

Prototype: int getPageSize()

Description: Getting the number of items of the search results returned per page.

Parameter: None.

Return: Int type, indicating the page size currently set.

H.4.8.2.5 getResultCount

Prototype: int getResultCount() throws org.ngb.toolkit.search.SearchException

Description: Getting the number of elements of the search result. If the search has not been completed and is partially updated, the number of results will also be updated according to the change of the search element. For each threshold limit, the application will receive a notification of SearchStatus.IN_PROGRESS.

Parameter: None.

Return: Int type, indicating the number of elements of the search result.

Exception: org.ngb.toolkit.search.SearchException – If the search has not started or has been terminated, this exception is thrown.

H.4.8.2.6 getSearchResultList

Prototype: org.ngb.toolkit.search.GlobalSearchResultList getSearchResultList()

throws org.ngb.toolkit.search.SearchException

Description: Getting global search results.

NOTE – Before calling this method, the application needs to wait for notification. The list returned by this method only contains the main content list. The results are stored in the same cache. Once a new search is initialized, the old GlobalSearchResultList object can no longer be used to obtain the previous search results.

Return: An org.ngb.toolkit.search.GlobalSearchResultList object, indicating a list of global search results.

Exception: org.ngb.toolkit.search.SearchException – If the search has not started or has been terminated, this exception is thrown.

H.4.8.2.7 retrievePage

Prototype: void retrievePage(int retrieveDirection)

throws org.ngb.toolkit.search.SearchException, java.lang.IllegalArgumentException

Description: Getting the data of the first page, the next page, or the previous page from the search result list.

Parameter: retrieveDirection – Int type, it is used to specify the result of obtaining the first page, the previous page or the next page. The value can be NEXT_PAGE, PREVIOUS_PAGE or FIRST_PAGE. For details, please refer to the "Search Direction" constant field definition of the RetrieveDirection interface.

Exception: org.ngb.toolkit.search.SearchException – If any failed operation occurs, this exception is thrown;

java.lang.IllegalArgumentException – If the search direction is incorrect, this exception is thrown.

H.4.8.2.8 saveRecentSearchQuery

Prototype: void saveRecentSearchQuery() throws org.ngb.toolkit.search.SearchException

Description: Saving the latest search results. This method can only be called before initializing a search or destroying the object.

Parameter: None.

Return: None.

Exception: org.ngb.toolkit.search.SearchException – If the operation fails, this exception is thrown.

H.4.8.2.9 setPageSize

Prototype: void setPageSize(int pageSize) throws java.lang.IllegalArgumentException

Description: Setting the number of items of the search results returned per page.

If this method is not called to set the page size, the default page size is DEFAULT_PAGE_SIZE.

Parameter: pageSize – Int type, indicating the number of results returned per page.

Exception: java.lang.IllegalArgumentException – If the parameter is less than or equal to 0 or exceeds the maximum value supported by the system, this exception is thrown.

H.4.9 Interface org.ngb.toolkit.search.RetrieveDirection

Prototype: public interface org.ngb.toolkit.search.RetrieveDirection

Description: It defines find direction constant of the search result.

H.4.9.1 Constant field – find direction

H.4.9.1.1 FIRST_PAGE

Prototype: public static final int FIRST_PAGE = 0

Description: Find direction – first page.

H.4.9.1.2 NEXT_PAGE

Prototype: public static final int NEXT_PAGE = 1

Description: Find direction – next page.

H.4.9.1.3 PREVIOUS_PAGE

Prototype: public static final int PREVIOUS_PAGE = 2

Description: Find direction – previous page.

H.4.10 Interface org.ngb.toolkit.search.SearchContentType

Prototype: public interface org.ngb.toolkit.search.SearchContentType

Description: Content type constant definition.

H.4.10.1 Constant field – content type

H.4.10.1.1 ALL

Prototype: public static final int ALL = 0

Description: Content type – Audio and video.

H.4.10.1.2 AUDIO_ONLY

Prototype: public static final int AUDIO_ONLY = 1

Description: Content type – Audio.

H.4.10.1.3 VIDEO_ONLY

Prototype: public static final int VIDEO_ONLY = 2

Description: Content type – Video.

H.4.11 Interface org.ngb.toolkit.search.SearchCriteriaFlags

Prototype: public interface org.ngb.toolkit.search.SearchCriteriaFlags

Description: Filter condition tag definition.

H.4.11.1 Constant field – filter condition

H.4.11.1.1 FLAG_NONE

Prototype: public static final int FLAG_NONE = 0

Description: Filter condition flag – none.

H.4.11.1.2 FLAG_SD_EVENT

Prototype: public static final int FLAG_SD_EVENT = 1

Description: Filter condition flag – Filter standard definition (2D) content.

H.4.11.1.3 FLAG_HD_EVENT

Prototype: public static final int FLAG_HD_EVENT = 2

Description: Filter condition flag – Filter high definition (2D) content.

H.4.11.1.4 FLAG_3D_CONTENT

Prototype: public static final int FLAG_3D_CONTENT = 4

Description: Filter condition flag – Filter 3D content.

H.4.11.1.5 FLAG_CLEAR

Prototype: public static final int FLAG_CLEAR = 32

Description: Filter condition flag – Filter unscrambled content.

H.4.11.1.6 FLAG_SCRAMBLED

Prototype: public static final int FLAG_SCRAMBLED = 64

Description: Filter condition flag – Filter scrambled content.

H.4.12 Interface org.ngb.toolkit.search.SearchFields

Prototype: public interface org.ngb.toolkit.search.SearchFields

Description: Providing an interface for setting search fields.

H.4.12.1 Constant field – search field

H.4.12.1.1 ALL_STRING_FIELDS

Prototype: public static final int ALL_STRING_FIELDS = 0

Description: Searching all information.

H.4.12.1.2 SYNOPSIS

Prototype: public static final int SYNOPSIS = 1

Description: Searching only in synopsis.

H.4.12.1.3 TITLE

Prototype: public static final int TITLE = 2

Description:Searching only in title.

H.4.13 Interface org.ngb.toolkit.search.SearchHistoryItem

Prototype: public interface org.ngb.toolkit.search.SearchHistoryItem

Description: It describes a search history record and provides methods to obtain various information about the search history record.

H.4.13.1 Method

H.4.13.1.1 getContentType

Prototype: public int getContentType()

Description: Getting the content type of the search.

Parameter: None.

Return: Int type, indicating the content type of the search. For the value, please refer to the "content type" constant field definition of the org.ngb.toolkit.search.SearchContentType interface.

H.4.13.1.2 getCriteriaFlags

Prototype: public int getCriteriaFlags()

Description: Getting a criteria flag of the search.

Parameter: None.

Return: Int type, indicating serach criteria flag. For the value, please refer to the constant field definition of "filter condition" of org.ngb.toolkit.search.SearchCriterialFlags interface.

H.4.13.1.3 getSearchField

Prototype: public int getSearchField()

Description: Getting the search field for this search.

Parameter: None.

Return: Int type, indicating the search field. For the value, see the constant field definition of "Search Source" of org.ngb.toolkit.search.SearchFields interface for details.

H.4.13.1.4 getSearchString

Prototype: public java.lang.String getSearchString()

Description: Getting the matching character string for this search.

Parameter: None.

Return: A java.lang.String object, indicating search matching character string.

H.4.13.1.5 getSortCriteria

Prototype: public SortCriteria getSortCriteria()

Description: Getting sorting information of the search results of the search, such as sorting method and type.

Parameter: None.

Return: An org.ngb.toolkit.search.SortCriteria object, indicating the sorting information of search results.

H.4.13.1.6 getSources

Prototype: public int getSources()

Description: Getting the search data source for this search.

Parameter: None.

Return: Int type, indicating the search data source. For the value, see the constant field definition of "Search Data Source" of the SourceType interface for details.

H.4.14 Interface org.ngb.toolkit.search.SearchHistoryList

Prototype: public interface SearchHistoryList extends java.util.ListIterator

Description: It describes the search history record list and provides the function of traversing the search history record list.

NOTE – SearchHistoryList is a collection of SearchHistoryItem.

H.4.15 Interface org.ngb.toolkit.search.SearchStatus

Prototype: public interface org.ngb.toolkit.search.SearchStatus

Description: A search status interface defines the search status constant.

H.4.15.1 Constant field – search status

H.4.15.1.1 INITIATED

Prototype: public static final byte INITIATED = 0

Description: search status – Initialization is completed.

NOTE – Only valid for global search.

H.4.15.1.2 IN_PROGRESS

Prototype: public static final byte IN_PROGRESS = 1

Description: Search status – Ongoing. This status means that from now on, search results (but not all) can be provided to the application.

NOTE – Only valid for global search.

H.4.15.1.3 COMPLETED

Prototype: public static final byte COMPLETED = 2

Description: Search status – end. This status indicates that the search is complete and all results can be searched.

H.4.15.1.4 INTERRUPTED

Prototype: public static final byte INTERRUPTED = 3

Description: Search status – interrupted. When the search is in progress, the application calls the stopSearch() method of the GlobalSearchSession object to stop the search, and the application will be notified.

NOTE – Only valid for global search.

H.4.15.1.5 TIMEOUT

Prototype: public static final byte TIMEOUT = 4

Description: Search status – The search automatically stops after timeout. If the result is not searched within the timeout period, the application will be notified of this status. This state also indicates that the search has stopped successfully, and the application does not need to call the stopSearch() method of the GlobalSearchSession object or the stopSearch() method of the AutoCompleteSearchSession object to explicitly stop the search.

H.4.15.1.6 TIMEOUT_STOP_FAILED

Prototype: public static final byte TIMEOUT_STOP_FAILED = 5

Description: Search status – The search fails to stop after the timeout. If the result is not searched within the timeout period, the application will be notified of this status. This state also indicates that the search was not successfully stopped, andthe application should call the stopSearch() method of the GlobalSearchSession object to explicitly stop the search.

NOTE – Only valid for global search.

H.4.15.1.7 FAILED

Prototype: public static final byte FAILED = 6

Description: Search status – The search failed. This status indicates that the start of the search failed, and no results can be returned to the application.

H.4.15.1.8 STOP_SUCCESS

Prototype: public static final byte STOP_SUCCESS = 7

Description: Search status – The search was stopped successfully.

H.4.15.1.9 STOP_FAILED

Prototype: public static final byte STOP_FAILED = 8

Description: Search status – The search failed to stop.

H.4.15.1.10 DISPOSE_SUCCESS

Prototype: public static final byte DISPOSE_SUCCESS = 9

Description: Search status – The search is closed successfully.

H.4.15.1.11 DISPOSE_FAILED

Prototype: public static final byte DISPOSE_FAILED = 10

Description: Search status – The search failed to close.

H.4.15.1.12 RETRIEVAL_SUCCESS

Prototype: public static final byte RETRIEVAL_SUCCESS = 11

Description: Search status – Search results have been successfully retrieved. This status indicates that the cache is successful, and the application can obtain the search result list through the getSearchResultList() method of the org.ngb.toolkit.search.GlobalSearchSession object.

NOTE – Only valid for global search.

H.4.15.1.13 RETRIEVAL_FAILED

Prototype: public static final byte RETRIEVAL_FAILED = 12

Description: Search status – Failed to get search results. This status indicates that the search results cannot be retrieved due to some failure reasons, and the search result list cannot be obtained through the getSearchResultList() method of the org.ngb.toolkit.search.GlobalSearchSession object.

NOTE – Only valid for global search.

H.4.15.1.14 RETRIEVAL_INSUFFICIENT

Prototype: public static final byte RETRIEVAL_INSUFFICIENT = 13

Description: Search status – Not enough search results are available. This status indicates that it cannot be retrieved because there are not enough search results, andthe list of search results cannot be obtained by calling GlobalSearchSession.getSearchResultList().

NOTE – Only valid for global search.

H.4.16 Interface org.ngb.toolkit.search.SourceType

Prototype: public interface org.ngb.toolkit.search.SourceType

Description: Search for data source constant definitions.

H.4.16.1 Constant field – search data source

H.4.16.1.1 ALL

Prototype: public static final int ALL = 0

Description: Search data source – Search from effective data sources such as SI information database and local recorded programs.

H.4.16.1.2 BROADCAST

Prototype: public static final int BROADCAST = 1

Description: Search data source – Only search SI information database.

H.4.16.1.3 RECORDED

Prototype: public static final int RECORDED = 2

Description: Search data source – Only search locally recorded/downloaded content.

H.4.17 Class org.ngb.toolkit.search.AutoCompleteSearchFilter

Prototype: public abstract class org.ngb.toolkit.search.AutoCompleteSearchFilter

Description: Matching search filter, providing matching search filter condition setting and obtaining method. The filter conditions can be:

– Search data sources;

– Search field;

– Language of text information;

– Maximum number of results returned;

– Search timeout time limit.

NOTE – The application uses AutoCompleteSearchFilter to obtain the following effects: According to the specified search data source and search field, a list of prompt character strings that match the characters input by the user are listed within the timeout time limit, and the number of character strings is less than or equal to the maximum value set by the application.

H.4.17.1 Constant field – default maximum number of search results

H.4.17.1.1 DEFAULT_MAX_AUTO_COMPLETE_SEARCH_RESULTS

Prototype: public static final int DEFAULT_MAX_AUTO_COMPLETE_SEARCH_RESULTS = 10

Description: Constant – It defines the default maximum number of search results.

This constant is the maximum value of the default search result. If the application needs to set it to another value, it can be achieved by calling the setMaxResults() method.

H.4.17.2 Method

H.4.17.2.1 getAutoCompleteSearchFilter

Prototype: public static AutoCompleteSearchFilter getAutoCompleteSearchFilter()

Description: Getting an instance of the AutoCompleteSearchFilter class implemented by the system.

Parameter: None.

Return: An AutoCompleteSearchFilter object, indicating an instance of the org.ngb.toolkit.search.AutoCompleteSearchFilter class implemented by the system.

H.4.17.2.2 getMaxResults

Prototype: public abstract int getMaxResults()

Description: Getting the maximum number of results returned by a matching search.

Parameter: None.

NOTE – If the application is not set, DEFAULT_MAX_AUTO_COMPLETE_SEARCH_RESULTS will be returned by default.

Return: Int type, indicating the maximum number of results returned by matching search.

H.4.17.2.3 getSearchField

Prototype: public abstract int getSearchField()

Description: Getting the fields that need to be found for matching search.

Parameter: None.

Return: Int type, indicating the field to be found for matching search, for the value, see the constant field definition of "Search Field" of org.ngb.toolkit.search.SearchFields interface for details.

H.4.17.2.4 getSearchLanguage

Prototype: public abstract String getSearchLanguage()

Description: Getting the language type of the text information that matches the search.

Parameter: None.

Return: A java.lang.String object, indicating the language type of the text information for matching search. The three-letter language code follows the GB/T 4880.2-2000 standard.

H.4.17.2.5 getSource

Prototype: public abstract int getSource()

Description: Getting the search data source that matches the search.

Parameter: None.

Return: Int type, indicating the search data source of the matching search. For the value, see the constant field definition of "Search Data Source" of the SourceType interface. If the application does not set a matching search data source, it will return SourceType.ALL by default.

H.4.17.2.6 getTimeLimit

Prototype: public abstract int getTimeLimit()

Description: Getting timeout time limit for matching search.

NOTE – If the application does not set a timeout time limit, it will return 0 by default.

Return: Int type, indicating the timeout time limit of matching search, in milliseconds.

H.4.17.2.7 setMaxResults

Prototype: public abstract void setMaxResults(int maxResults)

throws java.lang.IllegalArgumentException

Description: Setting the maximum number of results returned by matching search.

Parameter: maxResults – Int type, indicating the maximum number of results returned by matching search.

Return: None.

Exception: java.lang.IllegalArgumentException – If maxResults is less than or equal to 0 or exceeds the maximum value supported by the system, this exception is thrown.

H.4.17.2.8 setSearchField

Prototype: public abstract void setSearchField(int searchField) throws java.lang.IllegalArgumentException

Description: Setting the fields to be found for matching search. The search engine will search the following fields in the database:

– Title;

– Keyword;

– Synopsis.

Parameter: searchField – Int type, indicating the scope of the search field, for the value, see the constant field definition of "Search Field" of SearchFields interface for details.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter searchField is not defined in the "search field" constant field of the SearchFields interface, this exception is thrown.

H.4.17.2.9 setSearchLanguage

Prototype: public abstract void setSearchLanguage(java.lang.String iso639code) throws java.lang.IllegalArgumentException

Description: Setting the language type of the text information that matches the search.

NOTE – Only one search language can be set in a matching search. If it is not set, the default language is "zho".

Parameter: iso639code – A java.lang.String object, indicating the language type of the text information for matching search. The three-letter language code follows the GB/T 4880.2-2000 standard.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter iso639code does not meet the GB/T 4880.2-2000 standard, this exception is thrown.

H.4.17.2.10 setSource

Prototype: public abstract void setSource(int sourceType) throws java.lang.IllegalArgumentException

Description: Setting the search data source for matching search.

NOTE – If the application does not set a matching search data source, SourceType.ALL will be used by default.

Parameter: sourceType – Int type, indicating the data source of the matching search. For the value, please refer to the constant field definition of "Search Data Source" of org.ngb.toolkit.search.SourceType interface.

Return: None.

Exception: java.lang.IllegalArgumentException – If the search source specified by the parameter sourceType is not defined in the "search data source" constant field of the org.ngb.toolkit.search.SourceType interface, this exception is thrown.

H.4.17.2.11 setTimeLimit

Prototype: public abstract void setTimeLimit(int timeLimit) throws java.lang.IllegalArgumentException

Description: Setting the timeout time limit for matching search.

NOTE – If the timeout period is set, no matter whether there is a matching result or not, the matching search willreturn when the timeout period is reached.

Parameter: timeLimit – Int type, indicating the timeout period, in milliseconds. If you enter 0, it means always waiting.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter timeLimit is less than 0, this exception is thrown.

H.4.18 Class org.ngb.toolkit.search.GlobalSearchFilter

Prototype: public abstract class org.ngb.toolkit.search.GlobalSearchFilter

Description: It describes a global search filter, provides methods for setting and obtaining global search filter conditions. The filter conditions can be:

– Search data source;

– Search field;

– Language of text information;

– Maximum number of results returned;

– Search timeout time limit.

NOTE – The application uses org.ngb.toolkit.search.GlobalSearchFilter to obtain the following effects: According to the specified search data source and search field, the search result list that matches the keyword entered by the user is listed within the timeout time limit. The number of search results is less than or equal to the maximum value set by the application.

H.4.18.1 Constant field

H.4.18.1.1 DEFAULT_MAX_GLOBAL_SEARCH_RESULTS

Prototype: public static final int DEFAULT_MAX_GLOBAL_SEARCH_RESULTS = 50

Description: Constant – it defines the default maximum number of search results. This constant is the maximum value of the default search results. If the application needs to set other values, it can be achieved by calling the setMaxResults(int) method.

H.4.18.2 Method

H.4.18.2.1 getGlobalSearchFilter

Prototype: public static GlobalSearchFilter getGlobalSearchFilter()

Description: Getting an instance of the GlobalSearchFilter class implemented by the system.

Parameter: None.

Return: An org.ngb.toolkit.search.GlobalSearchFilter object, indicating an instance of the GlobalSearchFilter class implemented by the system.

H.4.18.2.2 getContentNibble

Prototype: public abstract int getContentNibble()

Description: Getting the class of the program to be filtered.

Parameter: None.

Return: Int type, indicating the class of the program to be filtered.

H.4.18.2.3 getContentType

Prototype: public abstract int getContentType()

Description: Getting the content type of the program to be filtered.

Parameter: None.

Return: Int type, indicating the content type of the program to be filtered. See the "Content Type" constant field definition of the SearchContentType interface for the value. If the application does not call the setContentType() method to set this value, it will return SearchContentType.ALL by default.

H.4.18.2.4 getCriteriaFlags

Prototype: public abstract long getCriteriaFlags()

Description: Getting the search criteria flag.

Parameter: None.

Return: Int type, indicating the search condition flag. For the value, please refer to the "filter condition" constant field definition of the SearchCriteriaFlags interface. If the application does not use the setCriteriaFlags() method to set this value, it will return SearchCriteriaFlags.FLAG_NONE by default.

H.4.18.2.5 getMaxResults

Prototype: public abstract int getMaxResults()

Description: Getting the maximum number of results returned by the global search.

Parameter: None.

Return: Int type, indicating the maximum number of results returned by the global search. If the application is not set, DEFAULT_MAX_GLOBAL_SEARCH_RESULTS will be returned by default.

H.4.18.2.6 getSearchField

Prototype: public abstract int getSearchField()

Description: Getting the fields to be found in the global search.

Parameter: None.

Return: Int type, indicating the field to be found in the global search. For the value, please refer to the "search field" constant field definition of the org.ngb.toolkit.search.SearchFields interface.

H.4.18.2.7 getSearchLanguage

Prototype: public abstract String getSearchLanguage()

Description: Getting the language type of the text information for global search.

Parameter: None.

Return: A java.lang.String object, indicating the language type of the text information for global search. The three-letter language code follows the GB/T 4880.2-2000 standard.

H.4.18.2.8 getSource

Prototype: public abstract int getSource()

Description: Getting the search data source of the global search.

Parameter: None.

Return: Int type, indicating the search data source of the global search. For the value, please refer to the constant field definition of the "search data source" of the org.ngb.toolkit.search.SourceType interface. If the application does not set the global search data source, it will return SourceType.ALL by default.

H.4.18.2.9 getThreshold

Prototype: public abstract int getThreshold()

Description: Getting the threshold of search query results.

Parameter: None.

Return: Int type, indicating the threshold of search query results. If the application does not set a threshold, it will return 0 by default.

H.4.18.2.10 getTimeLimit

Prototype: public abstract int getTimeLimit()

Description: Getting the timeout time limit of the global search.

Parameter: None.

Return: Int type, indicating the timeout time limit for matching global search, in milliseconds. If the application does not set a timeout time limit, it will return 0 by default.

H.4.18.2.11 setContentNibble

Prototype: public abstract void setContentNibble(int contentNibble) throws java.lang.IllegalArgumentException

Description: Setting the class of the program to be filtered.

Parameter: contentNibble – Int type, indicating the class of the program to be filtered.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter is illegal, this exception is thrown.

H.4.18.2.12 setContentType

Prototype: public abstract void setContentType(int contentType) throws java.lang.IllegalArgumentException

Description: Setting the content type of the program to be filtered.

Parameter: contentType – Int type, indicating the content type of the program to be filtered. The possible values are SearchContentType.ALL, SearchContentType.AUDIO_ONLY, SearchContentType.VIDEO_ONLY, see the "Content Type" constant field definition of the SearchContentType interface for details.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter contentType is not defined in the constant field of the "content type" of the org.ngb.toolkit.search.SearchContentType interface, this exception is thrown.

H.4.18.2.13 setCriteriaFlags

Prototype: public abstract void setCriteriaFlags(long criteriaFlags) throws java.lang.IllegalArgumentException

Description: Setting search criteria flags.

Parameter: criteriaFlags – long type, indicating the search condition, see the "filter criteria" constant field definition of the org.ngb.toolkit.search.SearchCriteriaFlags interface.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter criteriaFlags is not defined in the "filter criteria" constant field of the org.ngb.toolkit.search.SearchCriteriaFlags interface, this exception is thrown.

H.4.18.2.14 setMaxResults

Prototype: public abstract void setMaxResults(int maxResults)

throws java.lang.IllegalArgumentException

Description: Setting the maximum number of results returned by the global search.

Parameter: maxResults – Int type, indicating the maximum number of results returned by the global search.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter maxResults is less than or equal to 0 or exceeds the maximum value supported by the system, this exception is thrown.

H.4.18.2.15 setSearchField

Prototype: public abstract void setSearchField(int searchField)

throws java.lang.IllegalArgumentException

Description: Setting the fields to be found in the global search. The search engine will search the following fields in the database:

– Title;

– Keyword; and

– Synopsis.

Parameter: searchField – Int type, indicating the range of the search field. For the value, please refer to the "search field" constant field definition of the org.ngb.toolkit.search.SearchFields interface.

Return: None.

Exception: java.lang.IllegalArgumentException – If the value of the input parameter searchField is not defined in the "search field" constant field of the org.ngb.toolkit.search.SearchFields interface, this exception is thrown.

H.4.18.2.16 setSearchLanguage

Prototype: public abstract void setSearchLanguage(java.lang.String iso639code) throws java.lang.IllegalArgumentException

Description: Setting the language type of text information for global search.

NOTE – Only one language can be set in a global search. If it is not set, the default language is Chinese "zho".

Parameter: iso639code – A java.lang.String object, indicating the language type of the text information for global search. The three-letter language code follows the provisions of GB/T 4880.2-2000.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter iso639code does not meet the requirements of GB/T 4880.2-2000, the exception is thrown.

H.4.18.2.17 setSource

Prototype: public abstract void setSource(int sourceType) throws java.lang.IllegalArgumentException

Description: Setting the data source to perform the search.

NOTE – If the application does not set the global search data source, SourceType.ALL will be used by default.

Parameter: sourceType – Int type, indicating the global search data source. For the value, please refer to the "Search Data Source" constant field definition of the org.ngb.toolkit.search.SourceType interface.

Return: None.

Exception: java.lang.IllegalArgumentException – If the search source specified by the parameter sourceType is not defined in the "search data source" constant field of the org.ngb.toolkit.search.SourceType interface, this exception is thrown.

H.4.18.2.18 setThreshold

Prototype: public abstract void setThreshold(int thresholdLimit) throws java.lang.IllegalArgumentException

Description: Setting the threshold of search query results. Every time the threshold is reached, the application will receive a notification of SearchStatus.IN_PROGRESS status. If the default threshold is set to 0, the application will receive a status notification of SearchStatus.COMPLETED.

Parameter: thresholdLimit – Int type, indicating the threshold of search query results.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter thresholdLimit is less than 0, this exception is thrown.

H.4.18.2.19 setTimeLimit

Prototype: public abstract void setTimeLimit(int timeLimit) throws java.lang.IllegalArgumentException

Description: Setting the timeout time limit for the global search.

NOTE – If the timeout period is set, the global search will return when the timeout period is reached, regardless of whether there is a result.

Parameter: timeLimit – Int type, indicating the timeout period, in milliseconds. If you enter 0, it means always waiting.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter timeLimit is less than 0, this exception is thrown.

H.4.19 Class org.ngb.toolkit.search.SearchManager

Prototype: public class org.ngb.toolkit.search.SearchManager

Description: A search manager of global search and matching search, being the entry class of the search function module.

The application must follow the following steps to implement the matching search function:

– Create an AutoCompleteSearchFilter object;
– Call the getAutoCompleteSearchSession() method to obtain the AutoCompleteSearchSession object;
– Waiting for the user to enter the query character string;
– After the user input is complete, call the startSearch() method of the org.ngb.toolkit.search.AutoCompleteSearchSession object to start the matching search;
– Within the timeout period, once there are results, the search engine will notify the application of the search status, that is, notify the application through the callback method of the org.ngb.toolkit.search.AutoCompleteSearchListener object;
– After the application obtains the search status from the search engine, it calls the getSearchResultList() method of the org.ngb.toolkit.search.AutoCompleteSearchSession object to obtain a list of matching search results.

The application must follow the following steps to realize the global search function:

– Create org.ngb.toolkit.search.GlobalSearchFilter and org.ngb.toolkit.search.SortCriteria and set the value of each field;
– Call the getGlobalSearchSession() method to get the GlobalSearchSession object;
– Call the setPageSize() method of the org.ngb.toolkit.search.GlobalSearchSession object to set the number of search result items saved on each page;
– Waiting for the user to enter the query character string;
– After the user input is complete, call the startSearch() method of the org.ngb.toolkit.search.GlobalSearchSession object to start the global search function;
– Within the timeout period, once there are results, the search engine will notify the application of the search status, that is, to notify the application through the callback method of the org.ngb.toolkit.search.GlobalSearchListener object;
– After the application gets the search status from the search engine, it calls the retrievePage() method of the org.ngb.toolkit.search.GlobalSearchSession object to obtain the paged search results;
– After the retrieval is successful, the application obtains the global search result list by calling the getSearchResultList() method of the org.ngb.toolkit.search.GlobalSearchSession object.

Tip: Programmers should note that, whether it is global search or matching search, the number of sessions supported by the system are not unlimited. Therefore, the most effective means should be used to realize the search function.

H.4.19.1 Method

H.4.19.1.1 getInstance

Prototype: public static org.ngb.toolkit.search.SearchManager getInstance()

Description: Getting the only instance of the org.ngb.toolkit.search.SearchManager class implemented by the system.

Parameter: None.

Return: An org.ngb.toolkit.search.SearchManager class singleton.

H.4.19.1.2 getAutoCompleteSearchSession

Prototype: public org.ngb.toolkit.search.AutoCompleteSearchSession getAutoCompleteSearchSession (org.ngb.toolkit.search.AutoCompleteSearchFilter filter, org.ngb.toolkit.search.AutoCompleteSearchListener listener)

throws java.lang.IllegalArgumentException

Description: Getting the matching search session object. Each matching search session object is an independent search session. In order to have multiple matching search sessions at the same time, the application needs not only to obtain different objects, but also to handle the corresponding callback function.

Parameter: filter – An org.ngb.toolkit.search.AutoCompleteSearchFilter object, indicating a matching search filter;

listener – An org.ngb.toolkit.search.AutoCompleteSearchListener object, indicating a matching search process listener.

Return: An org.ngb.toolkit.search.AutoCompleteSearchSession object, indicating a matching search session instance.

Exception: java.lang.IllegalArgumentException – If the input parameter is illegal, this exception is thrown.

H.4.19.1.3 getGlobalSearchSession

Prototype: public org.ngb.toolkit.search.GlobalSearchSession getGlobalSearchSession

(org.ngb.toolkit.search.GlobalSearchFilter filter,

org.ngb.toolkit.search.SortCriteria sortCriteria,

org.ngb.toolkit.search.GlobalSearchListener listener)

throws java.lang.IllegalArgumentException

Description: Getting the global search session object. Each global search session object is an independent search session. In order to have multiple global search sessions at the same time, the application needs not only to obtain different objects, but to also handle the corresponding callback function. According to the terminal capabilities, the system decides whether to support multiple global search sessions at the same time.

Parameter: filter – An org.ngb.toolkit.search.GlobalSearchFilter object, indicating a global search filter;

sortCriteria – An org.ngb.toolkit.search.SortCriteria object, indicating the criteria for sorting search results;

listener – An org.ngb.toolkit.search.GlobalSearchListener object, indicating a global search process listener.

Return: An org.ngb.toolkit.search.GlobalSearchSession object, indicating a global search session instance.

Exception: java.lang.IllegalArgumentException – If the input parameter is illegal, this exception is thrown.

H.4.19.1.4 getSearchHistory

Prototype: public SearchHistoryList getSearchHistory()

throws org.ngb.toolkit.search.SearchException

Description: Getting a list of historical search information.

– If the application has completed some searches before this search, this list will contain the search records, and each item in the list contains information such as search keywords and search result sources;
– If the application has not performed any search before this search, no object list will be returned, and an exception of org.ngb.toolkit.search.SearchException is thrown to the application.

Parameter: None.

Return: An org.ngb.toolkit.search.SearchHistoryList object, indicating history search records. If there is no history search record, null is returned.

Exception: org.ngb.toolkit.search.SearchException – If the history search record fails to be obtained, this exception is thrown.

H.4.19.1.5 clearHistory

Prototype: public void clearHistory() throws Org.ngb.toolkit.search.SearchException

Description: Clearing history search records. At any time, the application can call this method to clear history search records to protect personal privacy.

Parameter: None.

Return: None.

Exception: org.ngb.toolkit.search.SearchException – If the history record fails to be cleared, this exception is thrown.

H.4.20 Class org.ngb.toolkit.search.SortCriteria

Prototype: public class org.ngb.toolkit.search.SortCriteria

Description: The sorting constants and sorting methods are defined.

H.4.20.1 Constant field – sorting method

H.4.20.1.1 SORT_ORDER_NONE

Prototype: public static final byte SORT_ORDER_NONE = 0

Description: Sorting method – None.

H.4.20.1.2 SORT_ORDER_ASCENDING

Prototype: public static final byte SORT_ORDER_ASCENDING = 1

Description: Sorting method – ascending.

H.4.20.1.3 SORT_ORDER_DESCENDING

Prototype: public static final byte SORT_ORDER_DESCENDING = 2

Description: Sorting method – descending.

H.4.20.2 Constant field – sorting element

H.4.20.2.1 SORT_TYPE_NONE

Prototype: public static final int SORT_TYPE_NONE = 0

Description: Sorting element – Do not specify the sort field.

H.4.20.2.2 SORT_TYPE_TITLE

Prototype: public static final int SORT_TYPE_TITLE = 1

Description: Sorting element–Sorting based on title.

H.4.20.2.3 SORT_TYPE_START_TIME

Prototype: public static final int SORT_TYPE_START_TIME = 2

Description: Sorting element–Sorting based on start time.

H.4.20.2.4 SORT_TYPE_CONTENT_NIBBLE

Prototype: public static final int SORT_TYPE_CONTENT_NIBBLE = 15

Description: Sorting element–Sorting based on event type.

H.4.20.3 Method

H.4.20.3.1 SortCriteria

Prototype: public SortCriteria()

Description: A Construction method, creating a search sorting criterion object.

Parameter: None.

H.4.20.3.2 getOrder

Prototype: public int getOrder()

Description: Getting a sorting method.

Parameter: None.

Return: Int type, indicating the sorting method. The possible values are SORT_ORDER_NONE, SORT_ORDER_ASCENDING or SORT_ORDER_DESCENDING. For details, please refer to the "Sorting Method" constant field definition of the SortCriteria interface.

H.4.20.3.3 setOrder

Prototype: public void setOrder(int order) throws java.lang.IllegalArgumentException

Description: Setting the sorting method of search results returned.

Parameter: order – Int type, indicating the sorting method, and the possible values are SORT_ORDER_ASCENDING, SORT_ORDER_DESCENDING or SORT_ORDER_NONE. For details, see the "sorting method" constant field definition of the org.ngb.toolkit.search.SortCriteria interface.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter is illegal, this exception is thrown.

H.4.20.3.4 getType

Prototype: public int getType()

Description: Getting the sorting elements.

Parameter: None.

Return: Int type, indicating the sorting element, and the possible values are SORT_TYPE_NONE, SORT_TYPE_TITLE, SORT_TYPE_SOURCE, SORT_TYPE_START_TIME. For details, please refer to the "sorting element" constant field definition of the org.ngb.toolkit.search.SortCriteria interface.

H.4.20.3.5 setType

Prototype: public void setType(int sortType) throws java.lang.IllegalArgumentException

Description: Setting the sorting elements returned by the search results.

Parameter: sortType – Int type, indicating the sorting element, and the possible values are SORT_TYPE_NONE, SORT_TYPE_TITLE, SORT_TYPE_SOURCE, SORT_TYPE_START_TIME. For details, please refer to the "Sorting Element" constant field definition of the SortCriteria interface.

Return: None.

Exception: java.lang.IllegalArgumentException – If the input parameter is illegal, this exception is thrown.

H.4.21 Exception org.ngb.toolkit.search.SearchException

Prototype: public class org.ngb.toolkit.search.SearchException extends java.lang.Exception

Description: It describes the exception when an error occurs in the global search. The possible reasons for failure are:

– Failed to start search;

– Failed to stop search;

– Failed to terminate search session;

– Failed to query search;

– Failed to expand search;
– Failed to create result set.

H.4.21.1 Constant field – search exception

H.4.21.1.1 SEARCH_NOT_STARTED_PREVIOUSLY

Prototype: public static final int SEARCH_NOT_STARTED_PREVIOUSLY = 100

Description: Search exception–indicating the search has not started. This exception is thrown when the application tries to stop or destroy an unstarted search.

H.4.21.1.2 SEARCH_START_FAILED

Prototype: public static final int SEARCH_START_FAILED = 101

Description: Search exception–Search exception.

H.4.21.1.3 SEARCH_STOP_FAILED

Prototype: public static final int SEARCH_STOP_FAILED = 102

Description: Search exception–indicating that stop of the search has failed.

H.4.21.1.4 SEARCH_DISPOSE_FAILED

Prototype: public static final int SEARCH_DISPOSE_FAILED = 103

Description: Search exception–indicating that unregistration of the search has failed.

H.4.21.1.5 SEARCH_RETRIEVAL_FAILED

Prototype: public static final int SEARCH_RETRIEVAL_FAILED = 104

Description: Search exception–indicating that retrieval of the search result has failed.

H.4.21.1.6 SEARCH_EXPAND_FAILED

Prototype: public static final int SEARCH_EXPAND_FAILED = 105

Description: Search exception–indicating that expansion of the search has failed.

H.4.21.1.7 PREVIOUS_SEARCH_NOT_COMPLETED

Prototype: public static final int PREVIOUS_SEARCH_NOT_COMPLETED = 106

Description: Search exception–indicating that the previous search has not been completed.

H.4.21.1.8 RESULT_LIST_CREATION_FAILED

Prototype: public static final int RESULT_LIST_CREATION_FAILED = 107

Description: Search exception–indicating that creation of the search result has failed.

H.4.21.1.9 SAVING_HISTORY_FAILED

Prototype: public static final int SAVING_HISTORY_FAILED = 108

Description: Search exception–indicating that save of the search history record has failed.

H.4.21.1.10 CLEARING_HISTORY_FAILED

Prototype: public static final int CLEARING_HISTORY_FAILED = 109

Description: Search exception–indicating that clarance of the search history record has failed.

H.4.21.1.11 RETRIEVAL_IN_PROGRESS

Prototype: public static final int RETRIEVAL_IN_PROGRESS = 110

Description: Search exception–indicating that another search process is started before the current search process is completed. The application should not initiate a next new search request before a previous search is completed.

H.4.21.2 Method

H.4.21.2.1 getMessageId

Prototype: public int getMessageId()

Description: Getting an identifier of exception.

Parameter: None.

Return: Int type, indicating an identifier of exception. For the value, please refer to the "search exception" constant field definition of the org.ngb.toolkit.search.SearchException class.

Annex I

JAVA-multi-screen interactive unit

(This annex forms an integral part of this Recommendation.)

I.1 Overview

The multi-screen interactive LAN interface supported by this specification supports the finding and connection of the client and server in the LAN, as well as the JAVA interface provided by the multi-screen interactive component to the application.

I.1.1 Scene description

1 Assume that there are currently two TVOS system playback devices A and B, both A and B systems are connected to the same LAN.

2 The application of the A system uses startMultiScreenServer method to start multi-terminal linkage service through a multi-terminal linkage communication interface IMultiScreenService provided by the A system, and transmits multi-terminal linkage component information of the B system to the A system.

3 Next, the application uses findSPs method to search for the B system equipment under the same LAN, and realizes the connection with the B system through the A system with the connect method.

4 After the connection is successful, the A system program receives the connection status information through the IMultiScreenCallBack interface provided by the B system, and informs the A system of the connection status with the onConnected method. At this point, multi-terminal linkage communication between systems A and B have accomplished. The application program can use methods such as inputKeyCode provided by IMultiScreenService and IMultiScreenCallBack interfaces to transfer the operation data information of the A system, or more video data information, to the B system, and realize related operations and playings on the B system.

I.2 Multi-screen interactive module

The multi-screen interactive module can realize the functions of the application client to find, connect, and control the server equipment in the LAN.

The summary of the multi-screen interactive module is shown in Table I.1.

Table I.1 – Summary of multi-screen interactive module

Interface	
IMultiScreenService	A functional interface used to support multi-screen interactive component services in a LAN environment.
IMultiScreenCallBack	A remote access interface provided by the multi-screen interactive component to the outside.

I.2.1 Interface org.tvos.multiscreen.IMultiScreenService

Prototype: public interface org.tvos.multiscreen.IMultiScreenService

Description: A functional interface used to support multi-screen interactive component services in a LAN environment.

I.2.1.1 Constant field – command type

I.2.1.1.1 START_MULTISCREENSERVER

Prototype: public static final int START_MULTISCREENSERVER= (IBinder.FIRST_CALL_TRANSACTION + 0)

Description: Start the server.

I.2.1.1.2 STOP_MULTISCREENSERVER

Prototype: public static final int STOP_MULTISCREENSERVER = (IBinder.FIRST_CALL_TRANSACTION + 1)

Description: Stop the server.

I.2.1.1.3 START_MULTISCREENCLIENT

Prototype: public static final int START_MULTISCREENCLIENT= (IBinder.FIRST_CALL_TRANSACTION + 2)

Description: Start the client.

I.2.1.1.4 STOP_MULTISCREENCLIENT

Prototype: public static final int STOP_MULTISCREENCLIENT = (IBinder.FIRST_CALL_TRANSACTION + 3)

Description: Disconnect the client.

I.2.1.1.5 FIND_SPS

Prototype: public static final int FIND_SPS = (IBinder.FIRST_CALL_TRANSACTION + 4)

Description: Find the server.

I.2.1.1.6 CONNECT

Prototype: public static final int CONNECT = (IBinder.FIRST_CALL_TRANSACTION + 5)

Description: Connect the server.

I.2.1.1.7 SET_CALLBACK

Prototype: public static final int SET_CALLBACK = (IBinder.FIRST_CALL_TRANSACTION + 6)

Description: Setting callback.

I.2.1.1.8 QUERY_INFO

Prototype: public static final int QUERY_INFO = (IBinder.FIRST_CALL_TRANSACTION + 7)

Description: Query information.

I.2.1.1.9 EXEC_CMD

Prototype: public static final int EXEC_CMD = (IBinder.FIRST_CALL_TRANSACTION + 8)

Description: Execute the command.

I.2.1.1.10 INPUT_KEYCODE

Prototype: public static final int INPUT_KEYCODE = (IBinder.FIRST_CALL_TRANSACTION + 9)

Description: Call the key.

I.2.1.1.11 NOTIFY_ALL_REMOTE

Prototype: public static final int NOTIFY_ALL_REMOTE = (IBinder.FIRST_CALL_TRANSACTION + 10)

Description: Notifying all LAN devices.

I.2.1.2 Method

I.2.1.2.1 startMultiScreenServer

Prototype: public int startMultiScreenServer(java.lang.String spName, java.lang.String spDeviceType, java.lang.String spServiceInfo, java.lang.String spVersion,String ipaddress, int port, java.lang.String hostname) throws RemoteException

Description: A Remote interface, starting multi-screen interactive component server.

Parameter: spName – java.lang.String type, name of multi-screen interactive component server;

spVersion – java.lang.String type, version of multi-screen interactive component server;

deviceType – java.lang.String type, device type of multi-screen interactive component server;

port – int type, device port number of the searched multi-screen interactive component.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.2 stopMultiScreenServer

Prototype: public int stopMultiScreenServer() throws RemoteException

Description: A Remote interface, stopping multi-screen interactive component server.

Parameter: None.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.3 startMultiScreenClient

Prototype: public int startMultiScreenClient(java.lang.String clientName) throws RemoteException

Description: A Remote interface, starting multi-screen interactive component client.

Parameter: clientName – java.lang.String type, inputing parameter, name of the multi-screen interactive component client.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.4 stopMultiScreenClient

Prototype: public int stopMultiScreenClient() throws RemoteException

Description: A Remote interface, closing the multi-screen interactive component client.

Parameter: None.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.5 findSPs

Prototype: public int findSPs() throws RemoteException

Description: A Remote interface, searching the multi-screen interactive service component equipment under the LAN.

Parameter: None.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.6 connect

Prototype: public int connect(java.lang.String spName, java.lang.String spDeviceType, java.lang.String spServiceInfo, java.lang.String spVersion, java.lang.String ipaddress, int port, java.lang.String hostname) throws RemoteException

Description: A Remote interface, connected to the device of the multi-screen interactive service component server in the LAN.

Parameter: spName – java.lang.String type, input parameter, name of multi-screen interactive component server;

spDeviceType – java.lang.String type, input parameter, device type of multi-screen interactive component server;

spServiceInfo – java.lang.String type, input parameter, service information of multi-screen interactive component server;

spVersion – java.lang.String type, input parameter, version of multi-screen interactive component server;

ipaddress – java.lang.String type, input parameter, ip address of searched multi-screen interactive component device;

port – int type, input parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, input parameter, host name of searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.7 setCallBack

Prototype: public int setCallBack(org.tvos.os.IBinder ibinder) throws RemoteException

Description: A Remote interface, setting up a remote call callback interface.

Parameter: ibinder – org.tvos.os.Ibinder object, input parameter, remote interface callback instance.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.8 queryInfo

Prototype: public int queryInfo(java.lang.String ipaddress, int port, java.lang.String hostname, java.lang.String cmdid, java.lang.String atrribute, java.lang.String params)throws RemoteException

Description: A Remote interface, multi-screen interactive component client requesting to obtain information.

Parameter: ipaddress – java.lang.String type, input parameter, ip address of searched multi-screen interactive component device;

port – int type, input parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, input parameter, host name of searched multi-screen interactive component device;

cmdid – java.lang.String type, input parameter, instruction id of request information;

atrribute – java.lang.String type, input parameter, instruction name of request information;

params – java.lang.String type, input parameter, instruction parameter of request information.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.9 execCmd

Prototype: public int execCmd(java.lang.String ipaddress, int port, java.lang.String hostname, java.lang.String cmd, java.lang.String param)throws RemoteException

Description: A Remote interface, multi-screen interactive component client requesting to execute commands.

Parameter: ipaddress – java.lang.Sting type, input parameter, ip address of searched multi-screen interactive component device;

port – int type, input parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.Sting type, input parameter, host name of searched multi-screen interactive component device;

action – java.lang.Sting type, input parameter, key command;

param – java.lang.Sting type, input parameter, parameters attached to key commands.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.10 inputKeyCode

Prototype: public int inputKeyCode(java.lang.String ipaddress, int port, java.lang.String hostname, java.lang.String action, java.lang.String param)throws RemoteException

Description: A Remote interface, multi-screen interactive component client sending commands input by virtual keys.

Parameter: ipaddress – java.lang.String type, input parameter, ip address of searched multi-screen interactive component device;

port – int type, input parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, input parameter, host name of searched multi-screen interactive component device;

action – java.lang.String type, input parameter, key command;

param – java.lang.String type, input parameter, parameters attached to key commands.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.1.2.11 boardCastAllDevice

Prototype: public int boardCastAllDevice(java.lang.String cmd, java.lang.String param)throws RemoteException

Description: A Remote interface, multi-screen interactive component server sending broadcast to all connected devices.

Parameter: cmd – java.lang.String type, input parameter, broadcast command;

param – java.lang.String type, input parameter, parameters attached to broadcast commands.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2 Interface org.tvos.multiscreen.IMultiScreenCallBack

Prototype: public interface org.tvos.multiscreen.IMultiScreenCallBack

Description: A functional interface used to support multi-screen interactive component services in a LAN environment.

I.2.2.1 Constant field – state callback

I.2.2.1.1 ON_SP_FOUNDED

Prototype: static final int ON_SP_FOUNDED = (IBinder.FIRST_CALL_TRANSACTION + 0)

Description: When found by the client.

I.2.2.1.2 ON_CONNECTED

Prototype: static final int ON_CONNECTED = (IBinder.FIRST_CALL_TRANSACTION + 1)

Description: When connected to the server.

I.2.2.1.3 ON_CONNECTEDREFUSED

Prototype: static final int ON_CONNECTEDREFUSED= (IBinder.FIRST_CALL_TRANSACTION + 2)

Description: When connection is refused.

I.2.2.1.4 ON_DISCONNECTED

Prototype: static final int ON_DISCONNECTED = (IBinder.FIRST_CALL_TRANSACTION + 3)

Description: When the server is disconnected.

I.2.2.1.5 ON_SERVICE_ACTIVITED

Prototype: static final int ON_SERVICE_ACTIVITED = (IBinder.FIRST_CALL_TRANSACTION + 4)

Description: When the server actively responds.

I.2.2.1.6 ON_SERVICE_DEACTIVITED

Prototype: static final int ON_SERVICE_DEACTIVITED = (IBinder.FIRST_CALL_TRANSACTION + 5)

Description: When the server does not respond.

I.2.2.1.7 ON_QUERY_INFO

Prototype: static final int ON_QUERY_INFO = (IBinder.FIRST_CALL_TRANSACTION + 6)

Description: When the server responds to the query information.

I.2.2.1.8 ON_QUERY_RESPONSE

Prototype: static final int ON_QUERY_RESPONSE= (IBinder.FIRST_CALL_TRANSACTION + 7)

Description: When querying response information.

I.2.2.1.9 ON_EXECUTE

Prototype: static final int ON_EXECUTE = (IBinder.FIRST_CALL_TRANSACTION + 8)

Description: Returning result after the command is executed.

I.2.2.1.10 ON_INPUT

Prototype: static final int ON_INPUT = (IBinder.FIRST_CALL_TRANSACTION + 9)

Description: Returning when callback of input mothod.

I.2.2.1.11 ON_NOTIFY

Prototype: static final int ON_NOTIFY= (IBinder.FIRST_CALL_TRANSACTION + 10)

Description: Returning when notification is received.

I.2.2.2 Method

I.2.2.2.1 onSpFounded

Prototype: public int onSpFounded(java.lang.String spName, java.lang.String spDeviceType, java.lang.String spServiceInfo, java.lang.String spVersion, java.lang.String ipaddress, int port, java.lang.String hostname) throws RemoteException

Description: Notifying JAVA adaptation layer that the multi-screen interactive component service search is completed.

Parameter: spName –java.lang.String type, input parameter, name of searched multi-screen interactive component service;

spDeviceType – java.lang.String type, input parameter, type of searched multi-screen interactive component device;

spServiceInfo – java.lang.String type, input parameter, information of searched multi-screen interactive component server;

spVersion – java.lang.String type, input parameter, version of searched multi-screen interactive component server service;

ipaddress – java.lang.String type, input parameter, ip address of searched multi-screen interactive component device;

port – int type, input parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, input parameter, host name of the searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.2 onConnected

Prototype: public int onConnected(java.lang.String spName, java.lang.String spDeviceType, java.lang.String spServiceInfo, java.lang.String spVersion, java.lang.String ipaddress, int port, java.lang.String hostname) throws RemoteException

Description: Notifying JAVA adaptation layer that the service connection of the multi-screen interactive component is completed.

Parameter: spName – java.lang.String type, input parameter, name of searched multi-screen interactive component service;

spDeviceType – java.lang.String type, input parameter, type of searched multi-screen interactive component device;

spServiceInfo – java.lang.String type, input parameter, information of searched multi-screen interactive component server;

spVersion – java.lang.String type, input parameter, version of searched multi-screen interactive component server service;

ipaddress – java.lang.String type, input parameter, ip address of searched multi-screen interactive component device;

port – int type, input parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, input parameter, host name of the searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.3 onConnectRefused

Prototype: public int onConnectRefused(java.lang.String spName, java.lang.String spDeviceType, java.lang.String spServiceInfo, java.lang.String spVersion, java.lang.String ipaddress, int port, java.lang.String hostname) throws RemoteException

Description: Notifying JAVA adaptation layer that the service connection of the multi-screen interactive component is rejected.

Parameter: spName – java.lang.String type, output parameter, name of searched multi-screen interactive component service;

spDeviceType – java.lang.String type, output parameter, type of searched multi-screen interactive component device;

spServiceInfo – java.lang.String type, output parameter, information of searched multi-screen interactive component server;

spVersion – java.lang.String type, output parameter, version of searched multi-screen interactive component server service;

ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.4 onDisconnected

Prototype: public int onDisconnected(java.lang.String spName, java.lang.String spDeviceType, java.lang.String spServiceInfo, java.lang.String spVersion, java.lang.String ipaddress, int port, java.lang.String hostname) throws RemoteException

Description: Notifying JAVA adaptation layer that the service connection of the multi-screen interactive component is disconnected.

Parameter: spName – java.lang.String type, output parameter, name of searched multi-screen interactive component service;

spDeviceType – java.lang.String type, output parameter, type of searched multi-screen interactive component device;

spServiceInfo – java.lang.String type, output parameter, information of searched multi-screen interactive component server;

spVersion – java.lang.String type, output parameter, version of searched multi-screen interactive component server service;

ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.5 onServiceActivited

Prototype: public int onServiceActivited(java.lang.String ipaddress, int port, java.lang.String hostname) throws RemoteException

Description: Notifying JAVA adaptation layer that the multi-screen interactive component service is activated.

Parameter: ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.6 onServiceDeactived

Prototype: public int onServiceDeactived(java.lang.String ipaddress, int port, java.lang.String hostname) throws RemoteException

Description: Notifying JAVA adaptation layer that the multi-screen interactive component service has been unregistered.

Parameter: ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.7 onQueryInfo

Prototype: public int onQueryInfo(java.lang.String ipaddress, int port, java.lang.String hostname, java.lang.String id, java.lang.String attribute, java.lang.String param) throws RemoteException

Description: Notifying JAVA adaptation layer to have received the data request sent by the client of the multi-screen interactive component.

Parameter: ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device;

id – java.lang.String type, output parameter, command id of the received request;

attribute – java.lang.String type, output parameter, command properties of the received request;

param – java.lang.String type, output parameter, attached parameters of the received request.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.8 onQueryResponse

Prototype: public int onQueryResponse(java.lang.String ipaddress, int port, java.lang.String hostname, java.lang.String id, java.lang.String attribute, java.lang.String param) throws RemoteException

Description: Notifying JAVA adaptation layer that the server of the multi-screen interactive component has responded to the data request.

Parameter: ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device;

id – java.lang.String type, output parameter, command id of the received request;

attribute – java.lang.String type, output parameter, command properties of the received request;

param – java.lang.String type, output parameter, attached parameters of the received request.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.9 onExecute

Prototype: public int onExecute(java.lang.String ipaddress, int port, java.lang.String hostname, java.lang.String cmd, java.lang.String param) throws RemoteException

Description: Notifying the JAVA adaptation layer that the multi-screen interactive component client has sent an execution instruction request.

Parameter: ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device;

cmd – java.lang.String type, output parameter, commands executed;

param – java.lang.String type, output parameter, execute the parameters attached to the command.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.10 onInputKeyCode

Prototype: public int onInputKeyCode(java.lang.String ipaddress, int port, java.lang.String hostname, java.lang.String action, java.lang.String param) throws RemoteException

Description: Notifying the JAVA adaptation layer that the client of the multi-screen interactive component has sent a request to perform key injection.

Parameter: ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device;

cmd – java.lang.String type, output parameter, commands executed;

param – java.lang.String type, output parameter, execute the parameters attached to the command.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.11 onNotify

Prototype: public int onNotify(java.lang.String ipaddress, int port, java.lang.String hostname, java.lang.String cmd, java.lang.String param) throws RemoteException

Description: Notifying the JAVA adaptation layer that the multi-screen interactive component has received the notification.

Parameter: ipaddress – java.lang.String type, output parameter, ip address of searched multi-screen interactive component device;

port – int type, output parameter, port number of searched multi-screen interactive component device;

hostname – java.lang.String type, output parameter, host name of searched multi-screen interactive component device;

cmd – java.lang.String type, output parameter, commands received;

param – java.lang.String type, output parameter, parameters of commands received.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

I.2.2.2.12 onTransact

Prototype: public boolean onTransact(int code, Parcel data, Parcel reply, int flags) throws RemoteException

Description: An AIDL remote call callback interface.

Parameter: code – int type, output parameter, corresponding interface ID.

data – Parcel object, output parameter, corresponding interface data;

reply – Parcel object, input parameter, corresponding interface reply data;

flags – int type, output parameter, remote interface flag bit.

Return: int type, returning 0 if the remote interface is successfully called, otherwise returning an error code.

Annex J

JAVA-DRM management unit

(This annex forms an integral part of this Recommendation.)

J.1 Overview

This specification defines JAVA interfaces related to DRM management.

J.2 DRM management module

The summary of DRM management module is shown in Table J.1.

Table J.1 – Summary of DRM management module

Class	
DrmManager	DRM module management class.
DrmTeeRetVal	Return type class of Drm_SendCommandToTEE method of DrmManager class.

J.2.1 Class org.ngb.drm.services. DrmManager

Prototype: public class DrmManager

Description: A Class of DRM module management.

J.2.1.1 Interface

J.2.1.1.1 OnMessageListener

Prototype: public interface OnMessageListener

Description: A DRM message notification interface, called when DRM Manager sends a notification message, and implemented by DRM APP.

Parameter: None.

Return: None.

J.2.1.1.2 OnLicenseListener

Prototype: public interface OnLicenseListener

Description: A DRM message notification interface, called when DRM Manager sends request message for obtaining license, and implemented by DRM APP.

Parameter: None.

Return: None.

J.2.1.1.3 OnDecryptListener

Prototype: public interface OnDecryptListener

Description: A DRM message notification interface, called when DRM Manager sends a decryption request message, and implemented by DRM APP.

Parameter: None.

Return: None.

J.2.1.2 Method

J.2.1.2.1 setOnMessageListener

Prototype: public synchronized void setOnMessageListener(org.ngb.drm.services.DrmManager.

OnMessageListener messageListener)

Description: Setting a callback function for message notification, which is called when DRM Manager sends a notification message.

Parameter: messageListener – OnMessageListener type, indicating the callback function defined by the interface.

Return: None.

J.2.1.2.2 setOnLicenseListener

Prototype: public synchronized void setOnLicenseListener(org.ngb.drm.services.DrmManager.

OnLicenseListener licenseListener)

Description: Setting a callback function for obtaining a license, which is called when DRM Manager sends a request message for obtaining a license.

Parameter: licenseListener – OnLicenseListener type, indicating the callback function defined by the interface.

Return: None.

J.2.1.2.3 setOnDecryptListener

Prototype: public synchronized void setOnDecryptListener(org.ngb.drm.services.DrmManager.

OnDecryptListener decryptListener)

Description: Setting a callback function for data decryption, which is called when DRM Manager sends a decryption request message.

Parameter: decryptListener – OnDecryptListener type, indicating the callback function defined by the interface.

Return: None.

J.2.1.2.4 registerApp

Prototype: public int registerApp(java.lang.String drmId,byte[] uuid,int register_commandId,

byte[] register_pridata,int enflag,int licensereq_commmandId,

int decrypt_commandId)

Description: DRM APP registration, you can privately define the command ID of licensereq and decrypt.

Parameter: drm_Id – A java.lang.String object, indicating the unique identifier of DRM APP;

uuid – byte[] type, indicating the unique identifier of the TApp corresponding to the DRM APP;

register_commandId – Int type, indicating the registered commandid for communicating with TApp;

register_pridata – Int type, indicating the private registration data carried in the registration for communication with TApp, used for TApp to verify the legality of DRM APP;

enflag – Int type, indicating the method of decryption call;

licensereq_commandId – Int type, indicating the commandid corresponding to the query license;

decrypt_commandId – Int type, indicating the commandid corresponding to the decrypted data.

Return: int type, zero indicating success, non-zero indicating failure.

J.2.1.2.5 unRegisterApp

Prototype: public int unRegisterApp()

Description: DRM APP unregistration.

Parameter: None.

Return: int type, zero indicating success, non-zero indicating failure.

J.2.1.2.6 sendCommandToTEE

Prototype: public DrmTeeRetVal sendCommandToTEE(int commandId, byte[] inputData)

Description: Send commands to TEE, and get data.

Parameter: commandId – Int type, indicating command ID;

inputData – byte[] type, indicating incoming data of TEE.

Return: object of DrmTeeRetVal class.

J.2.1.2.7 sendMessageToPlayer

Prototype: public int sendMessageToPlayer(int type, byte[] message)

Description: Send a message to the player.

Parameter: type – Int type, indicating the type of message;

message – byte[] type, indicating message.

Return: int type, zero indicating success, non-zero indicating failure.

J.2.2 Class org.ngb.drm.services.DrmTeeRetVal

Prototype: public class DrmTeeRetVal

Description: A Returning type class of Drm_SendCommandToTEE method of DrmManager class.

J.2.2.1 Method

J.2.2.1.1 getData

Prototype: public byte[] getData()

Description: Getting data returned by DRM Tapp.

Parameter: None.

Return: data of byte[] type.

J.2.2.1.2 getDatalen

Prototype: public long getDatalen()

Description: Getting length of the data returned by DRM Tapp.

Parameter: None.

Return: long type, length of the data.

J.2.2.1.3 getOriginCode

Prototype: public int getOriginCode()

Description: Getting the source of the return value.

Parameter: None.

Return: int type, 1-indicating API, 2-indicating communication (COMMS), 3-indicating TEE, 4-indicating DRM Tapp.

J.2.2.1.4 getReturnCode

Prototype: public int getReturnCode()

Description: Getting the return value.

Parameter: None.

Return: int type, zero indicating success, non-zero indicating failure.

Annex K

JAVA-DCAS management unit

(This annex forms an integral part of this Recommendation.)

K.1 Overview

This specification defines JAVA interfaces related to DCAS management.

K.2 CAS descrambling module

The CAS descrambling module provides the upper API of the DCAS terminal software platform.

The summary of CAS descrambling module is shown in Table K.1.

Table K.1 – Summary of CAS descrambling module

Interface	
CASModule	A CASModule object used to represent a request to descramble a set of elementary streams.
CASDataUtils	It is used to obtain and set CA information, read and write DCAS data.
CADescriptor	It provides information about CA descriptors, and it is possible to provide CA descriptors in the PMT of a given service. In addition, CA descriptors will also appear in CAT.
CAServiceComponentInfo	It is used to extract the information of specific CA service components, such as ECM PID and DescramblerContext for loading control words.
CASPacketListener	DCAS application receives out-of-band CAS Packets (such as EMMs) through this interface.
CASSession	It provides information related to CAS session.
CAStatus	DCAS application sends CAStatus every time the descrambling status in the DescramblerContext changes. This status is used to indicate whether the descrambling is successful or not. If any one of the descrambling components fails, this state must report the failure of the descrambling request of the entire service. When the terminal software platform receives a new CAStatus, it should notify other applications through the CAS event described in this paragraph.
CATListener	DCAS application needs to implement this interface and use CA descriptor in CAT to filter in-band EMM.
CATNotifier	DCAS application uses this method to register a listener for CAT update notifications.
Class	
CASModuleManager	It is used to register all CASModules implemented by DCAS applications.
CASPermission	Any DCAS application must obtain CASPersmission to access CASModuleManager. This mechanism is used to ensure that only DCAS applications authorized by the network operator can use DCAS API.

K.2.1 Interface org.ngb.net.cas.module.CASModule

Prototype: public interface org.ngb.net.cas.module.CASModule

Description: This interface describes CASModule object used to represent a request to descramble a set of elementary streams.

K.2.1.1 Method

K.2.1.1.1 startDescrambling

Prototype: public void startDescrambling(org.ngb.net.cas.module.CASSession cassession,

org.ngb.net.cas.module.CAServiceComponentInfo[] casci)

Description: This method is called by the terminal software platform to request CASModule to descramble a set of elementary streams in a given session.

DCAS application can obtain a relevant NetworkInterface object from CAS session.

From NetworkInterface, you can get the current TransportStream object, which is used in sectionsAPI for ECM section filtering.

Parameter: cassession – An org.ngb.net.cas.module.CASSession object, session for requesting descrambling operation;

casci – An org.ngb.net.cas.module.CAServiceComponentInfo array, CA service component information array. The array can be used to obtain the relevant ECM PID and the DescramblerContext object used for DCAS to load the control word.

Return: None.

K.2.1.1.2 updateDescrambling

Prototype: public void updateDescrambling(org.ngb.net.cas.module.CASSession casSession, org.ngb.net.cas.module.CAServiceComponentInfo[] casci)

Description: This method is called by a terminal software platform to update the list of descrambling components in the CASModule.

Upon request, CASModule will start descrambling the components added to the array, and stop descrambling the removed components.

For the unchanged components after the update, no changes will occur.

NOTE – This method is rarely called, usually due to PMT changes in a session.

The terminal software platform can also notify the CASModule by calling this method when the CAS session, such as the session type, changes.

Parameter: casSession – An org.ngb.net.cas.module.CASSession object, session for requesting descrambling operation;

casci – org.ngb.net.cas.module.CAServiceComponentInfo array, CA service component information array, this array can be used to obtain the relevant ECM PID and the DescramblerContext object used for DCAS to load the control word.

Return: None.

K.2.1.1.3 stopDescrambling

Prototype: public void stopDescrambling(org.ngb.net.cas.module.CASSession casSession)

Description: The terminal software platform calls this interface to request CASModule to stop descrambling all components in a given session.

Parameter: casSession – org.ngb.net.cas.module.CASSession object, session for requesting descrambling operation.

Return: None.

K.2.1.1.4 getCAInfo

Prototype: public java.lang.String getCAInfo(int cmdId, java.lang.String data)

Description: The terminal software platform calls this interface to obtain CA information.

Parameter: cmdId – int type, unique ID of the command, it can be expanded according to actual projects;

data – java.lang.String type, query parameter.

Return: java.lang.String type, CA information data.

K.2.1.1.5 setCAInfo

Prototype: public int setCAInfo(int cmdId, java.lang.String data)

Description: The terminal software platform calls this interface to set CA information.

Parameter: cmdId – int type, unique ID of the command, it can be expanded according to actual projects;

data – java.lang.String type, CA information data.

Return: int type, return 0 if the setting is successful.

K.2.2 Interface org.ngb.net.cas.module.CASDataUtils

Prototype: public interface org.ngb.net.cas.module.CASDataUtils

Description: This interface description is used to indicate a general interface for obtaining CA-related information.

K.2.2.1 Method

K.2.2.1.1 getCAInfo

Prototype: public java.lang.String getCAInfo(int casId, int cmdId, java.lang.String data)

Description: The terminal software platform obtains CA information from the DCAS application in response to user operations.

Parameter: casId – int type, designated CAS supplier;

cmdId – int type, unique ID of command, it can be expanded according to actual projects;

data – java.lang.String type, query parameter.

Return: java.lang.String type, CA information data.

K.2.2.1.2 setCAInfo

Prototype: public int setCAInfo(int casId, int cmdId, java.lang.String data)

Description: The terminal software platform sets CA information to DCAS application in response to user operations.

Parameter: casId – int type, designated CAS supplier;

cmdId – int type, unique ID of command, it can be expanded according to actual projects;

data – java.lang.String type, query parameter.

Return: int type, return 0 if the setting is successful.

K.2.2.1.3 getData

Prototype: public java.lang.String getData(int casId, int cmdId, int[] type)

Description: The terminal software platform obtains data from DCAS manager in response to user operations.

Parameter: casId – int type, designated CAS supplier;

cmdId – int type, unique ID of command, it can be expanded according to actual projects;

type – an int array, data type reference;

Return: java.lang.String type, data required.

K.2.2.1.4 setData

Prototype: public int setData(int casId, int cmdId, int type, java.lang.String data)

Description: The terminal software platform sets CA information to DCAS application in response to user operations.

Parameter: casId – int type, designated CAS supplier;

cmdId – int type, unique ID of command, it can be expanded according to actual projects;

data – java.lang.String type, data related to DCAS;

type – int type, data type.

Return: int type, return 0 if the setting is successful.

K.2.3 Interface org.ngb.net.cas.module.CADescriptor

Prototype: public interface org.ngb.net.cas.module.CADescripter

Description: This interface provides information of CA descriptor, and it is possible to provide CA descriptor in PMT of a given service. In addition, CA descriptors will also appear in CAT.

K.2.3.1 Method

K.2.3.1.1 getCASystemId

Prototype: public int getCASystemId()

Description: This method returns a CASystemId of CA descriptor.

Parameter: None.

Return: Int type, indicating CASystemId.

K.2.3.1.2 getPid

Prototype: public int getPid()

Description: This method returns a PID (ECM PID or EMM PID) in CA descriptor.

Parameter: None.

Return: Int type, indictaes PID value.

K.2.3.1.3 getPrivateData

Prototype: public byte[] getPrivateData()

Description: This method returns a Private data array of CA descriptor terminal.

Parameter: None.

Return: A byte array, indicating privateData.

K.2.4 Interface org.ngb.net.cas.module.CAServiceComponentInfo

Prototype: public interface org.ngb.net.cas.module.CAServiceComponentInfo

Description: This interface is used to extract information about specific CA service components, such as ECM PID and DescramblerContext for loading control words.

K.2.4.1 Method

K.2.4.1.1 getDescramblerContext

Prototype: public org.ngb.net.cas.controller.DescramblerContext getDescramblerContext()

Description: This method returns DecramberContext object used for DCAS application to load the control word. In the case of multiple occurrences of the same CA descriptor (same ECMPID and private data) in the component loop of PMT, there should be only one DecramberContext.

Parameter: None.

Return: An org.ngb.net.cas.controller.DescramblerContext object.

K.2.4.1.2 getCADescriptor

Prototype: public org.ngb.net.cas.module.CADescriptor getCADescriptor()

Description: This method returns a CA descriptor related to the service component, and CADescriptor instance is generated from CA information in PMT.

Parameter: None.

Return: An org.ngb.net.cas.module.CADescriptor object.

K.2.4.1.3 getComponentStreamPIDs

Prototype: public int[] getComponentStreamPIDs()

Description: This method returns an Elementary stream array described in PMT. The order of the array elements should be the same as the order of the array elements returned by getComponentStreamType.

Parameter: None.

Return: An ES (Elementary Stream) PID array.

K.2.4.1.4 getComponentStreamTypes

Prototype: public int[] getComponentStreamTypes()

Description: This method returns stream type array in PMT, and stream type value should follow the MPEG standard ISO/IEC 13818-1. The order of the array elements should be consistent with the order of the array elements returned by getComponentStreamPID.

Parameter: None.

Return: An int[], stream type array.

K.2.4.1.5 getServiceIdentifiers

Prototype: public int[] getServiceIdentifiers()

Description: This method returns an array of service identifiers associated with the object, and the representation form of service identifiers depends on the specific use environment.

Parameter: None.

Return: An int[],ServiceID array.

K.2.5 Interface org.ngb.net.cas.module.CASPacketListener

Prototype: public interface org.ngb.net.cas.module.CASPacketListener

Description: DCAS application receives out-of-band CAS Packets (such as EMMs) through this interface. DCAS application registers the listener through registerCasPacketListener method provided by CASModuleManager class according to the given CA system identifier. The CA system

identifier is represented by parameter casId. The receiving of CAS packages depends on the terminal software platform.

K.2.5.1 Method

K.2.5.1.1 casPacketArrived

Prototype: public void casPacketArrived(int casId, byte [] casPacketData, byte [] casPacketHeader)

Description: DCAS application, obtaining the CAS package through the registered listener.

Parameter: casId – int type, CA system identifier;

casPacketData – A byte array, CAS packet data;

casPacketHeader – A byte array, depending on CAS header of the terminal software platform.

Return: None.

K.2.6 Interface org.ngb.net.cas.module.CASSession

Prototype: public interface org.ngb.net.cas.module.CASSession

Description: This interface provides Information related to the session.

K.2.6.1 Constant field – session type

K.2.6.1.1 TYPE_PRESENTATION

Prototype: public static final int TYPE_PRESENTATION = 0x00000001

Description: It indicates that the session is TYPE_PRESENTATION type.

K.2.6.1.2 TYPE_RECORDING

Prototype: public static final int TYPE_RECORDING = 0x00000002

Description: It indicates the session is TYPE_RECORDING type.

K.2.6.1.3 TYPE_BUFFERING

Prototype: public static final int TYPE_BUFFERING = 0x00000004

Description: It indicates the session is TYPE_BUFFERING type.

K.2.6.2 Method

K.2.6.2.1 getType

Prototype: public int getType()

Description: This method returns an Operation type of this session.

Parameter: None.

Return: int type, operation type, it can be one or a combination of the values defined in this interface.

K.2.6.2.2 getNetworkInterface

Prototype: public org.davic.net.tuning.NetworkInterface getNetworkInterface()

Description: This method returns NetworkInterface associated with the CAS session, and DCAS application can obtain the relevant NetworkInterface object from the CAS session. Using NetworkInterface, the DCAS application can first get the TransportStream object, which is used to call the sections application interface for ECM section filtering.

Parameter: None.

Return: One org.davic.net.tuning.NetworkInterface object.

K.2.6.2.3 getAssociatedService

Prototype: public java.lang.Object getAssociatedService()

Description: This method returns a service related to CAS session.

Parameter: None.

Return: One Service object.

K.2.6.2.4 getServiceContext

Prototype: public java.lang.Object getServiceContext()

Description: This method returns ServiceContext related to CAS session.

NOTE – In some implementations, ServiceContext has no practical meaning, and this method will return null.

Parameter: None.

Return: One ServiceContext object.

K.2.7 Interface org.ngb.net.cas.module.CAStatus

Prototype: public interface org.ngb.net.cas.module.CAStatus

Description: A DCAS application uses this interface when calling sendDescramblingEvent method in CASModuleManager. The DCAS application sends CAStatus every time the descrambling status in DescramblerContext changes. This status is used to indicate the success or failure of the descrambling. If any of the descrambling components fails, this status must report the failure of the descrambling request of the entire service. When the terminal software platform receives a new CAStatus, it should notify other applications through the CAS event described in this paragraph. The specific application interface is described in the extended application interface section.

K.2.7.1 Method

K.2.7.1.1 isSuccess

Prototype: public boolean isSuccess()

Description: This method returns Status of descrambling request.

Parameter: None.

Return: boolean type, returning true if descrambling succeeds, and returning false if descrambling fails.

K.2.7.1.2 getCAToken

Prototype: public int getCAToken()

Description: This method returns Parameters for other applications to query DCAS application for network-related information through IXC.

Parameter: None.

Return: Int type, indicating a CA token.

K.2.8 Interface org.ngb.net.cas.module.CATListener

Prototype: public interface org.ngb.net.cas.module.CATListener

Description: DCAS application needs to implement this interface and use CA descriptor in CAT to filter in-band EMM. DCAS application needs to register this listener through registerCATListener defined in CATNotifier interface.

K.2.8.1 Method

K.2.8.1.1 catUpdate

Prototype: public void catUpdate(org.ngb.net.cas.module.CADescriptor desc, org.davic.net.tuning.NetworkInterface ni)

Description: This interface is used to notify the DCAS application of CAT updates on a specific network interface. A DCAS application can obtain the current TransportStream object through NetworkInterface object. The TransportStream object can use the sections application program interface to implement EMM section filtering. The terminal software platform notifies the CAT update to the registered CAT listener that matches the casId.

Parameter: desc – An org.ngb.net.cas.module.CADescriptor object, DCAS application obtains EMM PID through a CASDescriptor object;

ni – An org.davic.net.tuning.NetworkInterface object, which is a NetworkInterface where the CAT update is located.

Return: None.

K.2.9 Interface org.ngb.net.cas.module.CATNotifier

Prototype: public interface org.ngb.net.cas.module.CATNotifier

Description: The DCAS application uses this method to register the listener used to obtain the CAT update notification, and the DCAS application uses the CAT information to filter the in-band EMM.

K.2.9.1 Method

K.2.9.1.1 registerCATListener

Prototype: public void registerCATListener(int casId, org.ngb.net.cas.module.CATListener catListener)

Description: This method is called by a DCAS application to register a CATListener.

Parameter: casId – int type, CA system identifier;

catListener – org.ngb.net.cas.module.CATListener object, owing to the registered CATListener.

Return: None.

K.2.9.1.2 unregisterCATListener

Prototype: public void unregisterCATListener(org.ngb.net.cas.module.CATListener catListener)

Description: This method is called by a DCAS application to unregister a CATListener.

Parameter: catListener – an org.ngb.net.cas.module.CATListener object, which is a CATListener needs to be unregistered.

Return: None.

K.2.10 Class org.ngb.net.cas.module.CASModuleManager

Prototype: public class org.ngb.net.cas.module.CASModuleManager

Description: It is used to register all CASModules implemented by DCAS applications.

K.2.10.1 Method

K.2.10.1.1 getInstance

Prototype: public static org.ngb.net.cas.module.CASModuleManager getInstance() throws java.lang.SecurityException

Description: This method is used to obtain a CASModuleManager singleton.

Parameter: None.

Return: An org.ngb.net.cas.module.CASModuleManager instance.

Exception: java.lang.SecurityException – This exception is thrown when a security policy is forcedly enabled without a CASPermission given to the caller.

K.2.10.1.2 registerCASmodule

Prototype: public void registerCASModule(org.ngb.net.cas.module.CASModule aModule, int caSystemId, int networkCAPriority, java.lang.Object context)

throws java.lang.IllegalArgumentException

Description: This method is used for a DCAS application to register a CASModule on the terminal software platform.

Parameter: aModule – org.ngb.net.cas.module.CASModule object, CASModule that needs to be registered;

caSystemId – int type, caSystemId managed by CASModule;

networkCAPriority – int type, used when more than one CASModule is registered in CASModuleManager. The operator can decide whether this parameter is optional according to each CASModule, when the priority strategy is enabled, the operator needs to specify priorities for each CASModule. The terminal software platform should select the registered, the highest priority, and the managed caSystemId-int type, CASModule with the corresponding CA descriptor in PMT to send the descrambling request. When the priority strategy is disabled, the DCAS application should set this parameter to zero, and the determination method of CASModule is implemented by the terminal software platform;

context – A java.lang.Object object, Context of the DCAS application wishing to register the CASModule, being used for the terminal software platform to determine the identity of the DCAS application.

Return: None.

Exception: java.lang.IllegalArgumentException – If the specified CASModule instance has been registered, this exception is thrown.

K.2.10.1.3 updateCASystemId

Prototype: public void updateCASystemId(org.ngb.net.cas.module.CASModule aModule, int caSystemId)

throws java.lang.IllegalArgumentException

Description: This method is used for the DCAS application to update the CASystemId in a CASModule to the application software platform.

Parameter: aModule – org.ngb.net.cas.module.CASModule object, specify CASModule;

caSystemId – int type, new caSystemId managed by the module.

Return: None.

Exception: A java.lang.IllegalArgumentException, if the given CASModule instance is not yet registered.

K.2.10.1.4 sendDescramblingEvent

Prototype: public void sendDescramblingEvent(org.ngb.net.cas.module.CASModule aModule,

org.ngb.net.cas.module.CASSession casSession, org.ngb.net.cas.module.CAStatus aCAStatus)

throws java.lang.IllegalArgumentException

Description: This method is used by the DCAS application to return a CAStatus to the terminal software platform. When a certain DescramblerContext changes due to changes of a scrambling subpart of the scrambling component in the corresponding service, the DCAS application must send CAStatus to indicate whether the descrambling is successful. If any subpart fails to descramble, CAStatus must notify that the descrambling of the service failed. When the terminal software platform receives a new CAStatus, it should continue to pass the information to the corresponding application through the CAS Event defined in the extended API.

Parameter: aModule – An org.ngb.net.cas.module.CASModule object, specifying CASModule;

casSession – An org.ngb.net.cas.module.CASSession object, descrambling the session that requests an operation;

aCAStatus – An org.ngb.net.cas.module.CAStatus object, being a CAStatus that needs to be sent.

Return: None.

Exception: java.lang.IllegalArgumentException-If the given CASModule instance is not registered, this exception is thrown.

K.2.10.1.5 unregisterCASModule

Prototype: public void unregisterCASModule(org.ngb.net.cas.module.CASModule aModule) throws java.lang.IllegalArgumentException

Description: This method is used for the DCAS application to cancel the registration of CASModule from the terminal software platform.

Parameter: aModule – org.ngb.net.cas.module.CASModule object, CASModule that needs to be unregistered.

Return: None.

Exception: java.lang.IllegalArgumentException, if the given CASModule has not been registered, this exception is thrown.

K.2.10.1.6 getChipControllers

Prototype: public org.ngb.net.cas.controller.ChipController[] getChipControllers()

Description: This method is used by the DCAS application to request a list of chip controllers that can be used from the terminal software platform. This method returns a chip controller for each terminal security chip. For many terminals supporting only one single chip controller, any one of the arrays they return contains only one element.

Parameter: None.

Return: An org.ngb.net.cas.controller.ChipController array, an array of chip controllers.

K.2.10.1.7 setcurrentController

Prototype: public void setCurrentController(org.ngb.net.cas.module.CASModule aModule, org.ngb.net.cas.controller.ChipController aChipController)throws
java.lang.IllegalArgumentException

Description: This method is used to set a chip controller used by default for descrambling operation according to the given CAModule. If this method is not called, the selection in CASModuleManager does not need to be specified.

Parameter: aModule – An org.ngb.net.cas.module.CASModule object, specifying CASModule;

aChipController – An org.ngb.net.cas.controller.ChipController object,

CASModule – default chip controller used.

Return: None.

Exception: java.lang.IllegalArgumentException, if the given CASModule has not been registered, this exception is thrown.

K.2.10.1.8 setCCIBits

Prototype: public void setCCIBits(org.ngb.net.cas.module.CASModule aModule, org.ngb.net.cas.module.CASSession casSession, int cciBits)

Description: This method is used to set the copy control information data (CCI bits) required by the terminal for copy protection of this service. The definition of CCI bit information is specified and interpreted by the terminal software platform.

Parameter: aModule – an org.ngb.net.cas.module.CASModule object, specifying CASModule;

casSession – An org.ngb.net.cas.module.CASSession object, descrambling a request session;

cciBits – int type, CCI bit value used by the current service;

Return: None.

K.2.10.1.9 setServiceListFilter

Prototype: public void setServiceListFilter(int filterData)

Description: This method is used to provide the terminal software platform with parameters for service list filtering, and the specific meaning of the service list parameters is specified and executed by the terminal software platform.

Parameter: filterData – int type, service list filter parameters.

Return: None.

K.2.10.1.10 registerCASPacketListener

Prototype: public void registerCASPacketListener(int casId, org.ngb.net.cas.module.CASPacketListener casPacketListener) throws java.lang.IllegalArgumentException

Description: This method is used for the DCAS application to register a CAPacketListener. The CAPacketListener is called by the terminal software platform and transmits CAS data packets (such as EMM) to the DCAS application. The CA system identifier is represented by the parameter casID, and the receiving of the CAS data packet is implemented by the terminal software platform.

Parameter: casId – int type, CasystemID;

casPacketListener – org.ngb.net.cas.module.CASPacketListener object, CASPacketListener that needs to be registered.

Return: None.

Exception: java.lang.IllegalArgumentException, thrown if the given casId has already registered the listener.

K.2.10.1.11 unregisterCASPacketListener

Prototype: public void unregisterCASPacketListener(

org.ngb.net.cas.module.CASPacketListener casPacketListener)

throws java.lang.IllegalArgumentException

Description: This method is used for DCAS application to cancel the registration of CASPacketListener.

Parameter: casPacketListener – An org.ngb.net.cas.module.CASPacketListener object, which is a listener that needs to be unregistered.

Return: None.

Exception: A java.lang.IllegalArgumentException, thrown if the given CASPacketListener has not been registered.

K.2.10.1.12 getDetachableSecurityDevices

Prototype: public org.ngb.net.cas.detachable.DetachableSecurityDevice[] getDetachableSecurityDevices()

Description: This method is used for the DCAS application to obtain object handles of detachable devices (smart cards, etc.).

Parameter: None.

Return: org.ngb.net.cas.detachable.DetachableSecurityDevice object array.

K.2.10.1.13 receiveOsdMsg

Prototype: public void receiveOsdMsg(byte[] msg, int[] flags)

Description: Display OSD information, the specific meaning of its parameters are related to specific items.

Parameter: msg – A byte array, OSD information content, which can include descriptive information in addition to the text;

flags – An int array, OSD type indication.

Return: None.

K.2.10.1.14 showFingerMsg

Prototype: public void showFingerMsg(org.ngb.net.cas.module.CASModule aModule,

org.ngb.net.cas.module.CASSession casSession, byte[] msg)

Description: Display OSD information, the specific meaning of its parameters are related to specific items.

Parameter: aModule – An org.ngb.net.cas.module.CASModule object, specifying CASModule;

casSession – An org.ngb.net.cas.module.CASSession object, the session that requests a descrambling operation;

msg – A byte array, displaying information when fingerprint information is empty.

Return: None.

K.2.10.1.15 receiveTuningAlert

Prototype: public void receiveTuningAlert(int[] serviceIdentifiers, int[] flags)

Description: Emergency broadcast. In some projects, the parameters of emergency broadcast are not issued through CA system, and in such cases, it is not necessary to implement this function.

Parameter: serviceIdentifiers –An int array, a set of values used to indicate the parameters of the emergency broadcast channel. The meaning of the value is defined by the specific item;

flags – An int array, can be used to indicate the type of emergency broadcast parameters.

Return: None.

K.2.10.1.16 getCATNotifier

Prototype: public org.ngb.net.cas.module.CATNotifier getCATNotifier()

Description: This method is called by the DCAS application to obtain the CATnotifer object, the DCAS application can register the listener for CAT update notification through the CAT notifier. The DCAS application requires CAT information to filter the in-band EMM.

Parameter: None.

Return: org.ngb.net.cas.module.CATNotifier object.

K.2.11 Class org.ngb.net.cas.module.CASPermission

Prototype: public class org.ngb.net.cas.module.CASPermission extends java.security.BasicPermission

Description: Any DCAS application must obtain CASPersmission to access CASModuleManager. This mechanism is used to ensure that only DCAS applications authorized by the network operator can use the DCAS API.

K.2.11.1 Method

K.2.11.1.1 CASPermission

Prototype: public CASPermission(java.lang.String name)

Description: Creating a new CASPermission. Name character string is not currently used and should be set to an empty character string.

Parameter: name – A java.lang.String type, name of this CASPermission.

Return: None.

K.2.11.1.2 CASPermission

Prototype: public CASPermission(java.lang.String name, java.lang.String actions)

Description: Creating a new CASPermission. Name character string is not used now and should be set to an empty character string, actions character string is not used now and should be set to null. This constructor method is used to instantiate a new Permission objects for the Policy object.

Parameter: name – java.lang.String type, name of this CASPermission.

actions – A java.lang.String type, action list.

Return: None.

K.3 CAS control module

CAS control module provides the underlying API of the DCAS terminal software platform.

The summary of CAS control module is shown in Table K.2.

Table K.2 – Summary of CAS control module

Interface	
DescramblerContext	A component used to control the descrambling function of the terminal security chip. Multiple DescramblerContext can be instantiated to use different keys to descramble multiple code streams.
ChipController	A component used to control the execution of the terminal security chip.
Class	

Table K.2 – Summary of CAS control module

Key	A basic cryptographic key. It is used to determine the cryptographic algorithm used by K-LAD and the output parameters of the cryptographic function.
CWKey	Descramble the key or control word.
CASTEEManager	Functional interface for communication with TA in TEE.

K.3.1 Interface org.ngb.net.cas.controller.DescramblerContext

Prototype: public interface org.ngb.net.cas.controller.DescramblerContext

Description: It indicates the component used to control the descrambling function of the terminal security chip. Multiple DescramblerContext can be instantiated to use different keys to descramble multiple code streams.

K.3.1.1 Method

K.3.1.1.1 loadCW

Prototype: public void loadCW(int Vendor_SysID, org.ngb.net.cas.controller.CWKey cwKey,

org.ngb.net.cas.controller.Key[] levelKeys, int schemeId)

throws org.ngb.net.cas.controller.CADriverException

Description: This method is used to notify the terminal software platform to load the control word into the descrambler and load the required key into the terminal security chip. A descrambler channel is a logical set of all streams descrambled by a single control word. The use of the descrambler channel depends on the DescrablerContext. In addition, the DCAS application should notify the terminal software platform that the current control word has expired (for example, due to authorization), and the terminal software platform should stop the corresponding descrambling behavior. In this case, the DCAS application will provide a null CWKey.

Parameter: Vendor_SysID – int type, this value is used to identify the CA manufacturer to support root key derivation in the controller. The root key of the terminal security chip is derived from this value, otherwise Vendor_SysID is ignored;

cwKey – org.ngb.net.cas.controller.CWKey control word object, if the control word is plain text, the levelKeys parameter is ignored. If cwKey is null, that means the DCAS application has not provided a valid control word;

levelKeys – org.ngb.net.cas.controller.Key array, a multi-level key used to install the terminal security chip. The index of the key array is equal to the absolute position in the terminal security chip. The value of a specific element in the array being Null indicates that the corresponding position in the terminal security chip should not be loaded with a key, that is:

levelKey[0] is Key1 (encrypted by Key2); levelKey[1] is Key2 (encrypted by Key3); levelKey[2] needs not to be used;

schemeId – int type, this schemeId is used to specify the encryption algorithm (for example, AES, TDES) of the terminal security chip, and a list of scheme values is defined in the ChipController interface. If the controller supports only one mode, this value is ignored.

Return: None.

Exception: org.ngb.net.cas.controller.CADriverException, if the loading fails, throw this exception.

K.3.1.1.2 overrideChipController

Prototype: public void overrideChipController(org.ngb.net.cas.controller.ChipController aChipController)

throws org.ngb.net.cas.controller.CADriverException

Description: This method is used for the realization of the DCAS application requesting the terminal software platform to cover the default terminal security chip level key. It can be set by calling the setCurrentController method of CASModuleManager. If this method is not called, the terminal security chip will use the default controller. This method is only used in a terminal security chip system that implements multiple terminal security chips.

Parameter: aChipController – An org.ngb.net.cas.controller.ChipController object, a controller to be covered.

Return: None.

Exception: An org.ngb.net.cas.controller.CADriverException, if the operation fails, this exception is thrown.

K.3.2 Interface org.ngb.net.cas.controller.Chipcontroller

Prototype: public interface org.ngb.net.cas.controller.ChipController

Description: It indicates the component that controls the execution of the terminal's security chip.

K.3.2.1 Constant field

K.3.2.1.1 SCHEME_TDES

Prototype: public static final int SCHEME_TDES=0

Description: A value to indicate that the terminal security chip should use TDES.

K.3.2.1.2 SCHEME_AES

Prototype: public static final int SCHEME_AES=1

Description: A value to indicate that the terminal security chip should use AES.

K.3.2.1.3 PROCESSING_MODE_REGULAR

Prototype: public static final int PROCESSING_MODE_REGULAR=0

Description: A value to indicate that no additional processing is required in the terminal security chip authentication response algorithm.

K.3.2.1.4 PROCESSING_MODE_POST_PROCESSING

Prototype: public static final int PROCESSING_MODE_POST_PROCESSING=1

Description: A value to indicate that post-processing stage needs to be implemented in the terminal security chip authentication response algorithm.

K.3.2.2 Method

K.3.2.2.1 getPublicId

Prototype: public byte[] getPublicId() throws org.ngb.net.cas.controller.CADriverException

Description: This method returns Public identifier of the terminal security chip.

Parameter: None.

Return: A byte array, public identifier publicId of the terminal security chip.

Exception: org.ngb.net.cas.controller.CADriverException, if there is a communication error during the drive of the security chip of the access terminal, this exception is thrown.

K.3.2.2.2 getChipType

Prototype: public byte[] getChipType() throws org.ngb.net.cas.controller.CADriverException

Description: This method returns Type identifier of the terminal security chip.

Parameter: None.

Return: A byte array, type of terminal security chip.

Exception: org.net.net.cas.controller.CADriverException, if there is a communication error during the drive of the security chip of the access terminal, this exception is thrown.

K.3.2.2.3 getChipControllerProperty

Prototype: public java.lang.String getChipControllerProperty(java.lang.String propertyName) throws org.ngb.net.cas.controller.CADriverException

Description: According to the provided terminal security chip property name, this method returns the value corresponding to the property. This function is reserved in this interface and can be used to read the properties of the controller that will be expanded in the future. No property names are defined at this stage.

Parameter: propertyName – java.lang.String type, property name.

Return: java.lang.String type, indicating property value.

Exception: org.ngb.net.cas.controller.CADriverException, if there is a communication error during the drive of the security chip of the access terminal, this exception is thrown.

K.3.2.2.4 authenticate

Prototype: public byte[] authenticate(int Vendor_SysID, byte[] challenge, org.ngb.net.cas.controller.Key[] levelKeys, int schemeId, int processingMode) throws org.ngb.net.cas.controller.CADriverException

Description: This method is used to authenticate the hierarchical key mechanism in the terminal security chip, and the terminal security chip should calculate the authentication information based on the sent random handshake information.

Parameter: Vendor_SysID – int type, this value is used to identify the CA supplier. It is used in the controller to support root key derivation. The root key of the terminal security chip is derived from this value. Otherwise, Vendor_SysID is ignored;

Challenge – A byte array, handshake information, random number;

levelKeys – An org.ngb.net.cas.controller.Key array, all levels of keys required by hierarchical keys. The index of the key array is equal to the absolute position in the terminal security chip. The value of a specific element in the array is Null, indicating that the corresponding position in the terminal security chip should not be loaded with a key. That is: levelKey[0] is null; levelKey[1] is Key 2 (encrypted by Key 3); levelKey[2] does not need to be used.

schemeId – int type, this schemeId is used to specify the encryption algorithm (for example, AES, TDES) of the terminal security chip. If the controller only supports one method, this value is ignored;

processingMode – int type, a value used to specify whether to implement additional post-processing during the calculation of the response result. If the controller only supports no post-processing mode, this parameter is ignored.

Return: A byte array, a response response calculated by the terminal security chip.

Exception: An org.ngb.net.cas.controller.CADriverException, if there is a communication error during the drive of the security chip of the access terminal, this exception is thrown.

K.3.2.2.5 encryptData

Prototype: public void encryptData(int Vendor_SysID,

org.ngb.net.cas.controller.CWKey cwKey,

org.ngb.net.cas.controller.Key[] levelKeys,

int schemeId,

int encryptionId,

byte[] src,

int srcPos,

byte[] dest,

int destPos,

int length)

throws org.ngb.net.cas.controller.CADriverException

Description: This method calls a function of the chip to encrypt data in the memory.

Parameter: Vendor_SysID – int type, this parameter being used to indicate the CA manufacturer. The security chip uses this value to derive the root key;

cwKey – An org.ngb.net.cas.controller. A CWKey object, control word for encryption. If the control word is not encrypted, the subsequent levelKeys will be ignored;

levelKeys – An org.ngb.net.cas.controller. A Key array, hierarchical key, the index value of the key in the array is equal to its absolute position in the hierarchical key. The Null element in the array indicates that no key needs to be set at this level;

schemeId – int type, an encryption algorithm used by the hierarchical key. If the chip only supports one algorithm, this parameter will be ignored;

encryptionId – int type, a data encryption/decryption algorithm (such as AES, TDES). If the chip only supports one algorithm, this parameter will be ignored;

src – A byte array, source data array;

srcPos – int type, starting position of the source data array;

dest – A byte array, destination data array;

destPos – int type, starting position of the destination data array;

length – int type, the number of data bytes that need to be processed.

Exception: When the hierarchical key communication error occurs, org.ngb.net.cas.controller.CADriverException is thrown.

K.3.2.2.6 decryptData

Prototype: public void decryptData(int Vendor_SysID,

org.ngb.net.cas.controller.CWKey cwKey,

org.ngb.net.cas.controller.Key[] levelKeys,

int schemeId,

int encryptionId,

byte[] src,

int srcPos,

byte[] dest,

int destPos,

int length)

throws org.ngb.net.cas.controller.CADriverException

Description: This method calls a function of the chip to decrypt the data in the memory.

Parameter: Vendor_SysID – int type, this parameter being used to indicate the CA manufacturer. The security chip uses this value to derive the root key;

cwKey – An org.ngb.net.cas.controller.CWKey object, control word for decryption. If the control word is not encrypted, the subsequent levelKeys will be ignored;

levelKeys – An org.ngb.net.cas.controller.Key array, hierarchical key. The index value of the key in the array is equal to its absolute position in the hierarchical key. The Null element in the array indicates that no key needs to be set at this level;

schemeId – int type, an encryption algorithm used by the hierarchical key. If the chip only supports one algorithm, this parameter will be ignored;

encryptionId – int type, data encryption/decryption algorithm (such as AES, TDES). If the chip only supports one algorithm, this parameter will be ignored;

src – A byte array, source data array;

srcPos – int type, starting position of the source data array;

dest – A byte array, destination data array;

destPos – int type, starting position of the destination data array;

length – int type, the number of data bytes that need to be processed.

Return: None.

Exception: When the hierarchical key communication error occurs, org.ngb.net.cas.controller.CADriverException is thrown.

K.3.3 Class org.ngb.net.cas.controller.Key

Prototype: public class org.ngb.net.cas.controller.Key

Description: It indicates a basic cryptographic key. It is used to determine the cryptographic algorithm used by K-LAD and the output parameters of the cryptographic function.

K.3.3.1 Method

K.3.3.1.1 Key

Prototype: public Key(byte[] value, boolean encrypted)

Parameter: value – A byte array, value of the key;

encrypted – boolean type, a sign of whether the key is encrypted, true indicating the key has been encrypted, false indicating the key is plain text.

K.3.3.1.2 getKeyValue

Prototype: public byte[] getKeyValue()

Description: This method returns Value of the key.

Parameter: None.

Return: A byte array, value of the key.

K.3.3.1.3 isEncrypted

Prototype: public boolean isEncrypted()

Description: When this method returns true, it means that the key is encrypted, and false means that the key is not encrypted.

Parameter: None.

Return: boolean type, true indicating the key is encrypted, false indicating the key is not encrypted.

K.3.4 Class org.ngb.net.cas.controller.CWKey

Prototype: public class org.ngb.net.cas.controller.CWKey extends org.ngb.net.cas.controller.Key

Description: It indicates the descrambling key or control word.

K.3.4.1 Constant field

K.3.4.1.1 PARITY_EVEN

Prototype: public static final int PARITY_EVEN = 0

Description: It indicates PARITY_EVEN type.

K.3.4.1.2 PARITY_ODD

Prototype: public static final int PARITY_ODD = 1

Description: It indicates PARITY_ODD type.

K.3.4.2 Method

K.3.4.2.1 CWKey

Prototype: public CWKey(byte[] value, boolean encrypted, int parity)

Description: Constructed function.

Parameter: value – A byte array, value of the key;

encrypted – boolean type, true value indicating that the key is encrypted, and false value indicating that the key is not encrypted;

parity – int type, parity value, which indicates the parity of the control word.

K.3.4.2.2 getParity

Prototype: public int getParity()

Description: This method returns parity of the control word.

Parameter: None.

Return: int type, parity of the control word.

K.3.5 Class org.ngb.net.cas.controller.CASTEEManager

Prototype: public class org.ngb.net.cas.controller.CASTEEManager

Description: Interface for communication with TA in TEE.

K.3.5.1 Method

K.3.5.1.1 sendCommandToTEE

Prototype: public byte[] sendCommandToTEE(byte[] teeAppUUID, int commandId, byte[] inputData) throws org.ngb.net.cas.controller.CADriverException

Description: The DCAS application selects the corresponding security application and sends data to the security application.

Parameter: teeAppUUID –A byte array, UUID identifier of TAPP.

commandId – int type, command type.

inputData – A byte array, entered data.

Return: A byte array, returned data.

Exception: org.ngb.net.cas.controller.CADriverException, if there is a communication error during the interactive drive with TEE.

K.4 CAS message module

The CAS message module provides DCAS to extend the API package, and TVOS's DCAS application needs to implement this package.

The summary of CAS message module is shown in Table K.3.

Table K.3 – Summary of CAS message module

Interface	
CASEventListener	It should be implemented by applications that need to receive CAS events.
CASAppInfo	It provides information about DCAS applications.
CASEventInfo	It provides CASEvent information.
Class	
CASEventManager	The application uses CASEventManager registration listener to obtain CAS events.

K.4.1 Interface org.ngb.net.cas.event.CASEventListener

Prototype: public interface org.ngb.net.cas.event.CASEventListener

Description: This interface should be implemented by applications that need to receive CAS events. CASevents provides the CA Status and basic information of the current ServiceContext.

K.4.1.1 Method

K.4.1.1.1 receiveCASEvent

Prototype: public void receiveCASEvent(java.lang.Object serviceContext, int appId, int orgId, boolean isSuccess, int caToken)

Description: This method is used to deliver CAS events to applications registered with CAS event listeners.

Parameter: serviceContext – A java.lang.Object object, a handle to which the CAS event belongs.

appId – int type, used to identify the DCAS application sending the event. These identifiers can be applied to communicate through IXC and DCAS applications. When there is no DCAS application that can descramble a given code stream, the terminal software platform should use the value null as the value of appId to call this method, and the application that obtains this kind of CAS event notification should handle this situation according to its own design and implementation;

orgId – int type, used to identify the DCAS application that sent the event. Identify the organization to which the App belongs;

isSuccess – boolean type, a boolean value to indicate whether descrambling was successful;

caToken – int type, a token of the DCAS application sent back through the IXC, and the application can use the token to query the DCAS application for specific network information through the IXC.

Return: None.

K.4.1.1.2 receiveCASOSDEvent

Prototype: public void receiveCASOSDEvent(java.lang.Object serviceContext, int appId, int orgId, byte[] msg, int[] flag)

Description: This method is used to deliver OSD events of CAS to applications registered with CAS event listeners.

Parameter: serviceContextCAS – java.lang.Object object, a handle to which the event belongs;

appId – int type, used to identify the DCAS application that sent the event. These identifiers can be applied to communicate through IXC and DCAS applications. When there is no DCAS application that can descramble a given code stream, the terminal software platform should use the value null as the value of casAppId to call this method, and the application that obtains such a CAS event notification should handle this situation according to its own design and implementation;

orgId – int type, used to identify the DCAS application that sent the event, and identify the organization to which the App belongs.

msg – A byte array, used to deliver OSD content;

flag – An int array, used to identify the type of OSD.

Return: None.

K.4.1.1.3 receiveCASFingerEvent

Prototype: public void receiveCASFingerEvent(java.lang.Object serviceContext, int appId, int orgId, byte[] msg)

Description: This method is used to pass the CAS fingerprint event to the application registered with the CAS event listener.

Parameter: serviceContext – java.lang.Object object, a handle to which the CAS event belongs;

appId – int type, used to identify the DCAS application sending the event. These identifiers can be applied to communicate through IXC and DCAS applications. When there is no DCAS application that can descramble a given code stream, the terminal software platform should use the value null as the value of casAppId to call this method, and the application that obtains such a CAS event notification should handle this situation according to its own design and implementation;

orgId – int type, used to identify the DCAS application that sent the event. Identify the organization to which the App belongs;

msg – A byte array, used to pass content of fingerprints.

Return: None.

K.4.2 Interface org.ngb.net.cas.event.CASAppInfo

Prototype: public interface org.ngb.net.cas.event.CASAppInfo

Description: This interface provides information about DCAS applications.

K.4.2.1 Method

K.4.2.1.1 getAID

Prototype: public int getAID()

Description: This method returns Application ID of DCAS application.

Parameter: None.

Return: int type, application ID of DCAS application.

K.4.2.1.2 getOID

Prototype: public int getOID()

Description: This method returns organization ID of the DCAS application.

Parameter: None.

Return: int type, organization ID of the DCAS application.

K.4.3 Interface org.ngb.net.cas.event.CASEventInfo

Prototype: public interface org.ngb.net.cas.event.CASEventInfo

Description: This interface provides CASEvent information.

K.4.3.1 Constant field

K.4.3.1.1 TYPE_PRESENTATION

Prototype: public static final int TYPE_PRESENTATION = 0x00000001

Description: It indicates the type of TYPE_PRESENTATION.

K.4.3.1.2 TYPE_RECORDING

Prototype: public static final int TYPE_RECORDING = 0x00000002

Description: It indicates the type of TYPE_RECORDING.

K.4.3.1.3 TYPE_BUFFERING

Prototype: public static final int TYPE_BUFFERING = 0x00000004

Description: It indicates the type of TYPE_PRESENTATION.

K.4.3.2 Method

K.4.3.2.1 getType

Prototype: public int getType()

Description: This method returns type of operation that generated the CAS Event.

Parameter: None.

Return: int type, type of operation, it can be one or a combination of the values defined in this interface.

K.4.3.2.2 getNetworkInterface

Prototype: public org.davic.net.tuning.NetworkInterface getNetworkInterface()

Description: This method returns NetworkInterface related to CAS Event.

Parameter: None.

Return: one org.davic.net.tuning.NetworkInterface object.

K.4.3.2.3 getAssociatedService

Prototype: public java.lang.Object getAssociatedService()

Description: This method returns service associated with CAS Event.

Parameter: None.

Return: one Service object.

K.4.3.2.4 getServiceContext

Prototype: public java.lang.Object getServiceContext()

Description:This method returns ServiceContext associated with CAS event.

Note that ServiceContext has no practical meaning in some operations. This method returns null.

Parameter: None.

Return: one ServiceContext object.

K.4.4 Class org.ngb.net.cas.event.CASEventManager

Prototype: public class org.ngb.net.cas.event.CASEventManager

Description: The application uses CASEventManager to register a listener to obtain CAS events.

CAS events provide the current CA Status and basic information.

K.4.4.1 Method

K.4.4.1.1 getInstance

Prototype: public static org.ngb.net.cas.event.CASEventManager getInstance()

Description: This method is used to obtain a single CASEventManager instance.

Parameter: None.

Return: org.ngb.net.cas.event.CASEventManager instance.

K.4.4.1.2 addListener

Prototype: public void addListener(org.ngb.net.cas.event.CASEventListener aCASEventListener)

Description: This method is used to register a CASEventListener for the application. The listener is used to pass all CAS events.

Parameter: aCASEventListener – org.ngb.net.cas.event.CASEventListener object, CASEventListener that needs to be registered.

Return: None.

K.4.4.1.3 removeListener

Prototype: public void removeListener(org.ngb.net.cas.event.CASEventListener aCASEventListener)

Description: This method is used to unregister a CASEventListener for the application.

Parameter: aCASEventListener – org.ngb.net.cas.event.CASEventListener object, CASEventListener that has been registered.

Return: None.

K.5 CAS listener module

The CAS listener module provides a DCAS detachable security device API.

The summary of the CAS listener module is shown in Table K.4.

Table K.4 – Summary of CAS listener module

Interface	
DetachableSecurityDevice	A listener for an application to register a detachable security device to obtain a plug-in/out status of the device.
DetachableSecurityDeviceListener	A listener for the detachable safety device, should be implemented by an application that needs to listen to the plug-in/out status of the device.

K.5.1 Interface org.ngb.net.cas.detachable.DetachableSecurityDevice

Prototype: public interface org.ngb.net.cas.detachable.DetachableSecurityDevice

Description: This interface indicates a component (smart card, etc.) used to control communication with a detachable safety device.

K.5.1.1 Method

K.5.1.1.1 open

Prototype: public void open() throws org.ngb.net.cas.controller.CADriverException

Description: This method is used to initiate a session between the detachable security devices for the DCAS application.

Parameter: None.

Return: None.

Exception: org.ngb.net.cas.controller.CADriverException, if a driver error occurs, this exception is thrown.

K.5.1.1.2 close

Prototype: public void close() throws org.ngb.net.cas.controller.CADriverException

Description: This method is used to close the session between the detachable security devices for the DCAS application.

Parameter: None.

Return: None.

Exception: org.ngb.net.cas.controller.CADriverException, if a driver error occurs, this exception is thrown.

K.5.1.1.3 reset

Prototype: public byte[] reset() throws org.ngb.net.cas.controller.CADriverException

Description: This method is used to reset the detachable security device and return the data (the data is ATR in the case of a smart card).

Parameter: None.

Return: A byte array, data returned by the reset of the storage device.

Exception: An org.ngb.net.cas.controller.CADriverException, if a driver error occurs, this exception is thrown.

K.5.1.1.4 sendData

Prototype: public void sendData(byte [] data) throws org.ngb.net.cas.controller.CADriverException

Description: This method is used in the DCAS application to send data to a detachable security device.

Parameter: data – A byte array, data that needs to be sent.

Return: None.

Exception: An org.ngb.net.cas.controller.CADriverException, if a driver error occurs, this exception is thrown.

K.5.1.1.5 registerListener

Prototype: public void registerListener(org.ngb.net.cas.detachable.DetachableSecurityDeviceListener aListener)

Description: This method is used for the DCAS application to register a listener that receives data sent by a detachable safety device.

Parameter: aListener – An org.ngb.net.cas.detachable.DetachableSecurityDeviceListener to be registered.

Return: None.

K.5.1.1.6 removeListener

Prototype: public void removeListener()

Description: This method is used in DCAS applications to delete registered listeners.

Parameter: None.

Return: None.

K.5.2 Interface org.ngb.net.cas.detachable.DetachableSecurityDeviceListener

Prototype: public interface org.ngb.net.cas.detachable.DetachalbeSecurityDeviceListener

Description: This interface should be implemented by the DCAS application to receive the status of the detachable safety device and the data sent.

K.5.2.1 Field

K.5.2.1.1 DEVICE_IN

Prototype: public static final int DEVICE_IN = 1

Description: It is used to describe a state of a detachable security device: plug-in (smart card is plugged in in the case of a smart card).

K.5.2.1.2 DEVICE_OUT

Prototype: public static final int DEVICE_OUT = 2

Description: It is used to describe a state of a detachable security device: plug-out (smart card is plugged out in the case of a smart card).

K.5.2.1.3 DEVICE_ERROR

Prototype: public static final int DEVICE_ERROR = 3

Description: It is used to describe a state of a detachable security device: error (it indicates an error of smart card in the case of a smart card).

K.5.2.2 Method

K.5.2.2.1 receiveDeviceStatus

Prototype: public void receiveDeviceStatus(int status)

Description: This method should be implemented by the DCAS application to receive the state of the detachable safety device. Notify the DCAS application when the state of the detachable safety device changes.

Parameter: status – int type, the state of the detachable safety device.

Return: None.

K.5.2.2.2 receiveData

Prototype: public void receiveData(byte[] data)

Description: This method is called when the detachable security device sends data to the DCAS application.

Parameter: data – A byte array, data sent by the detachable security device.

Return: None.

Annex L

JavaScript-Unidirectional broadcast network access unit

(This annex forms an integral part of this Recommendation.)

L.1 Overview

This annex defines the JavaScript interface related to one-way broadcast network access, mainly the tuning and demodulation module.

L.2 Tuning and demodulation module

This module defines JS objects related to tuning and demodulation: DvbcTuningParameters, AbsssTuningParameters, DtmbTuningParameters, DvbTune, DvbTunerInfo, DvbScan. The structural relationship is as follows:

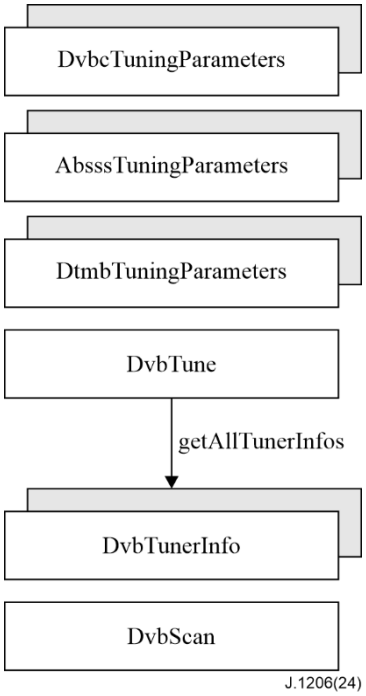


Figure L.1 – Object structure diagram of the tuning and demodulation module

L.2.1 Message

See Table L.1 for Message definitions that the tuning and demodulation module may send to the application layer.

Table L.1 – Tuning and demodulating module messages

Message name	event.which	event.modifiers	Message description
MSG_DVB_TUNE_SUCCESS	10001	number	Locking is successful, the JSON format of the message character string is: {"deliveryType":param1[Note 1], "freq":param2[Note 2]}

Table L.1 – Tuning and demodulating module messages

Message name	event.which	event.modifiers	Message description
MSG_DVB_TUNE_FAILED	10002	number	Locking failed, the JSON format of the message character string is: {"deliveryType":param1, "freq":param2}
MSG_DVB_TUNE_SIGNAL_WEAK	10003	number	The signal at the current frequency point is weak, and the JSON format of the message character string is: {"deliveryType":param1, "freq":param2}
MSG_DVB_TUNE_SIGNAL_LOST	10004	number	The signal at the current frequency point is lost, and the JSON format of the message character string is: {"deliveryType":param1, "freq":param2}
MSG_DVB_TUNE_SIGNAL_RECOVER	10005	number	The signal at the current frequency point is restored, and the JSON format of the message character string is: {"deliveryType":param1, "freq":param2}
Reserved	10006～10024	–	
MSG_DVB_SCAN_START	10025	number	The channel scan starts, and the JSON format of the message character string is: {"deliveryType":param1}
MSG_DVB_SCAN_FINISHED	10026	number	The channel scan is completed, the JSON format of the message character string is: {"deliveryType":param1}
MSG_DVB_SCAN_FAILED	10027	number	The channel scan failed, and the JSON format of the message character string is: {"deliveryType":param1} Send this message if no service is found after the channel scan is over.

Table L.1 – Tuning and demodulating module messages

Message name	event.which	event.modifiers	Message description

MSG_DVB_SCAN_FIND_SERVICES	10028	number	The channel scan finds the service, the JSON format of the message character string is:
			{
			"deliveryType": param1,
			"freq": param2,
			"serviceCount": param3
			}
			This message is sent when the current frequency point search is finished.

MSG_DVB_SCAN_STOP_SUCCESS	10029	number	Succeeded in terminating the search, the JSON format of the message character string is: {"deliveryType": param1}
MSG_DVB_SCAN_STOP_FAILED	10030	number	Failed to terminate the search, the JSON format of the message character string is: {"deliveryType": param1} (It's better to return the reason for failure)
Reserved	10031～10100	–	
MSG_DVB_SCAN_SAVE_SUCCESS	10101	–	Succeeded in saving the channel data to NVM.
MSG_DVB_SCAN_SAVE_FAILED	10102	–	Failed to save the channel data to NVM.
MSG_DVB_SCAN_REVERT_SUCCESS	10103	–	Succeeded in importing the channel data from NVM.
MSG_DVB_SCAN_REVERT_FAILED	10104	–	Failed to import the channel data from NVM.
MSG_DVB_SCAN_DELETE_SUCCESS	10105	–	Succeeded in clearing the channel data in NVM and RAM.
MSG_DVB_SCAN_DELETE_FAILED	10106	–	Failed to clear the channel data in NVM and RAM.
Reserved	10107～10200		

The value of event.modifiers is automatically given by the system, and its data type is:

– "number", indicating that the value is the ID of the message description character string, which can be obtained through the Utility.getEventInfo() method.

– "–", indicating event.modifiers is undefined.

NOTE 1 – param1: number type, see the constant definition of "Type of DVB delivery system" for the value.

NOTE 2 – param2: number type, indicating the frequency at the frequency point. If the value of deliveryType is 10 (indicating ABS-SS delivery system), it is measured in MHz; otherwise in kHz.

NOTE 3 – aram3: number type, indicating the number of services found in the frequency point specified by param2.

L.2.2 Constants

The definition of tuning and demodulation module constants is shown in Table L.2.

Table L.2 – Tuning and demodulation module constants

Constants	Description
Type of DVB delivery system	
const DVB_DELIVERY_TYPE_DVB_C = 1;	DVB-C delivery system
const DVB_DELIVERY_TYPE_ABS_SS = 10;	ABS-SS delivery system
const DVB_DELIVERY_TYPE_DTMB = 12;	DTMB delivery system
DVB-C Modulation mode	
const DVB_C_MOD_UNDEFINED = 0;	undefined
const DVB_C_MOD_QAM16 = 1;	16-QAM.
const DVB_C_MOD_QAM32 = 2;	32-QAM.
const DVB_C_MOD_QAM64 = 3;	64-QAM.
const DVB_C_MOD_QAM128 = 4;	128-QAM.
const DVB_C_MOD_QAM256 = 5;	256-QAM.
ABS-SS Polarization mode	
const ABS_SS_POLAR_LINEAR_H = 0;	Linear polarization-horizontal polarization.
const ABS_SS_POLAR_LINEAR_V = 1;	Linear polarization-vertical polarization.
const ABS_SS_POLAR_CIRCULAR_L = 2;	Circular polarization-left-hand circular polarization.
const ABS_SS_POLAR_CIRCULAR_R = 3;	Circular polarization-Right-hand circular polarization.
DTMB Modulation mode	
const DTMB_MOD_UNDEFINED = 0;	Undefined
const DTMB_MOD_QAM4 = 1;	4-QAM.
const DTMB_MOD_QAM4_NR = 2;	4-QAM-NR.
const DTMB_MOD_QAM16 = 3;	16-QAM.
const DTMB_MOD_QAM32 = 4;	32-QAM.
const DTMB_MOD_QAM64 = 5;	64-QAM.

L.2.3 DvbcTuningParameters object

DvbcTuningParameters object is a local object, which indicates the tuning and demodulation parameters applicable to the DVB-C delivery system.

Example 1:

```
//create DvbcTuningParameters object
var dvbcParams = new DvbcTuningParameters();
dvbcParams.frequency = 312000;          //312.000MHz
dvbcParams.symbol_rate = 27450;         //27.450Msymbol/s
dvbcParams.modulation = DVB_C_MOD_QAM64;    //64 QAM
```

Example 2:

```
//create DvbcTuningParameters object
var dvbcParams = new DvbcTuningParameters(312000, 27450, DVB_C_MOD_QAM64);
```

L.2.3.1 Property

The definition of the DvbcTuningParameters object property is shown in Table L.3.

NOTE – For the properties of DvbcTuningParameters, see the definition of cable_delivery_system_descriptor in GB/T 28161-2011.

Table L.3 – Properties of DvbcTuningParameters objects

Property name	Type	Properties of read and write	Description
frequency	number	Read/write	It indicates the tuning frequency of the DVB-C signal, in kHz.
symbol_rate	number	Read/write	It indicates the symbol rate of the DVB-C signal, in ksymbol/s.
modulation	number	Read/write	It indicates the DVB-C signal modulation mode.

L.2.3.2 Method

L.2.3.2.1 DvbcTuningParameters

Prototype: DvbcTuningParameters()

Description: A Construction method, creating a default DVB-C tuning and demodulation parameter object.

Parameter: None.

L.2.3.2.2 DvbcTuningParameters

Prototype: DvbcTuningParameters(frequency, symbolRate, modulation)

Description: A Construction method, creating a DVB-C tuning and demodulation parameter object according to the specified parameters.

Parameter: frequency – number type, indicating the tuning frequency of the DVB-C signal, in kHz;

symbolRate – number type, indicating the symbol rate of the DVB-C signal, in ksymbol/s;

modulation – number type, indicating the DVB-C signal modulation mode.

L.2.4 AbsssTuningParameters object

An AbsssTuningParameters object is a local object, indicating the tuning and demodulation parameters of the ABS-SS delivery system.

Example 1:

//create AbsssTuningParameters object:

var absssParams = new AbsssTuningParameters();

absssParams.frequency = 12020; //12.020GHz

absssParams.symbol_rate = 28800; //28.8Msymbol/s

absssParams.polarization = ABS_SS_POLAR_CIRCULAR_R; //Right-handed circular polarization

Example 2:

//create AbsssTuningParameters object

var absssParams = new AbsssTuningParameters(12020, 28800, ABS_SS_POLAR_CIRCULAR_R);

L.2.4.1 Properties

The definition of the AbsssTuningParameters object property is shown in Table L.4.

Table L.4 – Table of AbsssTuningParameters properties

Property name	Type	Property	Description
frequency	number	Read/write	It indicates the frequency of the ABS-SS signal, in MHz.
symbol_rate	number	Read/write	It indicates the symbol rate of the ABS-SS signal, in ksymbol/s.
polarization	number	Read/write	It indicates the polarization mode of the ABS-SS signal.

L.2.4.2 Method

L.2.4.2.1 AbsssTuningParameters

Prototype: AbsssTuningParameters()

Description: Construction method.

Parameter: None.

L.2.4.2.2 AbsssTuningParameters

Prototype: AbsssTuningParameters(frequency, symbol_rate, polarization)

Description: Construction method.

Parameter: frequency – number type, ABS-SS signal frequency, in MHz;

symbol_rate – number type, ABS-SS signal symbol rate, in ksymbol/s;

polarization– number type, indicating the polarization mode of the ABS-SS signal.

L.2.5 DtmbTuningParameters object

DtmbTuningParameters object is a local object, which indicates the tuning and demodulation parameters of the DTMB delivery system.

Example 1:

//Create DtmbTuningParameters object:

var dtmbParams = new DtmbTuningParameters();

dtmbParams.frequency = 714000; //714.000MHz

Example 2:

var dtmbParams = new DtmbTuningParameters(714000);

L.2.5.1 Properties

The definition of the DtmbTuningParameters object property is shown in Table L.5.

NOTE – For the DtmbTuningParameters property, please refer to the definition of terrestrial_delivery_system_descriptor in GB/T 28161-2011.

Table L.5 – Table of DtmbTuningParameters properties

Property name	Type	Property	Description
frequency	number	Read/write	It indicates the center frequency of the DTMB signal, in kHz.
modulation	number	Read only	It indicates the modulation mode of DTMB signal.
codingRatio	string	Read only	It indicates the coding efficiency of DTMB signal, which can be "0.4", "0.6" or "0.8".
PNMode	string	Read only	It indicates the frame header mode of DTMB signal, which can be a value such as "PN945". This property is meaningful only after successful tuning and demodulation.

L.2.5.2 Method

L.2.5.2.1 DtmbTuningParameters

Prototype: DtmbTuningParameters()

Description: A Construction method.

Parameter: None.

L.2.5.2.2 DtmbTuningParameters

Prototype: DtmbTuningParameters(frequency)

Description: A Construction method.

Parameter: frequency – number type, indicating the center frequency of DTMB signal, in kHz;

L.2.6 DvbTune object

DvbTune object is a built-in object, which realizes channel tuning and signal demodulation. Two types of objects are defined. NGBDvbTune() defaults tunerId to 0, and NGBDvbTune(tunerId) specifies the tunerId when creating the object.

L.2.6.1 Method

L.2.6.1.1 tune

Prototype: tune(deliveryType, paramsObj)

Description: An Asynchronous method, tuning to the specified frequency point.

– If the frequency lock is successful, the message MSG_DVB_TUNE_SUCCESS will be sent to the page;
– If the frequency lock fails, the message MSG_DVB_TUNE_FAILED will be sent to the page;
– If the signal is weak, the message MSG_DVB_TUNE_SIGNAL_WEAK will be sent to the page;
– If the signal is lost, the message MSG_DVB_TUNE_SIGNAL_LOST will be sent to the page;
– If the signal is restored, the message MSG_DVB_TUNE_SIGNAL_RECOVER will be sent to the page.

Parameter: deliveryType– number type, indicating the type of paramsObj object;

paramsObj – it indicates tuning and demodulation parameters, the type is specified by the deliveryType parameter:

– DVB_DELIVERY_TYPE_DVB_S – The type of paramsObj is DvbsTuningParameters;

– DVB_DELIVERY_TYPE_DVB_C – The type of paramsObj is DvbcTuningParameters;

– DVB_DELIVERY_TYPE_DVB_T – The type of paramsObj is DvbtTuningParameters;

– DVB_DELIVERY_TYPE_ABS_SS – The type of paramsObj is AbsssTuningParameters;

– DVB_DELIVERY_TYPE_DTMB – The type of paramsObj is DtmbTuningParameters.

Return: None.

L.2.6.1.2 getTunerSignalInfo

Prototype: string getTunerSignalInfo()

Description: A Synchronous method, acquirngTuner information, including signal quality, signal strength, bit error rate, signal level, and signal-to-noise ratio.

Parameter: None.

Return:

– If the acquisition succeeds, it will return a description character string in the form of "{SignalQuality:XXX,signalStrength:XXX,errorRate:XXX,signalLevel:XXX,signalNoiseRatio:XXX}".

– If the acquisition fails, it will return null.

L.2.6.1.3 getAllTunerInfos

Prototype: DvbTunerInfo[] getAllTunerInfos()

Description: Getting all Tuner information supported by the current system.

Parameter: None.

Return: A DvbTunerInfo object array.

L.2.7 DvbTunerInfo object

The DvbTunerInfo object is a local object, used to describe the matching relationship between TunerId and Tuner types.

L.2.7.1 Properties

The definition of the DvbTunerInfo object property is shown in Table L.6.

Table L.6 – DvbTunerInfo object property

Property name	Type	Read/write property	Description
tunerId	number	Read/write	It indicates the Tuner ID value.
tunerType	number	Read/write	It indicates the Tuner type corresponding to the Tuner ID.

L.2.8 DvbScan object

DvbScan object is a built-in object, which implements channel scan.

Channel scan method definition:

Manual search: According to the set tuning and demodulation parameters, searching for broadcast and TV program channels within a single frequency point;

Automatic search: Searching for NIT according to the starting channel specified by the operator, and then automatically search for broadcast and TV program channels in the entire network according to the instructions of the NIT; the operator may specify multiple starting channels, as long as the NIT is successfully found at any one of the frequency points, it can perform automatic search.

Zone search: According to china's digital television channel assignment table, searching for radio and television programs frequency by frequency within the specified starting and ending frequency range.

This specification does not mandate the storage mechanism of channel scan results in the receiving terminal. To help understand the channel data access method provided by the DvbScan object, this specification gives examples to illustrate the storage mechanism that the receiving terminal may use.

Example: After the channel scan is completed, the PSI/SI data (except for the EIT table) will be stored in three storage areas, namely A, B and C. The division of storage areas is shown in Figure L.2.

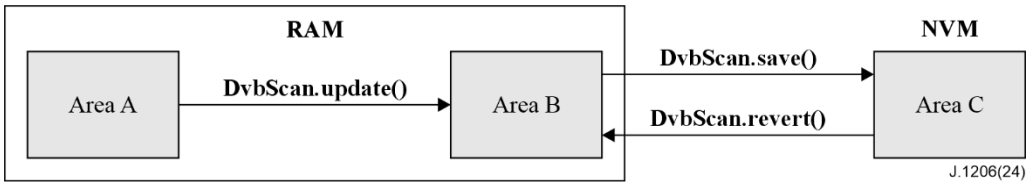


Figure L.2 – Schematic diagram of the storage area of channel scan results

The storage information in the NVM area is not lost when power is off. This type of storage includes Flash, E2PROM, and disk. The storage area C is assigned in the NVM to store the channel data successfully found by the receiving terminal.

When the receiving terminal is in the power-off state, the PSI/SI data is only stored in the NVM. When the receiving terminal is powered on, the system will automatically import the PSI/SI data from the area C to the area B in the RAM for applications to call.

In the process of searching channels, the newly searched data will be temporarily saved in Area A. After the search is completed and confirmed by the user, the application calls the DvbScan.update() method to update the newly searched data to area B. The application calls the DvbScan.save() method to save the data in area B to area C, otherwise the search results will be lost after power off. If the application wants to abandon the newly searched results, it can call the DvbScan.revert() method to restore the old data in the area C to the area B.

DvbScan has two construction methods, DvbScan() uses tunerId as 0 by default, and DvbScan(tunerId) can specify the tunerId value when searching.

L.2.8.1 Method

L.2.8.1.1 startScan

Prototype: startScan(scanType, deliveryType, objArray[])

Description: An Asynchronous method, starting channel scan, wherein search mode is determined by scanType parameter, the system will perform tuning and demodulation automatically.

– When the search starts, the message MSG_DVB_SCAN_START will be sent to the page;
– After the current frequency point search is completed, if the service is found, the message MSG_DVB_SCAN_FIND_SERVICES will be sent to the page;
– After all the channels are searched, the message MSG_DVB_SCAN_FINISHED will be sent to the page;
– If no channel is found, the message MSG_DVB_SCAN_FAILED will be sent to the page.

Parameter: scanType – number type, indicating the channel scan method, the value is as follows:

– 0-indicating manual search, the length of the parameter objArray array is 1, indicating the tuning and demodulation parameters of the frequency to be searched;

– 1-indicating automatic search, the parameter objArray array indicating the starting channel tuning and demodulation parameters, the length of the array is greater than or equal to 1; as long as the NIT is successfully analyzed at one of the frequency points, the search can be automatically completed according to the instructions of the NIT, and the other frequency point parameters of the array can be ignored;

– 2-indicating zone search, the length of the parameter objArray array is 2, objArray[0] indicating the tuning and demodulation parameters at the beginning of the interval, and objArray[1] indicating the tuning and demodulation parameters at the end of the interval; the interval settings should comply with relevant national regulations.

deliveryType – number type, indicating the type of the objArray object array.

objArray[] − indicating tuning and demodulation parameters, the type is specified by the deliveryType parameter;

– DVB_DELIVERY_TYPE_DVB_S– the type of objArray is 4er;

– DVB_DELIVERY_TYPE_DVB_C– the type of objArray is DvbcTuningParameters;

– DVB_DELIVERY_TYPE_DVB_T– the type of objArray is DvbtTuningParameters;

– DVB_DELIVERY_TYPE_ABS_SS – the type of objArray is AbsssTuningParameters;

– DVB_DELIVERY_TYPE_DTMB – the type of objArray is DtmbTuningParameters.

Return: None.

L.2.8.1.2 startScan

Prototype: startScan(pid, tableid, deliveryType, objArray[])

Description: An Asynchronous method, starting channel scan, wherein the search mode is determined by scanType parameter, the system will perform tuning and demodulation automatically.

– When the search starts, the message MSG_DVB_SCAN_START will be sent to the page

– After the search is completed, the message MSG_DVB_SCAN_FINISHED will be sent to the page;

– If no channel is found, the message MSG_DVB_SCAN_FAILED will be sent to the page.

Parameter: pid − number type, indicating the PID of the TS packet where the specified program information data needs to be searched;

tableid − number type, indicating the need to search for the tableid assigned by the specified program information data;

deliveryType – number type, indicating the type of objArray object array.

objArray[] – indicating tuning and demodulation parameters, the type being specified by the deliveryType parameter;

– DVB_DELIVERY_TYPE_DVB_S– the type of objArray is 4er;

– DVB_DELIVERY_TYPE_DVB_C– the type of objArray is DvbcTuningParameters;

– DVB_DELIVERY_TYPE_DVB_T– the type of objArray is DvbtTuningParameters;

– DVB_DELIVERY_TYPE_ABS_SS – the type of objArray is AbsssTuningParameters;

– DVB_DELIVERY_TYPE_DTMB – the type of objArray is DtmbTuningParameters.

Return: None.

L.2.8.1.3 startScan

Prototype: startScan(string jsonSIInfo)

Description: Transferring the program information data in JSON format to the DTV component for analysis and storage.

Since it is only data analysis and does not consume too much time, the synchronous mode is adopted.

Parameter: jsonSIInfo – string type, applying the PSI/SI information in JSON format obtained from the operator's front-end. The detailed field definitions of JSON information are shown in Table L.7, Table L.8, Table L.9, Table L.10, Table L.11, and Table L.12, Table L.13, Table L.14, Table L.15 and Table L.16.

Return: 0 – success;

-1 – data format error.

Table L.7 – Data structure definition of configuration table program information

Field	Type	Description
Version	Number	The version of the program data is used to ensure that the terminal and the front-end programs are consistent. The initial value of the front end is 1, and 1 is added to the value when the program data has any change, and it is consistent with the version in the NIT table.
OperatorID	Number	Operator number
NetWorkID	Number	Network ID
OperatorNames	Array	Operator name
DeliveryInfos	Object	Delivery stream information, frequency point information in the network.
Services	Array	Program information.
Groups	Array	Program grouping information.

Table L.8 – Data structure definition of operator name information

Field	Type	Description
lang	String	Language type.
Name	String	Operator name.

Table L.9 – Data structure definition of delivery information

Field	Type	Description
DeliveryType	Number	Delivery type, 1: DVB-C, 2: DTMB-T, 3: ABS-S.
TSNumb	Number	Number of delivery streams.
Translates	Object	Frequency point information.

Table L.10 – Data structure definition of frequency point information

Field	Type	Description
TsIndex	Number	Sequence number of delivery stream.
Name	String	Name of delivery stream.
OriginalNetworkID	Number	Original network ID.
TSId	Number	Delivery stream ID.
frequency	Number	Frequency, in kHz.
symboRate	Number	Symbol rate, in kbit/s.
Modulation	Number	Modulation mode, see SI definition: 0, reserved 1:16QAM, 2:32QAM, 3:64QAM 4:128QAM, 5:256QAM.

Table L.11 – Data structure definition of program information

Field	Type	Description
OriginalNetworkID	Number	Original network ID.
TSId	Number	Delivery stream ID.
ServerID	Number	Server ID.
TSId	Number	Delivery stream ID.
RegionCode	Array	Region code.
PcrPid	Number	PCR PID.
ServiceNames	Array	Program name (multi-encoding format name).
PMTPid	Number	Pmt Pid, keep consistent with the description in the SDT table.
ServiceType	Number	Program type, see SI definition.
logicNo	Number	Logical channel number.
SoundChannel	Number	0: Stereo, 1: Left channel, 2: Right channel, 3: Mono channel.
VolumeOffset	Number	Volume compensation value. The value range is negative 5 to positive 5.
Video	Object	Video stream information.
Audios	Array	Audio stream array.
OtherES	Array	Other data flow information.

Table L.12 – Data structure definition of video stream information

Field	Type	Description
VideoPid	Number	Video stream PID.
VideoType	Number	Video stream encoding type.
CAInfo	Array	Conditional access information.

Table L.13 – Data structure definition of audio stream information

Field	Type	Description
AudioLang	String	Language type.
AudioPid	Number	Audio PID.
AudioType	Number	Audio stream encoding type.
CAInfo	Array	Conditional access information.

Table L.14 – Data structure definition of other elementary stream information

Field	Type	Description
ESType	Number	Type of elementary stream.
Pid	Number	PID.
CAInfo	Array	Conditional access information.

Table L.15 – Data structure definition of conditional access information

Field	Type	Description
CASystemID	Number	Supplier ID of CA system.
EcmPid	Number	PID of control word.

Table L.16 – Data structure definition of program grouping information

Field	Type	Description
Names	Array	Information of group name.
GroupID	Number	Nummer of group name.
GroupServices	Array	List of program numbers in the group.

Example:

TVOS-SI


```
{
        "Version":0001,
"OperatorID":3356,
"NetWorkID":1234,
        "OperatorNames ":[
{ "Lang":"chi",
            "Name":" Jilin Broadcasting",
            },
],
        "DeliveryInfo":{
            "DeliveryType":1,
            "TSNumb": 25,
            "Translates":[
```

```
                "TsIndex":1,
                "Name":"ss"
                "OriginalNetworkID":223,
                "TSId":123,
                "frequency":256000,
                "symboRate":6875,
                "Modulation":"64QAM",
            ],
        },
/*      "DeliveryInfo":{
            "DeliveryType":"ABS-S",
            "TSNumb": xxx,
            "Translates":[
                "TsIndex":1,
                "Name":"ss"
                "OriginalNetworkID":223,
                "TSId":123,
                "Polarisation":00
                "frequency":300000,
                "symboRate":6875,
            ],
*/
        "Services":[
"OrgNetWorkID":223,
"TSID":123
            "ServerID":XXX,
            "RegionCode":[xxx,xxx],
            "PcrPid":2011,
            "ServiceNames":[
                { "lang":"chi",
                    "ServiceName":" Jilin Satellite TV ",
                    "ProviderName":"Jilin TV Station",
                    },
                { "lang":"eng",
                    "ServiceName":"jilinweishi",
                    "ProviderName":"jilin",
                },
```

```
],
"PMTPid":xxx,
"ServiceType":1,
"logicNo":xxx,
"SoundChannel":xxx,
"VolumeOffset":xxx,
"Video":{
    "VideoPid":xxx,
    "VideoType":1,
    "CAInfo":[
        "CASystemID":XXX
        "EcmPid":xxx,
    ],
},
"Audios":[
    {"AudioLang":"ch",
        "AudioPid":xxx,
        "AudioType":1,
        "CAInfo":[
            "CASystemID":XXX
            "EcmPid":xxx,
        ],
    },
    {"AudioLang":"eng",
        "AudioPid":xxx,
        "AudioType":1,
        "CAInfo":[
            "CASystemID":XXX
            "EcmPid":xxx,
        ],
    },
],
"OtherES":[
    "ESType":xx,
    "Pid":xxx,
    "CAInfo":[
        "CASystemID":XXX
```

```
                    "EcmPid":xxx,
                    ],
                ],
            ],
            "Groups":[
             {"Names":[
{ "lang":"chi",
                        "name":"HD",
},
]
                "GroupID":xxx
                "GroupServices":[
                        "ServiceID":123,
                        "ServiceID":125,
                    ]
                },
            ],
}
```

L.2.8.1.4 stopScan

Prototype: stopScan()

Description: An Asynchronous method, terminating the channel scan.

– If the termination succeeds, the message MSG_DVB_SCAN_STOP_SUCCESS is sent to the page;

– If the termination fails, the message MSG_DVB_SCAN_STOP_FAILED is sent to the page.

Parameter: None.

Return: None.

L.2.8.1.5 update

Prototype: number update()

Description: Update PSI/SI data.

Example: Use the PSI/SI data that has been successfully found in the area A to update the corresponding data in the area B, and the other data in the area B remain unchanged.

Parameter: None.

Return: number type, 1 indicating the update was successful, the update failed.

L.2.8.1.6 save

Prototype: save()

Description: An Asynchronous method, saving PSI/SI data to NVM. After the search is completed, call this method to update the data in the NVM, otherwise the original channel data will be maintained after restarting the receiving terminal.

– If the save succeeds, the message MSG_DVB_SCAN_SAVE_SUCCESS will be sent to the page;
– If the save fails, the message MSG_DVB_SCAN_SAVE_FAILED will be sent to the page.

Example: Saving the PSI/SI data in area B of RAM to area C in NVM.

Parameter: None.

Return: None.

L.2.8.1.7 revert

Prototype: revert()

Description: An Asynchronous method, importing PSI/SI data from NVM to RAM.

– If the import succeeds, the message MSG_DVB_SCAN_REVERT_SUCCESS will be sent to the page;
– If the import fails, MSG_DVB_SCAN_REVERT_FAILED will be sent to the page.

Example: After deleting all the data in the area B, reimport the PSI/SI data from the area C in the NVM.

Parameter: None.

Return: None.

L.2.8.1.8 deleteAll

Prototype: deleteAll()

Description: An Asynchronous method, clearing PSI/SI data in RAM and NVM.

– If the clearing succeeds, the message MSG_DVB_SCAN_DELETE_SUCCESS will be sent to the page;
– If the clearing fails, the message MSG_DVB_SCAN_DELETE_FAILED will be sent to the page.

Example: Clear PSI/SI data in area B and area C.

Parameter: None.

Return: None.

Annex M

JavaScript-Broadcast protocol processing unit

(This annex forms an integral part of this Recommendation.)

M.1 Overview

This annex defines JavaScript interfaces related to broadcast protocol processing, mainly including DVB protocol modules.

M.2 DVB protocol processing module

This module defines JS objects related to DVB broadcast protocol processing: DvbBroadcast, DvbNetwork, DvbBouquet, DvbTS, DvbService, DvbVideoES, DvbAudioES, DvbOtherES, the structure relationship is shown in Figure M.1.

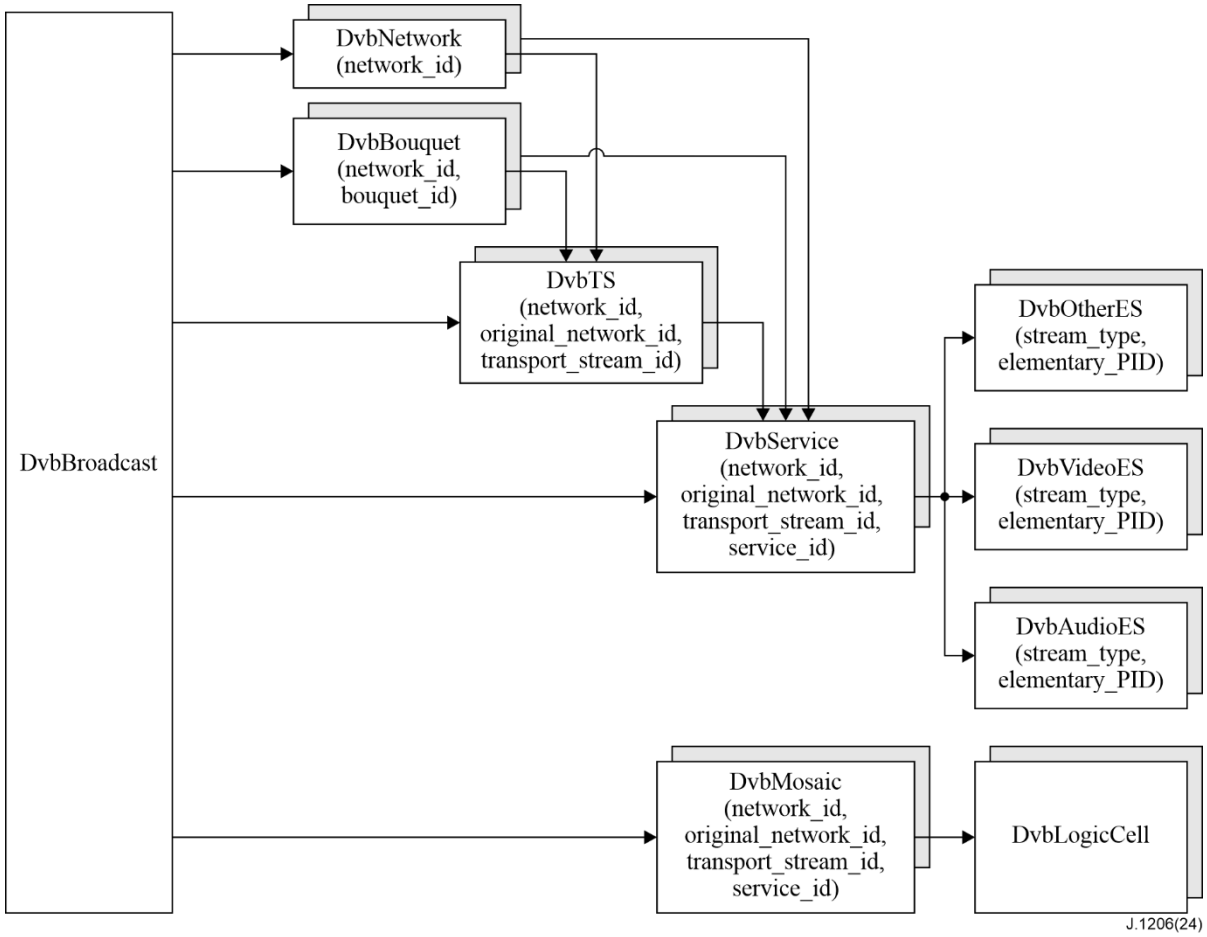


Figure M.1 – Structure diagram of DVB protocol processing module object

M.2.1 Message

The message definition that the DVB protocol processing module may send to the application layer is shown in Table M.1.

Table M.1 – Message definition of DVB protocol processing module

Message	event.which	event.modifiers	Description of message
MSG_DVB_NIT_CHANGE	11001	number	NIT version changes, the JSON format of the message character string is: {"oldVersion":param1, "newVersion":param2}
MSG_DVB_PAT_CHANGE	11002	number	PAT version changes, the JSON format of the message character string is: {"oldVersion":param1, "newVersion":param2}
Reserved	11003～11200	–	
The value of event.modifiers is automatically given by the system, and its data type: – "number", indicating that the value is the ID of the message description character string, which can be obtained through the Utility.getEventInfo() method. If the "message description" defines the JSON format of the message character string, the message content will be retrieved according to the format. – "–", indicating that event.modifiers is undefined.			

M.2.2 Constants

The constant definition of the DVB protocol processing module is shown in Table M.2.

Table M.2 – DVB protocol processing module constants

Constants	Description
Service type	
const DVB_SERVICE_TYPE_DTV = 0x01;	Digital TV Broadcasting Service
const DVB_SERVICE_TYPE_DAB = 0x02;	Digital Sound Broadcasting Service
const DVB_SERVICE_TYPE_TELETEXT = 0x03;	Teletext Service
const DVB_SERVICE_TYPE_NVOD_REF = 0x04;	NVOD Reference Service
const DVB_SERVICE_TYPE_NVOD_SHIFT = 0x05;	NVOD Time-shifted Service
const DVB_SERVICE_TYPE_MOSAIC = 0x06;	Mosaic Service
const DVB_SERVICE_TYPE_DATA = 0x0C;	Data Broadcasting Service
const DVB_SERVICE_TYPE_AVC_SD = 0x16;	Advanced coding standard definition digital TV (H.264)
const DVB_SERVICE_TYPE_AVC_SD_NVOD_SHIFT = 0x17;	Advanced coding standard definition NVOD time shift service (H.264)
const DVB_SERVICE_TYPE_AVC_SD_NVOD_REF = 0x18;	Advanced coding standard definition NVOD reference service (H.264)
const DVB_SERVICE_TYPE_AVC_HD = 0x19;	Advanced coding high-definition digital TV (H.264)
const DVB_SERVICE_TYPE_AVC_HD_NVOD_SHIFT = 0x1A;	Advanced encoding high-definition NVOD time shift service (H.264)

Table M.2 – DVB protocol processing module constants

Constants	Description
const DVB_SERVICE_TYPE_AVC_HD_NVOD_REF = 0x1B;	Advanced coding high-definition NVOD reference service (H.264)
const DVB_SERVICE_TYPE_AVC_3DHD = 0x1C;	Advanced encoding HD frame compatible with 3D digital TV (H.264)
const DVB_SERVICE_TYPE_AVC_3DHD_SHIFT = 0x1D;	Advanced encoding HD frame compatible with 3D NVOD time shift service (H.264)
const DVB_SERVICE_TYPE_AVC_3DHD_REF = 0x1E;	Advanced coding HD frame compatible with 3D NVOD reference service (H.264)
NOTE – The service type constant definition is quoted from GB/T 28161-2011 Table 70 "Service Type Coding" and ETSI EN 300 468 v1.12.1 (2011-10) Table 87 "Service type coding".	
Sorting basis	
const DVB_SORT_TYPE_NETWORK_ID = 0x01;	Sort by network_id.
const DVB_SORT_TYPE_BOUQUET_ID = 0x02;	Sort by bouquet_id.
const DVB_SORT_TYPE_ONET_ID = 0x03;	Sort by original_network_id.
const DVB_SORT_TYPE_TS_ID = 0x04;	Sort by transport_stream_id.
const DVB_SORT_TYPE_SERVICE_ID = 0x05;	Sort by service_id.
const DVB_SORT_TYPE_SERVICE_TYPE = 0x06;	Sort by service_type.
const DVB_SORT_TYPE_SERVICE_NAME = 0x07;	Sort by service_name.
const DVB_SORT_TYPE_CHANNEL_ID = 0x10;	Sort by user channel number.
const DVB_SORT_TYPE_LOGICAL_ID = 0x11;	Sort by logical channel number.
const DVB_SORT_TYPE_FTA_SCR = 0x20;	Sort according to whether it is encrypted or not.
Sorting method	
const DVB_SORT_ORDER_NONE = 0x00;	Do not sort.
const DVB_SORT_ORDER_ASC = 0x01;	– If the sorting basis is DVB_SORT_TYPE_FTA_SCR, it means non-encrypted to encrypted arrangement; – If the sorting basis is other, it means sorting in ascending order.
const DVB_SORT_ORDER_DESC = 0x02;	– If the sorting basis is DVB_SORT_TYPE_FTA_SCR, it means encrypted to non-encrypted sorting; – If the sorting basis is other, it means sorting in descending order.

M.2.3 DvbBroadcast object

The DvbBroadcast object is a built-in object, which provides a method for obtaining PSI/SI information under the DVB technology system.

M.2.3.1 Property

The definition of the DvbBroadcast object property is shown in Table M.3.

Table M.3 – DvbBroadcast object properties

Property name	Type	Read/write property	Description
currentDvbNetwork	DvbNetwork	Read only	It indicates the current DvbNetwork object.
currentDvbBouquets	DvbBouquet array	Read only	It indicates the current array of DvbBouquet objects.
currentDvbTS	DvbTS	Read only	It indicates the current DvbTS object.
currentDvbService	DvbService	Read only	It indicates the current DvbService object.
currentDvbMosaic	DvbMosaic	Read only	It indicates the current DvbMosaic object.

M.2.3.2 Method

M.2.3.2.1 getAllNetworks

Prototype: DvbNetwork[] getAllNetworks()

Description: Getting all unidirectional broadcast network objects that the current receiving terminal can access, without sorting.

Parameter: None.

Return: An array of DvbNetwork objects.

M.2.3.2.2 getAllNetworks

Prototype: DvbNetwork[] getAllNetworks(sortOrder)

Description: Getting all unidirectional broadcast network objects that the current receiving terminal can access, and sorting them in a specified way.

Parameter: sortOrder – number type, indicating the sorting method. The value is shown in the constant definition of "sorting method" in Table M.2.

NOTE – It can only be sorted based on network_id, that is, the sorting basis is DVB_SORT_TYPE_NETWORK_ID, see Table M.2 "Sorting basis" constant.

Return: An array of DvbNetwork objects.

M.2.3.2.3 getNetwork

Prototype: DvbNetwork getNetwork(network_id)

Description: Getting the specified one-way broadcast network object.

Parameter: network_id – number type, indicating network ID.

Return: DvbNetwork object. If the specified object does not exist, null is returned.

M.2.3.2.4 getAllBouquets

Prototype: DvbBouquet[] getAllBouquets()

Description: Getting all one-way broadcast service group objects without sorting.

Parameter: None.

Return: An array of DvbBouquet objects. If there is no service group object (for example, the BAT table is not found in the stream), the length of the returned array is 0.

M.2.3.2.5 getAllBouquets

Prototype: DvbBouquet[] getAllBouquets(sortTypeArray[], sortOrderArray[])

Description: Getting all one-way broadcast service group objects, sort them in a specified way.

Parameter: sortTypeArray[] – an array of number type, indicating the sorting basis of DvbBouquet objects. There are at most two sorting basis, but cannot be repeated. The priority of the sorting basis is related to the order of the array members. The smaller the subscript of the array member, the higher the priority; each member of the array can be DVB_SORT_TYPE_NETWORK_ID and DVB_SORT_TYPE_BOUQUET_ID, see Table M.2 in the "sorting basis" constant.

sortOrderArray[] – an array of number type, indicating the sorting method. The length of the array is consistent with the sortTypeArray parameter. The possible values of each member of the array are shown in Table M.2 "Sorting Method" constant definition. The sortOrderArray and sortTypeArray must correspond one-to-one, that is, sortOrderArray[i] is only applicable to sortTypeArray[i].

Return: An array of DvbBouquet objects. If there is no service group object (for example, the BAT table is not found in the stream), the length of the returned array is 0.

M.2.3.2.6 getBouquet

Prototype: DvbBouquet getBouquet(network_id, bouquet_id)

Description: Getting the specified one-way broadcast service group object.

Parameter: network_id – number type, indicating the network ID;

bouquet_id – number type, indicating service group ID.

Return: DvbBouquet objects. If the specified object does not exist, null is returned.

M.2.3.2.7 getAllTSs

Prototype: DvbTS[] getAllTSs()

Description: Getting all unidirectional broadcast delivery stream objects without sorting.

Parameter: None.

Return: An array of DvbTS objects.

M.2.3.2.8 getAllTSs

Prototype: DvbTS[] getAllTSs(sortTypeArray[], sortOrderArray[])

Description: Getting all unidirectional broadcast delivery stream objects, sort them in a specified way.

Parameter: sortTypeArray[] – an array of number type, indicating the sorting basis of DvbTS objects. There are at most three sorting bases, but cannot be repeated. The priority of sorting basis is related to the order of the array members. The smaller the subscript of the array member, the higher the priority. Each member of the array can take the values DVB_SORT_TYPE_NETWORK_ID, DVB_SORT_TYPE_ONET_ID and DVB_SORT_TYPE_TS_ID, as shown in Table M.2 "Sorting basis" constant definition.

sortOrderArray[] – an array of number type, indicating the sorting method. The length of the array is the same as the sortTypeArray parameter. For the value of each member of the array, see the constant definition of "Sorting method" in Table M.2". The sortOrderArray and sortTypeArray must correspond one-to-one, that is, sortOrderArray[i] is only applicable to sortTypeArray[i].

Return: An array of DvbTS objects.

Example:

```
//Sort DvbTS objects in ascending order by original_network_id, and then sort them in descending order by transport_stream_id
getAllTSs([DVB_SORT_TYPE_ONET_ID, DVB_SORT_TYPE_TS_ID],
[DVB_SORT_ORDER_ASC,DVB_SORT_ORDER_DESC]);
```

M.2.3.2.9 getTS

Prototype: DvbTS getTS(network_id, original_network_id, transport_stream_id)

Description: Getting the specified unidirectional broadcast delivery stream object.

Parameter: network_id – number type, indicating the network ID.

original_network_id – number type, indicating the original network ID;

transport_stream_id – number type, indicating the delivery stream ID.

Return: A DvbTS object. If the specified delivery stream object does not exist, null is returned.

M.2.3.2.10 getAllServices

Prototype: DvbService[] getAllServices()

Description: Getting all one-way broadcast service objects without sorting.

Parameter: None.

Return: An array of DvbService objects.

M.2.3.2.11 getAllServices

Prototype: DvbService[] getAllServices(sortTypeArray[], sortOrderArray[])

Description: Getting all one-way broadcast service objects and sort them in a specified way.

Parameter: sortTypeArray[] – an array of number type, indicating the sorting basis of DvbService objects. There can be one or more sorting basises, but there cannot be repeated ones. The priority of the sorting basis is related to the order of the array members. The smaller the subscript of the array member, the higher the priority. For the value of each member of the array, see the constant definition of "Sorting basis" in Table M.2". The sortOrderArray and sortTypeArray must correspond one-to-one, that is, sortOrderArray[i] is only applicable to sortTypeArray[i].

Return: An array of DvbService objects.

Example:

```
//Sort the DvbService objects in ascending order by original_network_id, then sort them in descending order by transport_stream_id, and then sort them in ascending order by service_id
getAllServices([DVB_SORT_TYPE_ONET_ID, DVB_SORT_TYPE_TS_ID, DVB_SORT_TYPE_SERVICE_ID],
[DVB_SORT_ORDER_ASC,DVB_SORT_ORDER_DESC,DVB_SORT_ORDER_ASC]);
```

M.2.3.2.12 getService

Prototype: DvbService getService (network_id, original_network_id, transport_stream_id, service_id)

Description: Getting the specified one-way broadcast service object.

Parameter: network_id – number type, indicating the network ID;

original_network_id – number type, indicating the original network ID;

transport_stream_id – number type, indicating the delivery stream ID;

service_id – number type, indicating the service ID.

Return: DvbService object. If the specified object does not exist, null is returned.

M.2.3.2.13 getAllMosaics

Prototype: DvbMosaic[] getAllMosaics()

Description: Getting all mosaic objects without sorting.

Parameter: None.

Return: An array of DvbMosaic objects. If there is no mosaic object, the length of the returned array is 0.

M.2.3.2.14 getEntryMosaic

Prototype: DvbMosaic getEntryMosaic(network_id)

Description: Getting the entrance mosaic object of the specified network.

Parameter: network_id – number type, indicating the network ID.

Return: A DvbMosaic object. If there is no mosaic object, null is returned.

M.2.4 DvbNetwork object

The DvbNetwork object is a local object, which is used to describe the relevant information of the unidirectional broadcast network and is uniquely identified by network_id.

Example 1:

//Getting the DvbNetwork object through the properties of the DvbBroadcast object

var dvbNetwork = DvbBroadcast.currentDvbNetwork;

Example 2:

//Get an array of DvbNetwork objects through the method of the DvbBroadcast object, and then get a single DvbNetwork object

var dvbNetworkArray[] = DvbBroadcast.getAllNetworks(sortOrder);

var dvbNetwork = dvbNetworkArray[i];

Example 3:

//Getting the specified DvbNetwork object through the method of the DvbBroadcast object

var dvbNetwork = DvbBroadcast.getNetwork(network_id);

M.2.4.1 Property

The definition of the DvbNetwork object property is shown in Table M.4.

Table M.4 – DvbNetwork object property

Property name	Type	Read/write property	Description
network_id	number	Read only	It indicates the network identifier of the one-way broadcast network, which is used to distinguish other one-way broadcast networks.
network_name	string	Read only	It indicates the network name of the one-way broadcast network.
network_type	number	Read only	It indicates the delivery type of a one-way broadcast network.

M.2.4.2 Method

M.2.4.2.1 getNetworkName

Prototype: string getNetworkName()

Description: Getting the full name of the network. The full name of the network is obtained from descriptors of the network_name_descriptor or from the multilingual_network_name_descriptor. This interface should return the full name of the network that matches the preferred language set by the user. If the full name of the network with the language encoding cannot be obtained from the descriptors of network_name_descriptor or multilingual_network_name_descriptor, it will return the full name of the network carried in the network_name_descriptor by default.

Parameter: None.

Return: string type, indicating the full name of the network.

M.2.4.2.2 getShortNetworkName

Prototype: string getShortNetworkName()

Description: Getting the abbreviation of the network name.

Parameter: None.

Return: string type, indicating the abbreviation of the network name, if there is no network name abbreviation, undefined is returned.

Example:"[0x86]Asterix[0x87] Digital Satellite TV Network"

Full name of the network:"Asterix Digital Satellite TV Network".

Abbreviation of the network name:"Asterix".

M.2.4.2.3 getAllTSs

Prototype: DvbTS[] getAllTSs()

Description: Getting all DvbTS objects in the current network object without sorting.

Parameter: None.

Return: An array of DvbTS objects.

M.2.4.2.4 getAllTSs

Prototype: DvbTS[] getAllTSs(sortTypeArray[], sortOrderArray[])

Description: Getting all DvbTS objects in the current network object and sort them in a specified way.

Parameter: sortTypeArray – An array of number type, indicating the sorting basis. There can be one or more sorting basises, but they cannot be repeated. The priority of the sorting basis is related to the order of the array members. The smaller the subscript of the array member, the higher the priority.

For the value of each member of the array, see the constant definition of "Sorting basis" in Table M.2".

sortOrderArray – An array of number type, indicating the sorting method. The length of the array is the same as the sortTypeArray parameter. For the value of each member of the array, see the constant definition of "Sorting method" in Table M.2. The sortOrderArray and sortTypeArray must correspond one-to-one, that is, sortOrderArray[i] is only applicable to sortTypeArray[i].

Return: An array of DvbTS objects.

Example:

```
//Sort DvbTS objects in ascending order by original_network_id, and then sort them in descending order by transport_stream_id
getAllTSs([DVB_SORT_TYPE_ONET_ID, DVB_SORT_TYPE_TS_ID],
    [DVB_SORT_ORDER_ASC,DVB_SORT_ORDER_DESC]);
```

M.2.4.2.5 getTS

Prototype: DvbTS getTS(original_network_id, transport_stream_id)

Description: Getting the specified DvbTS object in the current network object.

Parameter: original_network_id – number type, indicating the original network ID;

transport_stream_id – number type, indicating the delivery stream ID.

Return: DvbTS object, if the specified object does not exist, it returns null.

M.2.4.2.6 getAllServices

Prototype: DvbService[] getAllServices()

Description: Getting all unidirectional broadcast service objects in the current network objects, without sorting.

Parameter: None.

Return: An array of DvbService objects.

M.2.4.2.7 getAllServices

Prototype: DvbService[] getAllServices(sortTypeArray[], sortOrderArray[])

Description: Getting all one-way broadcast service objects in the current network objects, and sort them in a specified way.

Parameter: sortTypeArray[] – array of number type, indicating the sorting basis of DvbService objects. There can be one or more sorting basises, but they cannot be repeated. The priority of the sorting basis is related to the order of the array members. The smaller the subscript of the array member, the higher the priority. For the value of each member of the array, see the constant of "Sorting basis" in Table M.2".

sortOrderArray[] – An array of number type, indicating the sorting method of DvbService objects. The length of the array is the same as the sortTypeArray parameter. For the value of each member of the array, see the constant of "Sorting method" in Table M.2. The sortOrderArray and sortTypeArray must correspond one-to-one, that is, sortOrderArray[i] is only applicable to sortTypeArray[i].

Return: An array of DvbService objects.

Example:

```
//Sort the DvbService objects in ascending order by original_network_id, and then sort them in descending order by transport_stream_id,
```

```
    //Sort them in ascending order by service_id
getAllServices([DVB_SORT_TYPE_ONET_ID, DVB_SORT_TYPE_TS_ID,
    DVB_SORT_TYPE_SERVICE_ID],
[DVB_SORT_ORDER_ASC,DVB_SORT_ORDER_DESC,DVB_SORT_ORDER_ASC]);
```

M.2.4.2.8 getService

Prototype: DvbService getService(original_network_id, transport_stream_id, service_id)

Description: Getting the one-way broadcast service object specified in the current network object.

Parameter: original_network_id – number type, indicating the original network ID;

transport_stream_id – number type, indicating the delivery stream ID;

service_id – number type, indicating the service ID.

Return: DvbService object, if the specified object does not exist, null is returned.

M.2.5 DvbBouquet object

DvbBouquet object is a local object, used to describe one-way broadcast service group information, and is uniquely identified by network_id and bouquet_id.

Example 1:

```
//Get an array of DvbBouquet objects through the method of the DvbBroadcast object, and then get
    a single DvbBouquet object
var dvbBouquetArray[] = DvbBroadcast.getAllBouquets(sortTypeArray[], sortOrderArray[]);
var dvbBouquet = dvbBouquetArray[i];
```

Example 2:

```
//Getting the specified DvbBouquet object through the method of the Broadcast object
var dvbBouquet = DvbBroadcast.getBouquet(network_id, bouquet_id);
```

Example 3:

```
//Getting the current array of DvbBouquet objects through the properties of the Broadcast object
var dvbBouquets[] = DvbBroadcast.currentDvbBouquets;
```

M.2.5.1 Property

The definition of the DvbBouquet object property is shown in Table M.5.

Table M.5 – DvbBouquet property

Property name	Type	Read/write property	Description
network_id	number	Read only	It indicates the one-way broadcast network identifier.
bouquet_id	number	Read only	It indicates the one-way broadcast service group identifier.
bouquet_name	string	Read only	It indicates the name of the one-way broadcast service group.

M.2.5.2 Method

M.2.5.2.1 getBouquetName

Prototype: string getBouquetName()

Description: Getting the full name of the service group. The full name of the service group is obtained from descriptors of the bouquet_name_descriptor descriptor or from the multilingual_bouquet_name_descriptor. This interface should return the full name of the service group that matches the preferred language set by the user. If the full name of the service group encoded in the language cannot be obtained from descriptors of the bouquet_name_descriptor or multilingual_bouquet_name_descriptor, it will return the full name of the service group carried in the bouquet_name_descriptor by default.

Parameter: None.

Return: string type, indicating the full name of the service group.

M.2.5.2.2 getShortBouquetName

Prototype: string getShortBouquetName()

Description: Getting the abbreviation of the service group name.

Parameter: None.

Return: string type, indicating the abbreviation of the service group name. If the abbreviation of the service group name does not exist, undefined is returned.

M.2.5.2.3 getAllTSs

Prototype: DvbTS[] getAllTSs()

Description: Getting all DvbTS objects in the current service group objects without sorting.

Parameter: None.

Return: An array of DvbTS objects.

M.2.5.2.4 getAllTSs

Prototype: DvbTS[] getAllTSs(sortTypeArray[], sortOrderArray[])

Description: Getting all DvbTS objects in the current service group objects, and sort them in a specified way.

Parameter: sortTypeArray – An array of number type, indicating the sorting basis. There can be one or more sorting basises, but they cannot be repeated. The priority of sorting is related to the order of the array members. The smaller the subscript of the array member, the higher the priority. For the value of each member of the array, see the constant definition of "Sorting basis" in Table M.2.

sortOrderArray – An array of number type, indicating the sorting method. The length of the array is the same as the sortTypeArray parameter. For the value of each member of the array, see the constant definition of "Sorting method" in Table M.2. The sortOrderArray and sortTypeArray must correspond one-to-one, that is, sortOrderArray[i] is only applicable to sortTypeArray[i].

Return: An array of DvbTS objects.

M.2.5.2.5 getTS

Prototype: DvbTS getTS(original_network_id, transport_stream_id)

Description: Getting the specified DvbTS object in the current service group objects.

Parameter: original_network_id – number type, indicating the original network ID;

transport_stream_id – number type, indicating the delivery stream ID.

Return: DvbTS object. If the specified object does not exist, null is returned.

M.2.5.2.6 getAllServices

Prototype: DvbService[] getAllServices()

Description: Getting all DvbService objects in the current service group without sorting.

Parameter: None.

Return: An array of DvbService objects.

M.2.5.2.7 getAllServices

Prototype: DvbService[] getAllServices(sortTypeArray[], sortOrderArray[])

Description: Getting all DvbService objects in the current service group, and sort them in a specified way.

Parameter: sortTypeArray[] – An array of number type, indicating the sorting basis of DvbService objects. There can be one or more sorting basises, but they cannot be repeated. The priority of sorting is related to the order of the array members. The smaller the subscript of the array member, the higher the priority. For the value of each member of the array, see the constant definition of "Sorting basis" in Table M.2.

sortOrderArray[] – An array of number type, indicating the sorting method of DvbService objects. The length of the array is the same as the sortTypeArray parameter. For the value of each member of the array, see the constant definition of "Sorting method" in Table M.2. The sortOrderArray and sortTypeArray must correspond one-to-one, that is, sortOrderArray[i] is only applicable to sortTypeArray[i].

Return: An array of DvbService objects.

M.2.5.2.8 getService

Prototype: DvbService getService(original_network_id, transport_stream_id, service_id)

Description: Getting the specified DvbService object in the current service group.

Parameter: original_network_id – number type, indicating the original network ID;

transport_stream_id – number type, indicating the delivery stream ID;

service_id – number type, indicating the service ID.

Return: DvbService object. If the specified object does not exist, null is returned.

M.2.6 DvbTS object

The DvbTS object is a local object, used to describe the information related to the unidirectional broadcast delivery stream, and is uniquely identified by network_id, original_network_id, and transport_stream_id.

Example 1:

```
//Getting the current DvbTS object through the properties of DvbBroadcast
    var dvbTS = DvbBroadcast.currentTS;
```

Example 2:

```
    //Getting an array of DvbTS objects through the method of the DvbBroadcast object, and then
        getting a single DvbTS object
    var dvbTSArray[] = DvbBroadcast.getAllTSs(sortTypeArray[], sortOrderArray[]);
```

```
var dvbTS = dvbTSArray[i];
```

Example 3:

```
//Getting the specified DvbTS object through the method of the DvbBroadcast object
var dvbTS = DvbBroadcast.getTS(network_id, original_network_id, transport_stream_id);
```

Example 4:

```
//Getting an array of DvbTS objects through the method of the DvbNetwork object, and then get a single DvbTS object
var dvbTSArray[] = dvbNetwork.getAllTSs(sortTypeArray[], sortOrderArray[]);
var dvbTS = dvbTSArray[i];
```

Example 5:

```
//Getting the specified DvbTS object through the method of the DvbNetwork object
var dvbTS = dvbNetwork.getTS(original_network_id, transport_stream_id);
```

Example 6:

```
//Getting an array of DvbTS objects through the method of the DvbBouquet object, and then get a single DvbTS object
var dvbTSArray[] = dvbBouquet.getAllTSs(sortTypeArray[], sortOrderArray[]);
var dvbTS = dvbTSArray[i];
```

Example 7:

```
//Getting the specified DvbTS object through the method of the DvbBouquet object
var dvbTS = dvbBouquet.getTS(original_network_id, transport_stream_id);
```

M.2.6.1 Property

The definition of the DvbTS object property is shown in Table M.6.

Table M.6 – DvbTS object property

Property name	Type	Read/write property	Description
network_id	number	Read only	It indicates the network ID to which the transport stream belongs.
original_network_id	number	Read only	It indicates the original network ID of the transport stream.
transport_stream_id	number	Read only	It indicates the transport stream ID.
deliveryType	number	Read only	It indicates the type of transport system, see the constant definition of "DVB transport system type" for the value.
dvbsTuningParams	DvbsTuning Parameters	Read only	It indicates DVB-S tuning and demodulation parameters. When deliveryType=DVB_DELIVERY_TYPE_DVB_S, this property is valid.
dvbcTuningParams	DvbcTuning Parameters	Read only	It indicates DVB-C tuning and demodulation parameters. When deliveryType=DVB_DELIVERY_TYPE_DVB_C, this property is valid.

Table M.6 – DvbTS object property

Property name	Type	Read/write property	Description
dvbtTuningParams	DvbtTuningParameters	Read only	It indicates DVB-T tuning and demodulation parameters. When deliveryType=DVB_DELIVERY_TYPE_DVB_T, this property is valid.
abssTuningParams	AbsssTuningParameters	Read only	It indicates ABS-SS tuning and demodulation parameters. When deliveryType=DVB_DELIVERY_TYPE_ABS_SS, this property is valid.
dtmbTuningParams	DtmbTuningParameters	Read only	It indicates DTMB tuning and demodulation parameters. When deliveryType=DVB_DELIVERY_TYPE_DTMB, this property is valid.
signalQuality	number	Read only	It indicates the signal quality of the frequency point where the transport stream is located, the value range being 0-100, 0 indicating the worst signal, and 100 indicating the best signal.
signalStrength	number	Read only	It indicates the signal strength of the frequency point where the transport stream is located, with a value range of 0-100, 0 indicating the weakest signal, and 100 indicating the strongest signal.
errorRate	string	Read only	It indicates the bit error rate of the frequency point where the transport stream is located.
signalLevel	string	Read only	It indicates the signal level of the frequency point where the transport stream is located.
signalNoiseRatio	string	Read only	It indicates the signal-to-noise ratio of the frequency point where the transport stream is located.

M.2.6.2 Method

M.2.6.2.1 getAllServices

Prototype: DvbService[] getAllServices()

Description: Getting all DvbService objects in the current delivery stream, without sorting.

Parameter: None.

Return: An array of DvbService objects.

M.2.6.2.2 getAllServices

Prototype: DvbService[] getAllServices(sortTypeArray[], sortOrderArray[])

Description: Getting all DvbService objects in the current delivery stream, and sorting them in a specified way.

Parameter: sortTypeArray[] – An array of number type, indicating the sorting basis of DvbService objects. There can be one or more sorting basises, but they cannot be repeated. The priority of sorting is related to the order of the array members. The smaller the subscript of the array member, the higher

the priority. For the value of each member of the array, see the constant definition of "Sorting basis" in Table M.2.

sortOrderArray[] – An array of number type, indicating the sorting method of DvbService objects. The length of the array is the same as the sortTypeArray parameter. For the value of each member of the array, see the constant definition of "Sorting method" in Table M.2. The sortOrderArray and sortTypeArray must correspond one-to-one, that is, sortOrderArray[i] is only applicable to sortTypeArray[i].

Return: An array of DvbService objects.

M.2.6.2.3 getServiceByID

Prototype: DvbService getServiceByID(service_id)

Description: Getting the specified DvbService object in the current delivery stream.

Parameter: service_id – number type, indicating the service ID.

Return: DvbService object. If the specified object does not exist, null is returned.

M.2.6.2.4 getServicesByType

Prototype: DvbService[] getServicesByType(service_type)

Description: Getting all DvbService objects of the specified service type in the current delivery stream.

Parameter: service_type – number type, indicating the service type, for the value, please see the constant definition of "Service type" in Table M.2.

Return: An array of DvbService objects. If the specified service object does not exist, the length of the returned array is 0.

M.2.7 DvbService object

The DvbService object is a local object, used to describe information related to one-way broadcast services, and is uniquely identified by network_id, original_network_id, transport_stream_id, and service_id.

Example 1:

```
//Getting the current DvbService object through the properties of the DvbBroadcast object.
var dvbService = DvbBroadcast.currentDvbService;
```

Example 2:

```
//Through the method of the DvbBroadcast object, get an array of DvbService objects, and then get a single DvbService object.
    var dvbServiceArray = DvbBroadcast.getAllServices(sortTypeArray[], sortOrderArray[]);
    var dvbService = dvbServiceArray[i];
```

Example 3:

```
    //Through the method of the DvbNetwork object, get an array of DvbService objects, and then get a single DvbService object.
    var dvbServiceArray = dvbNetwork.getAllServices(sortTypeArray[], sortOrderArray[]);
    var dvbService = dvbServiceArray[i];
```

Example 4:

```
    //Getting the specified DvbService object through the method of the DvbNetwork object.
```

var dvbService = dvbNetwork.getService(original_network_id, transport_stream_id, service_id);

Example 5:

//Through the method of the DvbBouquet object, get an array of DvbService objects, and then get a single DvbService object.

var dvbServiceArray = dvbBouquet.getAllServices(sortTypeArray[], sortOrderArray[]);

var dvbService = dvbServiceArray[i];

Example 6:

//Getting the specified array of the DvbService object through the method of the DvbBouquet object.

var dvbService = dvbBouquet.getService(original_network_id, transport_stream_id, service_id);

Example 7:

//Through the method of the DvbTS object, get an array of DvbService objects, and then get a single DvbService object.

var dvbServiceArray = dvbTS.getAllServices(sortTypeArray[], sortOrderArray[]);

var dvbService = dvbServiceArray[i];

Example 8:

//Getting the specified DvbService object through the method of the DvbTS object.

var dvbService = dvbTS.getServiceByID(service_id);

Example 9:

//Getting the specified array of the DvbService object through the method of the DvbTS object, and then get a single DvbService object.

var dvbServiceArray = dvbTS.getServicesByType(service_type);

var dvbService = dvbServiceArray[i];

M.2.7.1 Property

The definition of the DvbService object property is shown in Table M.7.

Table M.7 – DvbService object property

Property name	Type	Read/write property	Description
network_id	number	Read only	It indicates the network ID where the service is located.
original_network_id	number	Read only	It indicates the original network ID where the service is located.
transport_stream_id	number	Read only	It indicates the transport stream ID where the service is located.
service_id	number	Read only	It indicates the service ID.
service_name	string	Read only	It indicates the service name.
service_type	number	Read only	It indicates the service type, see the constant definition of "Service type" for the value.

Table M.7 – DvbService object property

Property name	Type	Read/write property	Description
service_provider_name	string	Read only	It indicates the name of the service provider.
running_status	number	Read only	It indicates the operating status of the service, and the value is: – 0-undefined; – 1-not running; – 2-Start after a few seconds (for example, video recording); – 3-Pause; – 4-Running; – 5-7-Reserved for use. NOTE – For NVOD service, running_status=0.
EIT_present_following_flag	boolean	Read only	It describes whether the current/follow-up information of the service exists in the current transport stream, the value is: – true-indicating that the current/follow-up information of the EIT of the service exists in the current transport stream; – false-indicating that the current/follow-up information of the EIT of the service is not in the current transport stream.
EIT_schedule_flag	boolean	Read only	It describes whether the EIT schedule information of the service exists in the current transport stream, the value is: – true-indicating that the EIT schedule information of the service exists in the current transport stream. – false-indicating that the EIT schedule information of the service is not in the current transport stream.
free_CA_mode	boolean	Read only	It describes whether the service is scrambled, the value is: – true-the receiving of one or more code streams is controlled by the CA system; – false-all components of the service are not scrambled.
bouquetIDs	number Array	Read only	Get the ID array of all the service groups to which the service belongs, and arrange them in ascending order by bouquet_id.
referServiceID	number	Read only	If service_type=SERVICE_TYPE_NVOD_SHIFT, the service_id of the corresponding reference service can be obtained through this property, otherwise the property returns undefined.
timeShiftServiceIDs	number Array	Read only	If service_type = SERVICE_TYPE_NVOD_REF, the service_id of all corresponding time-shift services can be obtained through this property, otherwise the property returns undefined.

Table M.7 – DvbService object property

Property name	Type	Read/write property	Description
curVideoObj	DvbVideoES	Read only	It indicates the video ES currently being played by the service.
curAudioObj	DvbAudioES	Read only	It indicates the audio ES currently being played by the service.
PCRPID	number	Read only	It indicates the PCR PID referenced by the service.

M.2.7.2 Method

M.2.7.2.1 getServiceName

Prototype: string getServiceName()

Description: Getting the full name of the service. The full name of the service is obtained from descriptors of the service_descriptor or the multilingual_service_name_descriptor. This interface should return the full name of the service that matches the preferred language set by the user. If the full name of the service encoded in the language cannot be obtained from descriptors of the service_descriptor or multilingual_service_name_descriptor, it will return the full name of the service carried in the service_descriptor by default.

Parameter: None.

Return: string type, indicating the full name of the service.

M.2.7.2.2 getShortServiceName

Prototype: string getShortServiceName()

Description: Getting the abbreviation of the service name.

Parameter: None.

Return: string type, indicating the abbreviation of the service name. If the abbreviation of the service name does not exist, undefined is returned.

Example: The [0x86]P[0x87]ay [0x86]M[0x87]ovie [0x86]C[0x87]hannel

Full name of the service – "The Pay Movie Channel".

Abbreviation of the service – "PMC".

M.2.7.2.3 getLocation

Prototype: string getLocation()

Description: Getting the service path described by the three elements of original_network_id, transport_stream_id, and service_id.

Parameter: None.

Return: string type.

M.2.7.2.4 getEvents

Prototype: DvbEvent[] getEvents(start, end)

Description: Getting the DvbEvent object within the specified time range of the service.

Parameter: start – string type, indicating the starting time, the format is "YYYY-MM-DD hh:mm:ss";

end – string type, indicating the ending time, the format is "YYYY-MM-DD hh:mm:ss".

Return: An array of DvbEvent objects. If the specified object does not exist, the length of the returned array is 0.

M.2.7.2.5 getVideoESs

Prototype: DvbVideoES[] getVideoESs()

Description: Getting all the video streams included in the service (including any video format type).

Parameter: None.

Return: An array of DvbVideoES objects. If there is no such object, the length of the returned array is 0.

getAudioESs

Prototype: DvbAudioES[] getAudioESs()

Description: Getting all audio streams included in the service (including any audio format types).

Parameter: None.

Return: An array of DvbAudioES objects. If there is no such object, the length of the returned array is 0.

M.2.7.2.6 getOtherESs

Prototype: DvbOtherES[] getOtherESs()

Description: Getting all non-video and audio streams included in the service.

Parameter: None.

Return: An array of DvbOtherES objects. If there is no such object, the length of the returned array is 0.

DvbVideoES object

The DvbVideoES object is a local object, used to describe the video elementary stream information in a certain service.

M.2.7.3 Property

The definition of the DvbVideoES object property is shown in Table M.8.

Table M.8 – Table of DvbVideoES property

Property name	Type	Read/write property	Description
stream_type	number	Read only	It indicates the type of video elementary stream, the value is: – 0x01-GB/T 17191.2 video; – 0x02-ISO/IEC 13818-2 video or GB/T 17191.2 restricted parameter video stream.
elementary_PID	number	Read only	Get the PID of the transport stream carrying the video elementary stream.
component_tag	number	Read only	It indicates the component ID. The system should obtain information from the stream_identifier_descriptor related to the elementary stream. If there is no information available, it is set to -1 by default.

M.2.8 DvbAudioES object

The DvbAudioES object is a local object, used to describe the audio elementary stream information in a certain service.

M.2.8.1 Property

The definition of the DvbAudioES object property is shown in Table M.9.

Table M.9 – Table of DvbAudioES property

Property name	Type	Read/write property	Description
stream_type	number	Read only	It indicates the type of audio elementary stream, the value is: – 0x03-GB/T 17191.3 audio; – 0x04-ISO/IEC 13818-3 audio.
elementary_PID	number	Read only	It indicates the PID of the transport stream that carries the audio elementary stream.
component_tag	number	Read only	It indicates the component ID. The system should obtain information from the stream_identifier_descriptor related to the elementary stream. If there is no information available, it is set to -1 by default.
Lingual	string	Read only	It indicates the audio language, and the three-letter code of the language follows the GB/T 4880.2-2000 standard.

M.2.9 DvbOtherES object

The DvbOtherES object is a local object, which is used to store the elementary stream information in a certain service except for audio and video.

M.2.9.1 Property

The definition of the DvbOtherES object property is shown in Table M.10.

Table M.10 – Table of DvbOtherES property

Property name	Type	Read/write property	Description
stream_type	number	Read only	It indicates the type of elementary stream, and the value is other allowed values except 0x01, 0x02, 0x03 and 0x04.
elementary_PID	number	Read only	It indicates the PID of the transport stream that carries the elementary stream.
component_tag	number	Read only	It indicates the component ID. The system should obtain information from the stream_identifier_descriptor related to the elementary stream. If there is no information available, it is set to -1 by default.

M.2.10 DvbEvent object

The DvbEvent object is a local object, used to save program event information in digital TV broadcasting and sound broadcasting channels, and is uniquely identified by network_id, original_network_id, transport_stream_id, service, and event_id.

Example:

//The DvbEvent object can be obtained in the following ways.

var dvbEventArray[] = dvbService.getEvents(startDate, endDate).

M.2.10.1 Property

The definition of the DvbEvent object property is shown in Table M.11.

Table M.11 – Table of DvbEvent property

Property name	Type	Read/write property	Description
network_id	number	Read only	It indicates the network ID to which the program event belongs.
original_network_id	number	Read only	It indicates the original network ID to which the program event belongs.
transport_stream_id	number	Read only	It indicates the transport stream ID to which the program event belongs.
service_id	number	Read only	It indicates the service ID to which the program event belongs.
event_id	number	Read only	It indicates the program event ID.
event_name	string	Read only	It indicates the name of the program event.
event_description	string	Read only	It indicates the description of the program event.
running_status	number	Read only	It indicates the running status of the program event, the value is: – 0-undefined; – 1-not running; – 2-Start after a few seconds (for example, video recording); – 3-Pause; – 4-Running; – 5-7-Reserved for use. NOTE – For NVOD reference program events, running_status=0.
startDate	string	Read only	It indicates the starting date of the event playing, the format is "YYYY-MM-DD".
startTime	string	Read only	It indicates the starting time of the event playing, the format is "hh:mm:ss".
Duration	number	Read only	It indicates the duration of the event playback, in seconds.
endDate	string	Read only	It indicates the ending date of the event playing, the format is "YYYY-MM-DD".
endTime	string	Read only	It indicates the ending time of the event playback, the format is "hh:mm:ss".
content_nibble	number	Read only	It indicates the event classification value (8 bits), where the upper 4 bits are the first-level program content classification value (content_nibble_level_1), and the lower 4 bits are the second-level program content classification value (content_nibble_level_2).

Table M.11 – Table of DvbEvent property

Property name	Type	Read/write property	Description
user_nibble	number	Read only	It indicates the classification value (8 bits) defined by the event operator, where the upper 4 bits are the first-level program content classification value (content_nibble_level_1), and the lower 4 bits are the second-level program content classification value (content_nibble_level_2).
minAge	number	Read only	It indicates the minimum age at which the event can be watched.
free_CA_mode	boolean	Read only	It indicates whether the event is scrambled, the value is: –true-scrambled; –false-No scrambling.

M.2.10.2 Method

M.2.10.2.1 getEventName

Prototype: string getEventName()

Description: Getting the full name of the program event.

Parameter: None.

Return: string type, indicating the full name of the program event.

M.2.10.2.2 getShortEventName

Prototype: string getShortEventName()

Description: Getting the abbreviation of the program event.

Parameter: None.

Return: string type, indicating the abbreviation of the program event. If the abbreviation of the event does not exist, undefined is returned.

M.2.10.2.3 getEventDescription

Prototype: string getEventDescription()

Description: Getting the description information of the program event. The program event description information is obtained from the description of short_event_descriptor. This method should return the program event description information that matches the preferred language set by the user.

Parameter: None.

Return: string type, indicating the description of the program event.

M.2.10.2.4 getLocation

Prototype: string getLocation()

Description: Getting the event locator described by the four elements of original_network_id, transport_stream_id, service_id and event_id.

Parameter: None.

Return: string type, indicating the program event locator.

Annex N

JavaScript-Two-way broadband network access unit

(This annex forms an integral part of this Recommendation.)

N.1 Overview

This annex defines the functional modules related to two-way broadband network access control: broadband network settings.

N.2 Broadband network setting module

This module defines JS objects related to broadband network settings: Broadband, Ethernet, IP, Proxy, and the relationship is shown in Figure N.1.

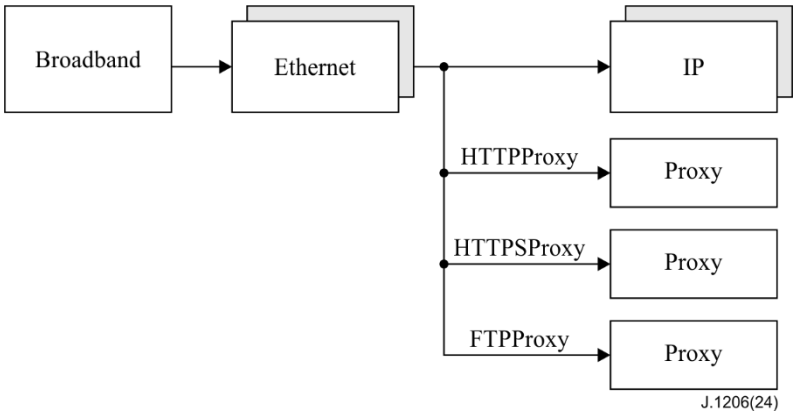


Figure N.1 – Schematic diagram of object relationship of broadband network setting module

N.2.1 Message

The definition of messages that the broadband network setting module may send to the application layer is shown in Table N.1.

Table N.1 – Messages of broadband network setting module

Message name	event.which	event.modifiers	Message description
MSG_BROADBAND_SAVE_CFG_SUCCESS	12001	–	The network configuration information was successfully written to the NVM.
MSG_BROADBAND_SAVE_CFG_FAILED	12002	–	Failed to write network configuration information to NVM.
MSG_BROADBAND_SUBMIT_SUCCESS	12003	–	The network configuration parameters were submitted successfully.
MSG_BROADBAND_SUBMIT_FAILED	12004	–	Failed to submit network configuration parameters.
MSG_BROADBAND_NTP_READY_SUCCESS	12005	–	The synchronization of the network NTP time is successful.
MSG_BROADBAND_NTP_SYNC_TIMEOUT	12006	–	The synchronization of the network NTP time timed out.

Table N.1 – Messages of broadband network setting module

Message name	event.which	event.modifiers	Message description
Reserved	12007～12014		
MSG_BROADBAND_DHCP_READY_SUCCESS	12015	–	The network DHCP function is successfully enabled.
MSG_BROADBAND_DHCP_TIMEOUT	12016	–	The network DHCP function has timed out.
Reserved	12017～12027		
MSG_BROADBAND_PING_RESPONSE	12028	number	PING command response.

MSG_BROADBAND_GET_NETWORK_STATE	12029	number	Network status, the message description string JSON format is:
			{
			"targetAddress":param1[Note 1],
			"lostPacketRate":param2[Note 2],
			"bandwidth":param3[Note 3],
			"delay":param4[Note 4]
			}

MSG_BROADBAND_LINK_AP_SUCCESS	12030	–	The wireless AP is successfully connected.
MSG_BROADBAND_LINK_AP_FAILED	12031	–	Failed to connect the wireless AP.
MSG_BROADBAND_SCAN_AP_FIND	12032	number	Each time a wireless hotspot is successfully found, the message is sent.
MSG_BROADBAND_SCAN_AP_SUCCESS	12033	number	Send the message when the designated number of wireless hotspots are searched.
MSG_BROADBAND_SCAN_AP_FAILED	12034	–	If no wireless hotspot is found within the specified timeout period, the message will be sent.
Reserved	12035～12055		
MSG_BROADBAND_DISCONNECTED	12056	–	The network cable is disconnected.
MSG_BROADBAND_CONNECTED	12057	–	The network cable is connected.
MSG_BROADBAND_LAN_DISCONNECTED	12058	–	This message is sent when the network is disconnected.
MSG_BROADBAND_LAN_CONNECTED	12059	–	This message is sent when the network is connected.
MSG_BROADBAND_IP_RENEW	12060	number	In DHCP mode, the IP address is automatically updated.
Reserved	12061～12500		

Table N.1 – Messages of broadband network setting module

Message name	event.which	event.modifiers	Message description
The value of event.modifiers is automatically given by the system, and its data type is: – "Number", indicating that the value is the ID of the message description string, which can be obtained through the Utility.getEventInfo() method. If the "message description" defines the message string JSON format, the message content will be retrieved according to the format. – "–", indicating that event.modifiers is undefined. NOTE 1 – param1: string type, indicating the target IP address. NOTE 2 – param2: number type, indicating packet loss rate. NOTE 3 – param3: number type, indicating bandwidth, in Mbps. NOTE 4 – param2: number type, indicating delay, in ms.			

N.2.2 Broadband object

The Broadband object is a built-in object that describes the properties of two-way broadband and network configuration information.

N.2.2.1 Property

The definition of Broadband object property is shown in Table N.2.

Table N.2 – Property of Broadband object

Property name	Type	Property	Description
portalIP	string	Read only	It indicates the IP address of the portal server, the format is: – IPv4-"xxx.xxx.xxx.xxx", see IETF RFC 791 for details; – IPv6-"x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.
portalPort	number	Read only	It indicates the port of the portal server.
appServerIP	string	Read only	It indicates the IP address of the application download server, format: – IPv4-"xxx.xxx.xxx.xxx", see IETF RFC 791 for details; – IPv6-"x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.
appServerPort	number	Read only	It indicates the port of the application download server.
Host	string	Read only	It indicates a character string with which the network recognizes the terminal device, that is, the host name, which does not exceed 255 bytes. The terminal device can also be identified by this name in the LAN.
workGroup	string	Read only	It indicates the workgroup to which the network belongs.

N.2.2.2 Method

N.2.2.2.1 getAllEthernets

Prototype: Ethernet[] getAllEthernets()

Description: Getting all broadband network equipment objects of the receiving terminal.

Parameter: empty.

Return: Array of Ethernet objects. If there is no such object, the length of the returned array is 0.

N.2.2.2.2 ping

Prototype: ping(address, parameter)

Description: An Asynchronous method, executing the ping command of the operating system. Each time the system receives a ping packet response, it sends a message MSG_BROADBAND_PING_RESPONSE to the page, the event.modifiers property returns the ID of the message description string, and the message description string is obtained through the Utility.getEventInfo() method.

Parameter: address – string type, indicating the target address of the ping command, which can be IP or domain name.

parameter – string type, indicating the parameter of the ping command, the value is:

– "-t"–indicating continuous ping the target address until manually stopped.

Return: None.

N.2.2.2.3 cancelPing

Prototype: number cancelPing()

Description: Cancel the ongoing ping operation.

Parameter: None.

Return: number type, the value being:

– 0-indicating that the current ping operation has ended or there is currently no ping operation;

– 1-indicating that the ping operation being executed has been successfully stopped.

N.2.2.2.4 queryNetworkState

Prototype: queryNetworkState(targetAddress)

Description: An Asynchronous method, getting the network status from the current network to the specified target address. When the operation is completed, the system should send the message MSG_BROADBAND_GET_NETWORK_STATE to the page, the property of event.modifiers returns the ID of the message description string, and the message description string is obtained through the Utility.getEventInfo(ID) method. The JSON format is:

```
{
"targetAddress":"192.168.1.12",
"lostPacketRate":0,
"bandwidth":8,
"delay":100
}
```

where:

– targetAddress – string type, indicating the target address, the format is:

- IPv4 – "xxx.xxx.xxx.xxx", see IETF RFC 791 for details;
- IPv6 – "x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.

– lostPacketRate – number type, indicating the packet loss rate, percentage, and the value range being 0～100;

– bandwidth – number type, indicating the bandwidth, in Mbps;

– delay – number type, indicating the delay, in milliseconds.

Parameter: targetAddress–string type, indicating the target IP address, the format is:

- IPv4 – "xxx.xxx.xxx.xxx", see IETF RFC 791 for details;
- IPv6 – "x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.

Return: None.

N.2.2.2.5 getDeviceState

Prototype: number getDeviceState(device)

Description: Getting device status.

Parameter: device – String type, indicating the device name, the device type can be network card or cable modem.

– If it is a network card, the value is "eth0", "eth1", etc.;

– If it is a cable modem, the value is "cm".

Return: number type, the device status value is shown in Table N.3.

Table N.3 – Table of device status

Device	Return value	Description
All devices are applicable	0	The device does not exist and cannot be accessed.
	1	The device exists, but the status cannot be obtained.
eth0, eth1 and other network card devices	101	The network cable is connected.
	102	The network cable is off.
Cm	201	The network connection is normal.
	202	The network connection is interrupted.
	203	The network status does not change, but it is bad.
	204	The communication between the Cable Modem and the receiving terminal is interrupted.
	205	The communication between the Cable Modem and the receiving terminal is interrupted, the reconnection is performed.

N.2.2.2.6 NTPUpdate

Prototype: boolean NTPUpdate()

Description: An asynchronous method, synchronizing NTP time.

– If the page gets the message MSG_BROADBAND_NTP_READY_SUCCESS, it indicates that the synchronization of the network NTP time is successful;

– If the page gets the message MSG_BROADBAND_NTP_SYNC_TIMEOUT, it indicates that the synchronization of the network NTP time has timed out.

Parameter: None.

Return: boolean type, true indicating that the execution of the synchronization of NTP time starts, false indicating that the execution is not performed.

NOTE – The return value only indicates whether the program starts to perform the synchronization of NTP time, and does not indicate the actual synchronization result. The actual synchronization result is notified by a message.

If it returns false, it may be caused by reasons such as not setting the NTP server.

N.2.2.7 save

Prototype: save()

Description: An Asynchronous method, writing network information into NVM.

– If the save is successful, send the message MSG_BROADBAND_SAVE_CFG_SUCCESS;

– If the save fails, send the message MSG_BROADBAND_SAVE_CFG_FAILED.

Parameter: None.

Return: None.

N.2.3 Ethernet object

The Ethernet object is a local object, used to describe the detailed information of the local network card, and corresponds to the network card one-to-one, that is, the number of the network cards corresponds the number of the Ethernet objects.

Example:

```
var ethernetArray = Broadband.getAllEthernets();//Getting an array of all network card objects of the receiving terminal
var ethernet = ethernetArray[i];
```

N.2.3.1 Property

The definition of the Ethernet object property is shown in Table N.4.

Table N.4 – Ethernet object property

Property name	Type	Read/write property	Description
enableFlag	number	Read/write	It indicates whether the network card is available, which is equivalent to the enable or disable function on the PC, the value is: – 1-indicating that the network card is available; – 0-indicating that the network card is not available. NOTE – The submitParameters() method takes effect only upon being called.
DHCPflag	boolean	Read/write	It indicates whether to enable DHCP, the value is: – true-indicating DHCP is enabled; – false-indicating DHCP is disabled. NOTE – The submitParameters() method take effect only upon being called.
DHCPAutoGetDNS	number	Read/write	It indicates whether this Ethernet object uses automatically obtained DNS, the value is: – 0-indicating do not use dynamically obtained DNS, but use manually set DNS; – 1-indicating use dynamically obtained DNS instead of manually set DNS. If Ethernet.DHCPflag=false; Setting this property is invalid. NOTE – The submitParameters() method takes effect only upon being called.
description	string	Read only	It indicates the description information of the network card.

Table N.4 – Ethernet object property

Property name	Type	Read/write property	Description
MACAddress	string	Read only	It indicates the 6-byte physical address of the current network card, and every two hexadecimal numbers are separated by "-", such as "00-15-F2-63-7A-83".
Ips	IP Array	Read only	It indicates an array of IP objects.
DNSs	string Array	Read only	It indicates an array of DNS server IP addresses, the format is: –IPv4-"xxx.xxx.xxx.xxx", see IETF RFC 791 for details; –IPv6-"x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.
DHCPLeaseObtained	string	Read only	It indicates the start time of IP address lease in DHCP mode, the format is: yyyy-mm-dd hh:mm:ss.
DHCPLeaseExpires	string	Read only	It indicates the expiration time of the IP address in DHCP mode, the format is: yyyy-mm-dd hh:mm:ss.
DHCPIP	IP	Read only	It indicates an IP object dynamically assigned by DHCP.
DHCPServer	string	Read only	It indicates that the IP address of the DHCP server is provided on the current network, the format is: –IPv4-"xxx.xxx.xxx.xxx", see IETF RFC 791 for details; –IPv6-"x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.
DHCPPort	number	Read only	It indicates the port number of DHCP.
communicateWay	number	Read/write	It indicates network bandwidth and communication method, the value is: – 0-indicating adaptive; – 1-indicating 10M full duplex; – 2-indicating 10M half-duplex; – 3-indicating 100M full duplex; – 4-indicating 100M half-duplex.
LANStatus	number	Read only	It indicates the connection status of Ethernet mode, the value is: – 0-indicating no connection; – 1-indicating connected.
sentPackages	number	Read only	It indicates the number of data packets sent by the network card.
receivedPackages	number	Read only	It indicates the number of data packets received by the network card.
currentConnectionType	string	Read only	It indicates the connection type of the current network card, the value is "ethernet", "ppp" or "pppoe", etc.
HTTPProxy	Proxy	Read only	It indicates the hyper text transfer protocol (HTTP) proxy object.
HTTPSProxy	Proxy	Read only	It indicates the HTTPS proxy object.
FTPProxy	Proxy	Read only	It indicates the FTP proxy object.

N.2.3.2 Method

N.2.3.2.1 setDNS

Prototype: setDNS(index, dns)

Description: Modifying the DNS server.

Parameter: index − index of the DNS server to be modified;

dns – string type, DNS IP address, the format is:

– IPv4-"xxx.xxx.xxx.xxx", see IETF RFC 791 for details;

– IPv6-"x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.

Return: None.

N.2.3.2.2 addIP

Prototype: boolean addIP(ip)

Description: Add an IP object to the network card.

Parameter: ip – IP object.

Return: boolean type, true indicating success, false indicating failure.

N.2.3.2.3 deleteIPByIndex

Prototype: boolean deleteIPByIndex(index)

Description: Delete the IP object set in the network card through the IP array subscript.

Parameter: index – number type, indicating the array subscript.

Return: boolean type, true indicating success, false indicating failure.

N.2.3.2.4 deleteIPByAddress

Prototype: boolean deleteIPByAddress(address)

Description: Delete the IP object set in the network card by IP address.

Parameter: address – string type, indicating the IP address, the format is:

– IPv4-"xxx.xxx.xxx.xxx", see IETF RFC 791 for details;

– IPv6-"x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.

Return: boolean type, true indicating success; false indicating failure.

N.2.3.2.5 getAPs

Prototype: AP[] getAPs()

Description: Getting all wireless hotspot objects. Only used for wireless network card.

Parameter: None.

Return: An AP object array.

N.2.3.2.6 scanAP

Prototype: scanAP(maxCount, timeOut)

Description: An Asynchronous method, searching for wireless AP.

– If no wireless hotspot is found within the timeout period specified by the parameter timeOut, send the message MSG_BROADBAND_SCAN_AP_FAILED to the page;

– Each time a wireless hotspot is searched, send the message MSG_BROADBAND_SCAN_AP_FIND to the page. The event.modifiers property carries the JSON string ID of the AP object, and the JSON string is obtained through the Utility.getEventInfo(ID) method.

– If the search for maxCount wireless hotspots has been completed, send the message MSG_BROADBAND_SCAN_AP_SUCCESS to the page.

Parameter: maxCount – number type, indicating the maximum number of APs to be searched.

timeOut – number type, indicating the search timeout period, in seconds.

Return: None.

N.2.3.2.7 connectAP

Prototype: connectAP(essId, keyType, key)

Description: An Asynchronous method, connecting to the wireless network card access point.

– If the connection is successful, send the message MSG_BROADBAND_LINK_AP_SUCCESS;

– If the connection fails, send the message MSG_BROADBAND_LINK_AP_FAILED.

Parameter: essId – string type, indicating the wireless network card ID.

keyType – number type, indicating the key data type of the wireless network card.

key – string type, indicating an authentication key.

Return: None.

N.2.3.2.8 disconnectAP

Prototype: number disconnectAP()

Description: Disconnecting from the wireless hotspot.

Parameter: None.

Return: number type, returning 0 if it succeeds, and returning an error code if it fails.

N.2.3.2.9 getConnectedAP

Prototype: AP getConnectedAP()

Description: Getting the connected wireless hotspot object.

Parameter: None.

Return: AP object.

N.2.3.2.10 submitParameters

Prototype: submitParameters()

Description: An Asynchronous method, submitting all the properties of the Ethernet object of the network card to the bottom layer at one time.

– If the submission is successful, send the message MSG_BROADBAND_SUBMIT_SUCCESS to the page;

– If the submission fails, send the message MSG_BROADBAND_SUBMIT_FAILED to the page.

Parameter: None.

Return: None.

N.2.4 AP object

The AP object is a local object, which is used to describe wireless hotspot information.

N.2.4.1 Property

The definition of AP object property is shown in Table N.5.

Table N.5 – Table of AP object property

Property name	Type	Read/write property	Description
essId	string	Read only	It indicates the ID of the wireless hotspot.
signalStrength	string	Read only	It indicates the signal strength of the wireless hotspot.
linkQuality	string	Read only	It indicates the link quality of the wireless hotspot.
encType	number	Read only	It indicates the wireless hotspot authentication encryption type.

The definition of wireless network card authentication encryption type is shown in Table N.6.

Table N.6 – Authentication encryption type of wireless network card

Type value	Description
0	No encryption.
1	WEPOpen encryption.
2	Shared sharing mode to encrypt WEP information.
3	Wi-Fi Protected Access Protocol.
4	Wi-Fi Protected Access Protocol 2.

The error code comparison table of the wireless network card is shown in Table N.7.

Table N.7 – Error code comparison table of wireless network card

Error code	Meaning
0	Success.
1	Parameter error.
2	The wireless network card failed to start.
3	The wireless network card failed to be switched off.
4	Error in connecting.
5	Error in setting the key.
6	Error in clearing the authentication information.
7	The name of the wireless network card access point was not obtained.

The definition of the data type of wireless network card key is shown in Table N.8.

Table N.8 – Data type of wireless network card key

keyType value	Description
0	Encryption in hexadecimal mode.
1	Encryption in ansi coding mode.
2	Encryption in string method.

N.2.5 IP object

The IP object is a local object, used to describe IP address information.

Example:

```
//Create and set IP objects through the network card
var ethernetArray = Broadband.getAllEthernets();//Get an array of all network card objects of the receiving terminal
var ethernet = ethernetArray[0];
var ip = ethernet.IPs[0];//Getting the 0th IP object in the IP object array under the 0th network card object
//Through the construction method, create an IP object
ip = new IP(address, mask, gateway);
```

N.2.5.1 Property

The definition of the IP object property is shown in Table N.9.

Table N.9 – Table of IP object property

Property name	Type	Property	Description
Address	string	Read/write	It indicates the IP address, the format is: – IPv4-"xxx.xxx.xxx.xxx", see IETF RFC 791 for details; – IPv6-"x:x:x:x:x:x:x:x", see IETF RFC 2373 for details.
Mask	string	Read/write	It indicates the subnet mask of the IP address, and the format corresponds to the IP address.
Gateway	string	Read/write	It indicates the gateway of the IP address, the format is: – IPv4 – "Xxx.xxx.xxx.xxx", see IETF RFC 791 for details; – IPv6 – "X:x:x:x:x:x:x:x", see IETF RFC 2373 for details.

N.2.5.2 Method

N.2.5.2.1 IP

Prototype: IP(address, mask, gateway)

Description: IP object construction method.

Parameter: address – string type, indicating the IP address, the format is the same as the description of the "address" property in Table T.9.

mask – string type, indicating the subnet mask of the IP address, the format is the same as the description of the "mask" property in Table T.9.

gateway – string type, indicating the gateway of the IP address, the format is the same as the description of the "gateway" property in Table T.9.

N.2.6 Proxy object

The Proxy object is a local object, describing the proxy server information in the network.

N.2.6.1 Property

The definition of the Proxy object property is shown in Table N.10.

Table N.10 – Table of Proxy object property

Property name	Type	Property	Description
userName	string	Read/write	It indicates the username of the current proxy server access mode, the value is 0-30 characters, and cannot be a space. The proxy server requires a username and password to connect to some proxies.
password	string	Read/write	It indicates the password of the current proxy server access mode, the value is 0-30 characters, and it can be a space. The proxy server requires a username and password to connect to some proxies.
Enable	number	Read/write	It indicates whether to enable proxy, the value is: – 0-indicating that the current proxy is invalid; – 1-indicating that the current proxy takes effect immediately.
unusedProxyURLs	string Array	Read/write	It indicates the URL that does not use a proxy, and each character string cannot exceed 255 characters at most.
Server	string	Read/write	It indicates the IP address or domain name address of the proxy server, which cannot exceed 255 characters.
Port	number	Read/write	It indicates the port number of the server that provides the proxy service, and the value is 0 to 5 numeric characters.

Annex O

JavaScript-Human-computer interaction unit

(This annex forms an integral part of this Recommendation.)

O.1 Overview

This annex defines functional modules related to human-computer interaction: user input module, front panel output module.

O.2 User input module

O.2.1 Overview of the user input module

User input refers to the user sending user instructions to the receiving terminal through some input devices such as remote control, mouse, keyboard, front panel keys, etc., and these user instructions are uniformly encapsulated into key messages for processing.

Sources of messages captured by the application layer:

Remote control, keyboard, mouse, front panel and other keys to trigger messages;

The application layer captures messages in the following ways:

- Keyboard: The application captures keyboard messages through document.onkeydown, document.onkeyup and document.onkeypress, and the message code is consistent with the PC method;
- Mouse: The application captures mouse messages through document.onmousedown, document.onmouseup, document.onmousemove, etc., and the message code is consistent with the PC method;
- Remote control: The application captures remote control messages through document.onkeydown and document.onkeyup, and the message code is compatible with the PC mode;
- Front panel: The application captures front panel messages through document.onkeydown and document.onkeyup, and the message code is consistent with the remote control method.

O.2.2 event object

O.2.2.1 Property

The event object is a built-in object, and the properties of the event object are shown in Table O.1.

Table O.1 – Properties of event object

Property name	Type	Read/write property	Description
event.type	number	Read only	It indicates the type of event that occurred, that is, the name of the event represented by the current event object, which has the same name as the registered event handle, such as "onclick"; or the property of the event handle deletes the prefix "on", such as "click".
event.source	number	Read only	It indicates the source of the message.
event.which	number	Read only	It indicates the code value of the message.

Table O.1 – Properties of event object

Property name	Type	Read/write property	Description
event.modifiers	number	Read only	It indicates the extended property of the message. If the extended property of the message is empty, modifiers returns 0; if the extended property of the message is number type, then modifiers returns the value; if the extended property of the message is a character string, then modifiers returns an ID value, which is generated internally by the system, as a pointer to the specific character string content, the application can call the Utility.getEventInfo(ID) method to retrieve the character string content.

O.2.2.2 Message source

The definition of the message source (event.source) is shown in Table O.2.

Table O.2 – Definition of event.source

event.source	Description
1002	It indicates the remote control key message.
1003	It indicates the front panel key message.

O.2.2.3 Key message

The definition of the key message is shown in Table D.2.1.5.

O.3 Front panel output module

This module defines the JS object related to the front panel output: FrontPanel.

O.3.1 FrontPanel object

The FrontPanel object is a built-in object that provides front panel operation interfaces such as character string display, status indication, time and date display, and information clearing.

O.3.1.1 Constant

Constant definitions of the FrontPanel object are shown in Table O.3.

Table O.3 – Constant of FrontPanel object

Constants	Description
Indication type	
const TYPE_MAIL = 0;	Mail
const TYPE_SIGNAL = 1;	Signal
const TYPE_POWER = 2;	Power supply
const TYPE_RADIO = 3;	Broadcast
Indicating status	
const STATUS_OFF = 0;	Off
const STATUS_ON = 1;	On

Table O.3 – Constant of FrontPanel object

Constants	Description
const STATUS_UNKNOWN = 2;	Unknown
Character string display alignment	
const ALIGN_CENTER = 0;	horizontal center align
const ALIGN_LEFT = 1;	horizontal left align
const ALIGN_RIGHT = 2;	horizontal right align

O.3.1.2 Method

O.3.1.2.1 clear

Prototype: boolean clear()

Description: Clearing the information displayed on the front panel, including the character string information displayed on the front panel, time and date information, etc.

Parameter: empty.

Return: boolean type, true indicating the removal is successful, false indicating the removal failed.

O.3.1.2.2 displayDate

Prototype: boolean displayDate(date)

Description: Displaying the current time and date information.

Parameter: date – Date type, indicating time and date.

Return: boolean type, true indicating display success, false indicating display failure. If the terminal does not support it, it does not respond to the call of this method and returns false.

O.3.1.2.3 displayText

Prototype: boolean displayText(str)

Description: displaying a character string, which is displayed in horizontal center alignment by default.

Parameter: str – string type, indicating the character string to be displayed.

Return: boolean type, true value indicating that the display is successful, and false value indicating that the display fails.

O.3.1.2.4 displayText

Prototype: boolean displayText(str, align)

Description: This character string is displayed with specified alignment. If the terminal does not support the specified alignment, this parameter can be ignored.

Parameter: str – string type, indicating the character string to be displayed.

align – number type, indicating the horizontal alignment.

Return: boolean type, true value indicating that the display is successful, and false value indicating that the display fails.

O.3.1.2.5 getMaxChars

Prototype: number getMaxChars()

Description: Getting the number of display characters supported by the front panel.

Parameter: None.

Return: number type, indicating the number of display characters supported by the front panel.

O.3.1.2.6 getStatus

Prototype: number getStatus(type)

Description: Getting the front panel indication status according to the type.

Parameter: type – number type, specify the indication type.

Return: number type, indicating the indication status. If the type parameter specifies a valid indication type, its actual status (value STATUS_ON or STATUS_OFF) is returned; if the type parameter specifies an invalid indication type, it returns STATUS_UNKNOWN.

O.3.1.2.7 setStatus

Prototype: boolean setStatus(type, status)

Description: Setting the front panel indication status.

Parameter: type – number type, specify the indication type;

status – number type, indication status.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

Annex P

JavaScript-AV setting unit

(This annex forms an integral part of this Recommendation.)

P.1 Overview

This annex defines the function modules related to AV settings: audio parameter settings and video parameter settings. All setting methods not only write the setting parameters into NVM, but also take effect immediately.

P.2 Audio and video parameter setting module

This module defines JS objects related to audio and video parameter settings: AudioSetting and VideoSetting. The system should verify the permissions of the application, and only privileged applications can call the methods provided by this class.

P.2.1 AudioSetting object

AudioSetting is a built-in object that provides a method for setting audio parameters.

P.2.1.1 Constant

The constant definition of AudioSetting object is shown in Table P.1.

Table P.1 – Constants of AudioSetting object

Constants	Description
Channel type	
const CHANNEL_STEREO = 0;	Stereo
const CHANNEL_LEFT = 1;	Left channel
const CHANNEL_RIGHT = 2;	Right channel
const CHANNEL_MIXED_MONO = 3;	Mixed sound
Audio port type	
const PORT_HDMI = 0;	HDMI port
const PORT_SPDIF = 1;	SPDIF port

P.2.1.2 Method

P.2.1.2.1 getOutputInterfaceList

Prototype: string[] getOutputInterfaceList()

Description: Getting a list of all available audio output ports of the receiving terminal.

Parameter: None.

Return: A string array, indicating the names of all available audio output interfaces of the receiving terminal, such as "RCA", "S/PDIF", "HDMI", etc. If there is no audio output port available, the length of the returned array is 0.

P.2.1.2.2 getOutputInterfaceStatus

Prototype: boolean getOutputInterfaceStatus(port)

Description: Getting the enabling status of the audio output port.

Parameter: port – string type, indicating the name of the audio output interface, obtained by the getOutputInterfaceList() method.

Return: boolean type, true value indicating the audio output port is allowed to output, and false value indicating that the audio output port is forbidden to output.

P.2.1.2.3 disableOutputInterface

Prototype: boolean disableOutputInterface(port)

Description: Disable output of the audio output port.

Parameter: port – string type, indicating the name of the audio output interface, obtained by the getOutputInterfaceList() method.

Return: boolean type, true value indicating the prohibition is successful, and false value indicating the prohibition fails.

P.2.1.2.4 enableOutputInterface

Prototype: boolean enableOutputInterface(port)

Description: Allow audio port output.

Parameter: port – string type, indicating the name of the audio output interface, obtained by the getOutputInterfaceList() method.

Return: boolean type, true value indicating success, and false value indicating failure.

P.2.1.2.5 getOutputVolume

Prototype: number getOutputVolume()

Description: Getting the global output volume.

Parameter: None.

Return: number type, indicating the global output volume, the value range being 0-100, 0 indicating mute, 100 indicating maximum volume.

P.2.1.2.6 setOutputVolume

Prototype: boolean setOutputVolume(volume)

Description: Setting the global output volume.

NOTE – The actual output volume of a broadcast program = global output volume + the increase of the broadcast program relative to the global output volume.

Parameter: volume – number type, indicating the global output volume, the value range being 0-100, 0 indicating mute, 100 indicating maximum volume.

Return: boolean type, true indicating setting is successful, false indicating setting failed.

P.2.1.2.7 getOutputChannelMode

Prototype: number getOutputChannelMode()

Description: Getting the current output channel type.

Parameter: None.

Return: number type.

P.2.1.2.8 setOutputChannelMode

Prototype: boolean setOutputChannelMode(audioChannel)

Description: Setting the current output channel type.

Parameter: audioChannel – number type.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.1.2.9 getOutputSPDIFMode

Prototype: number getOutputSPDIFMode()

Description: Getting the data format of the S/PDIF output interface (compressed or PCM format).

Parameter: None.

Return: number type, indicating the data format of the S/PDIF output interface, with a value of 0 for PCM format, and 1 for compressed format.

P.2.1.2.10 setOutputSPDIFMode

Prototype: setOutputSPDIFMode(mode)

Description: Setting the data format of the S/PDIF output interface (compressed or PCM format).

Parameter: mode − number type, indicating the audio mode of the SPDIF output interface, the value is:

– 0-indicating the PCM format, that is, compressed audio is decoded by the receiving terminal;
– 1-indicating the compressed format, that is, compressed audio is decoded by an external decoding device.

Return: None.

P.2.1.2.11 isMute

Prototype: boolean isMute()

Description: Getting the mute status flag.

Parameter: None.

Return: boolean type, true value indicating mute, false value indicating sound.

P.2.1.2.12 mute

Prototype: boolean mute()

Description: Setting mute.

Parameter: None.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.1.2.13 unMute

Prototype: boolean unMute()

Description: Unmute.

Parameter: None.

Return: boolean type, true value indicating the cancellation is successful, and false value indicating that the cancellation failed.

P.2.1.2.14 getOutputHDMIFMode

Prototype: number getOutputHDMIFMode()

Description: Getting the data format of the HDMI output interface.

Parameter: None.

Return: number type, indicating the data format of the HDMI output interface.

– 0-indicating off;
– 1-Auto-negotiation;
– 2-LPCM, output decoded signal;
– 3-RAW, output the original signal.

P.2.1.2.15 setOutputHDMIFMode

Prototype: setOutputHDMIMode(mode)

Description: Setting the data format of the HDMI output interface (compressed or PCM format).

Parameter: mode−number type, indicating the audio mode of the HDMI output interface, the value is:

– 0-indicating off;
– 1-Auto-negotiation;
– 2-LPCM, output decoded signal;
– 3-RAW, output the original signal.

Return: None.

P.2.2 VideoSetting object

VideoSetting is a built-in object that provides a method for setting video parameters.

P.2.2.1 Constant

The definitions of the VideoSetting object constants are shown in Table P.2.

Table P.2 – VideoSetting object constants

Constants	Description
Video output channel	
const VOUT_SD = 1;	SD output channel
const VOUT_HD = 2;	HD output channel
Video window matching mode	
const MATCH_METHOD_LETTER_BOX = 1;	Mailbox mode (letter_box)
const MATCH_METHOD_PAN_SCAN = 2;	Full screen mode (pan_scan)
const MATCH_METHOD_COMBINED = 3;	Combination mode (combined)
const MATCH_METHOD_IGNORE = 4;	Ignore mode (ignore)
Video output system	
const VOUT_STANDARD_UNKNOWN = 0;	unknown
const VOUT_STANDARD_NTSC_J = 101;	SD-NTSC-J/3.5795MHz color subcarrier
const VOUT_STANDARD_NTSC_M = 102;	SD-NTSC-M/3.5795MHz color subcarrier
const VOUT_STANDARD_NTSC_443 = 103;	SD-NTSC-443/4.4336MHz color subcarrier
const VOUT_STANDARD_PAL_B = 211;	SD-PAL-B (Australia)
constVOUT_STANDARD_PAL_B1 = 212;	SD-PAL-B1 (Hungary)
const VOUT_STANDARD_PAL_D = 213;	SD-PAL-D (Mainland China)
const VOUT_STANDARD_PAL_D1 = 214;	SD-PAL-D1 (Poland)

Table P.2 – VideoSetting object constants

Constants	Description
const VOUT_STANDARD_PAL_G = 215;	SD-PAL-G (Europe)
const = 216;	SD-PAL-H (Europe)
constVOUT_STANDARD_PAL_I = 217;	SD-PAL-I (UK, Hong Kong, Macau)
const VOUT_STANDARD_PAL_K = 218;	SD-PAL-K (Europe)
const VOUT_STANDARD_PAL_M = 220;	SD-PAL-M (Brazil)
const VOUT_STANDARD_PAL_N = 221;	SD-PAL-N (Jamaica, Uruguay)
const VOUT_STANDARD_PAL_NC = 222;	SD-PAL-NC (Argentina)
constVOUT_STANDARD_SECAM_B = 311;	SD-SECAM-B
const VOUT_STANDARD_SECAM_D = 312;	SD-SECAM-D
const VOUT_STANDARD_SECAM_G = 313;	SD-SECAM-G
const = 314;	SD-SECAM-I
const VOUT_STANDARD_SECAM_K = 315;	SD-SECAM-K
constVOUT_STANDARD_SMPTE274_1080I_50 = 27400;	HD-SMPTE274/1920x1080I/50HZ/1125 lines
const VOUT_STANDARD_SMPTE274_1080I_59_94 = 27401;	HD-SMPTE274/1920x1080I/59.94HZ/1125 lines
const VOUT_STANDARD_SMPTE274_1080I_60 = 27402;	HD-SMPTE274/1920x1080I/60HZ/1125 lines
const V= 27410;	HD-SMPTE274/1920x1080P/23.98HZ/1125 lines
const VOUT_STANDARD_SMPTE274_1080P_24 = 27411;	HD-SMPTE274/1920x1080P/24HZ/1125 lines
constVOUT_STANDARD_SMPTE274_1080P_25 = 27412;	HD-SMPTE274/1920x1080P/25HZ/1125 lines
const VOUT_STANDARD_SMPTE274_1080P_29_97 = 27413;	HD-SMPTE274/1920x1080P/29.97HZ/1125 lines
const OUT_STANDARD_SMPTE274_1080P_30 = 27414;	HD-SMPTE274/1920x1080P/30HZ/1125 lines
const VOUT_STANDARD_SMPTE274_1080P_50 = 27415;	HD-SMPTE274/1920x1080P/50HZ/1125 lines
const = 27416;	HD-SMPTE274/1920x1080P/59.94HZ/1125 lines
constVOUT_STANDARD_SMPTE274_1080P_60 = 27417;	HD-SMPTE274/1920x1080P/60HZ/1125 lines
const VOUT_STANDARD_SMPTE295_1080I_50 = 29500;	HD-SMPTE295/1920x1080I/50HZ/1250 lines
const VOUT_STANDARD_SMPTE295_1080P_50 = 29510;	HD-SMPTE295/1920x1080P/50HZ/1250 lines
const VOUT_STANDARD_SMPTE296_720P_23_98 = 29610;	HD-SMPTE296/1280x720P/23.98HZ/750 lines
const VOUT_STANDARD_SMPTE296_720P_24 = 29611;	HD-SMPTE296/1280x720P/24HZ/750 lines

Table P.2 – VideoSetting object constants

Constants	Description
constVOUT_STANDARD_SMPTE296_720P_25 = 29612;	HD-SMPTE296/1280x720P/25HZ/750 lines
const VOUT_STANDARD_SMPTE296_720P_29_97 = 29613;	HD-SMPTE296/1280x720P/29.97HZ/750 lines
const VOUT_STANDARD_SMPTE296_720P_30 = 29614;	HD-SMPTE296/1280x720P/30HZ/750 lines
const VOUT_STANDARD_SMPTE296_720P_50 = 29615;	HD-SMPTE296/1280x720P/50HZ/750 lines
const = 29616;	HD-SMPTE296/1280x720P/59.94HZ/750
constVOUT_STANDARD_SMPTE296_720P_60 = 29617;	HD-SMPTE296/1280x720P/60HZ/750 lines
const VOUT_STANDARD_2160P_24 = 29618;	HD-3840x2160P/24HZ
const VOUT_STANDARD_2160P_25 = 29619;	HD-3840x2160P/25HZ
const VOUT_STANDARD_2160P_30 = 29620;	HD-3840x2160P/30HZ
const VOUT_STANDARD_2160P_50 = 29621;	HD-3840x2160P/50HZ
constVOUT_STANDARD_2160P_60 = 29622;	HD-3840x2160P/60HZ
constVOUT_STANDARD_4096P_24 = 29623;	HD-4096x2160P/24HZ

P.2.2.2 Method

P.2.2.2.1 getOutputInterfaceList

Prototype: string[] getOutputInterfaceList()

Description: Getting a list of all available video output ports of the receiving terminal.

Parameter: None.

Return: string array, indicating the names of all available video output interfaces of the receiving terminal, such as "CVBS", "YUV", "HDMI-0", "HDMI-1", "DVO", etc. If no video output port is available, the length of the returned array is 0.

P.2.2.2.2 getOutputInterfaceStatus

Prototype: boolean getOutputInterfaceStatus(port)

Description: Getting the enabling status of the video output port.

Parameter: port – string type, indicating the name of the video output interface, obtained by the getOutputInterfaceList() method.

Return: boolean type, true value indicating the video output interface allows output, and false value indicating that the video output interface prohibits output.

P.2.2.2.3 disableOutputInterface

Prototype: boolean disableOutputInterface(port)

Description: Disable video output port output.

Parameter: port – string type, indicating the name of the video output interface, obtained by the getOutputInterfaceList() method.

Return: boolean type, true value indicating the prohibition is successful, and false value indicating that the prohibition fails.

P.2.2.2.4 enableOutputInterface

Prototype: boolean enableOutputInterface(port)

Description: Allow video output port output.

Parameter: port – string type, indicating the name of the video output interface, obtained by the getOutputInterfaceList() method.

Return: boolean type, true value indicating success is allowed, and false value indicating that failure is allowed.

P.2.2.2.5 getOutputBrightness

Prototype: number getOutputBrightness()

Description: Getting the brightness of the video output.

Parameter: None.

Return: number type, indicating the brightness of the video output. The value range being from 0 to 100, 0 indicating the darkest, and 100 indicating the brightest.

P.2.2.2.6 getOutputContrast

Prototype: number getOutputContrast()

Description: Getting the contrast of the video output.

Parameter: None.

Return: number type, indicating the contrast of the video output, and the value range being 0-100 from small to large.

P.2.2.2.7 getOutputSaturation

Prototype: number getOutputSaturation()

Description: Getting the saturation (chromaticity) of the video output.

Parameter: None.

Return: number type, indicating the saturation (chromaticity) of the video output, and the value range being 0-100 from small to large.

P.2.2.2.8 getOutputStandard

Prototype: number getOutputStandard(device)

Description: Getting the video output format.

Parameter: device – number type, indicating the video output channel, the value is VOUT_SD or VOUT_HD.

Return: number type, indicating the format of the video output.

P.2.2.2.9 getOutputTransparency

Prototype: number getOutputTransparency()

Description: Getting the transparency of the video output.

Parameter: None.

Return: number type, indicating transparency, the value range being 0-100, 0 indicating completely opaque, and 100 indicating completely transparent.

P.2.2.2.10 setOutputBrightness

Prototype: boolean setOutputBrightness(value)

Description: Setting the brightness of the video output. This setting cannot be applied to a single video output unit, and it takes effect for all video output units at the same time.

Parameter: value – number type, indicating the brightness of the video output, the value range being 0-100 from small to large, 0 indicating the darkest, and 100 indicating the brightest.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.11 setOutputContrast

Prototype: boolean setOutputContrast(value)

Description: The contrast of the video output. This setting cannot be applied to a single video output unit, and it takes effect for all video output units at the same time.

Parameter: value – number, indicating the contrast of the video output, the value range being 0-100 from small to large.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.12 setOutputSaturation

Prototype: boolean setOutputSaturation(value)

Description: Setting the saturation (chroma) of the video output. This setting cannot be applied to a single video output unit, and it takes effect for all video output units at the same time.

Parameter: value – number, indicating the saturation (chroma), and the value range being 0-100 from small to large.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.13 setOutputStandard

Prototype: number setOutputStandard(device, standard)

Description: Setting the video output format. The SD output unit and HD output unit need to be set separately.

Parameter: device – number type, indicating the video output unit;

standard – number type, indicating the video output format. The SD output unit can only choose the SD format, and the HD output unit can only choose the HD format.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.14 setOutputTransparency

Prototype: boolean setOutputTransparency(value)

Description: Setting the transparency of the video output. This method cannot be set for a single video output unit, and it is effective for all video output units.

Parameter: value – number, indicating transparency, the value range being 0-100, 0 indicating completely opaque, and 100 indicating completely transparent.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.15 setOutputMatchMethod

Prototype: setOutputMatchMethod(mode)

Description: Setting the video output window matching mode.

Parameter: mode – number type.

Return: None.

P.2.2.2.16 getOutputMatchMethod

Prototype: number getOutputMatchMethod()

Description: Getting the matching mode of the video output window.

Parameter: None.

Return: number type.

P.2.2.2.17 getOutputAspectRatio

Prototype: getOutputAspectRatio()

Description: Getting the aspect ratio of the video output.

Parameter: None.

Return: number type, 0 indicating 16:9; 1 indicating 4:3.

P.2.2.2.18 setOutputAspectRatio

Prototype: setOutputAspectRatio(mode)

Description: Setting the video output aspect ratio.

Parameter: mode – number type, 0 indicating 16:9; 1 indicating 4:3.

Return: None.

P.2.2.2.19 GetColorSpaceMode

Prototype: number GetColorSpaceMode()

Description: Getting the color space mode.

Parameter: None.

Return: number type, 0-RGB444, 1-YCBCR422, 2 -YCBCR444, 3-YCBCR420.

P.2.2.2.20 GetDeepColorMode

Prototype: number GetDeepColorMode()

Description: Getting the color space mode.

Parameter: None.

Return: number type, 0-COLOR_24BIT, 1-COLOR_30BIT, 2-COLOR_36BIT, 3-COLOR_DEEP_OFF.

P.2.2.2.21 SetColorSpaceAndDeepColor

Prototype: boolean SetColorSpaceAndDeepColor(colorSpace, deepColor)

Description: Setting the color space, dark mode.

Parameter: colorSpace – number type, 0-RGB444, 1-YCBCR422, 2-YCBCR444, 3-YCBCR420

deepColor – number type, 0-COLOR_24BIT, 1-COLOR_30BIT, 2-COLOR_36BIT, 3-COLOR_DEEP_OFF

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.22 GetHDRType

Prototype: number GetHDRType()

Description: Getting HDR mode.

Parameter: None.

Return: number type, 0-HDRTYPE_SDR, 1-HDRTYPE_DOLBY, 2-HDRTYPE_HDR10, 3-C HDRTYPE_AUTO.

P.2.2.2.23 SetHDRType

Prototype: boolean SetHDRType(type)

Description: Setting HDR mode.

Parameter: type – number type, 0-HDRTYPE_SDR, 1-HDRTYPE_DOLBY, 2-HDRTYPE_HDR10, 3-C HDRTYPE_AUTO

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.24 GetStereoOutMode

Prototype: number GetStereoOutMode()

Description: Getting the 3D output mode.

Parameter: None.

Return: number type, 0-3D_NONE, 1-3D_FRAME_PACKING, 2-3D_SIDE_BY_SIDE_HALF, 3-3D_TOP_AND_BOTTOM, 4-3D_FIELD_ALTERNATIVE, 5-3D_LINE_ALTERNATIVE, 6-3D_SIDE_BY_SIDE_FULL, 7-3D_L_DEPTH, 8-3D_L_DEPTH_GRAPHICS_GRAPHICS_DEPTH.

P.2.2.2.25 SetStereoOutMode

Prototype: boolean SetStereoOutMode(mode, fps)

Description: Setting the 3D output mode.

Parameter: mode – number type, 0-3D_NONE, 1-3D_FRAME_PACKING, 2-3D_SIDE_BY_SIDE_HALF, 3-3D_TOP_AND_BOTTOM, 4-3D_FIELD_ALTERNATIVE, 5-3D_LINE_ALTERNATIVE, 6-3D_SIDE_BY_SIDE_FULL, 7-3D_L_DEPTH, 8-3D_L_DEPTH_GRAPHICS_GRAPHICS_DEPTH

Fps – number type, video frame rate, the values are 23, 24, 25, 30, 50, 59, 60.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.26 GetRightEyeFirst

Prototype: number GetRightEyeFirst()

Description: Getting which eye of the 3D output signal is to come out first.

Parameter: None.

Return: number type, the values are 0-LEFT_EYE_FIRST, 1-RIGHT_EYE_FIRST.

P.2.2.2.27 SetRightEyeFirst

Prototype: boolean SetRightEyeFirst(Outpriority)

Description: Setting which eye of the 3D output signal is to come out first.

Parameter: Outpriority – number type, 0-LEFT_EYE_FIRST, 1-RIGHT_EYE_FIRST

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.28 GetStereoDepth

Prototype: number GetStereoDepth()

Description: Getting 3D picture depth adjustment information.

Parameter: None.

Return: number type, the value being 0-10.

P.2.2.2.29 SetStereoDepth

Prototype: boolean SetStereoDepth(depth)

Description: Setting 3D picture depth adjustment information.

Parameter: depth – number type, 0-10.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.30 getPictureMode()

Prototype: number getPictureMode()

Description: Getting the picture mode.

Parameter: None.

Return: number type, 0 standard, 1 dynamic, 2 soft, 4 users, 5 gorgeous, 6 natural, 7 sports.

P.2.2.2.31 setPictureMode

Prototype: boolean setPictureMode(mode)

Description: Setting the picture mode.

Parameter: mode – number type, 0 standard, 1 dynamic, 2 soft, 4 users, 5 gorgeous, 6 natural, 7 sports.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.32 getDisplayHue

Prototype: number getDisplayHue()

Description: Getting the picture mode.

Parameter: None.

Return: number type, the value range being 0-100, indicating the color adjustment value.

P.2.2.2.33 setDisplayHue

Prototype: boolean setDisplayHue(hue)

Description: Setting the picture mode.

Parameter: hue – number type, the value range being 0-100, indicating the color adjustment value.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.34 SaveDisplayFmt

Prototype: boolean SaveDisplayFmt()

Description: Saving the video output format permanently effective.

Parameter: None.

Return: boolean type, true value indicating success, false value indicating failure.

P.2.2.2.35 setOptimalFormatEnable

Prototype: boolean setOptimalFormatEnable(enabled)

Description: Setting automatic optimization video output format enable.

Parameter: enabled – number type, 0-indicating disable, 1-indicating enable.

Return: boolean type, true value indicating the setting is successful, false value indicating the setting fails.

P.2.2.2.36 getOptimalFormatEnable

Prototype: number getOptimalFormatEnable()

Description: Getting the automatic optimization video output format enable.

Parameter: None – number type, 0-indicating disable, 1-indicating enable.

Return: number type, 0 indicating disable, 1 indicating enable.

Annex Q

JavaScript-Media processing unit

(This annex forms an integral part of this Recommendation.)

Q.1 Overview

This annex defines a functional module related to media processing: media playback module.

Q.2 Media playback module

This module defines the JS object related to media playReturn: MediaPlayer.

Q.2.1 Message

Definitions of messages sent by the media player module to the application layer is shown in the following table, and the message received on the application uses document.onsystemevent.

Table Q.1 – Messages of media playback module

Name of messages	event.which	event.modifiers	Description of messages
MSG_MEDIA_URL_VALID	13001	–	The media source path is valid.
MSG_MEDIA_URL_INVALID	13002	–	The media source path is invalid.
MSG_MEDIA_PLAY_SUCCESS	13003	–	Playback started successfully.
MSG_MEDIA_PLAY_FAILED	13004	–	Failed to start playback.
MSG_MEDIA_SETPACE_SUCCESS	13005	–	The step size is set successfully.
MSG_MEDIA_SETPACE_FAILED	13006	–	Failed to set the step size.
MSG_MEDIA_SEEK_SUCCESS	13007	–	Playback time point set successfully.
MSG_MEDIA_SEEK_FAILED	13008	–	Failed to set the playback time point.
MSG_MEDIA_PAUSE_SUCCESS	13009	–	Playback paused successfully.
MSG_MEDIA_PAUSE_FAILED	13010	–	Failed to pause the playback.
MSG_MEDIA_RESUME_SUCCESS	13011	–	Playback resumed successfully.
MSG_MEDIA_RESUME_FAILED	13012	–	Failed to resume the playback.
MSG_MEDIA_STOP_SUCCESS	13013	–	Playback stopped successfully.
MSG_MEDIA_STOP_FAILED	13014	–	Failed to stop the playback.
Reserved	13015～13200		

Table Q.1 – Messages of media playback module

Name of messages	event.which	event.modifiers	Description of messages
The value of event.modifiers is automatically given by the system, and its data type: – "Number", indicating that the value is the ID of the message description character string, which can be obtained through the Utility.getEventInfo() method. If the "message description" defines the JSON format of the message character string, the message content will be retrieved according to the format. – "–", indicating that event.modifiers is undefined.			

Q.2.2 MediaPlayer object

The MediaPlayer object is a local object and needs to be created "new" before use.

This object defines the properties and methods for media playback in the Web page. The media source can be TV broadcast, sound broadcast or NVOD, it can also be a UDP unicast or multicast stream, or it can be a media file stored locally in the receiving terminal. This object only needs to know the media type and the location (URL) of the media to play this media.

The display of the video layer must be rendered transparent by the video element. In the page layout, the video element specifies the position, size, and level of the video on the page. The page must use <video src=""></video>, src Empty, for example: <video id="video" src="" style="position: absolute; z-index:18; top:200px; left:180px; width:400px; height:400px;"></video >

After the MediaPlayer object is constructed, it can only be used in a single web page, and is uniquely identified by the object name; the player instance corresponds to the media decoding resource of the receiving terminal, and is not bound to the web page, and can be used across pages, and is uniquely identified through the player instance ID, the player instance ID is automatically generated by the system. The browser must control the player instance to achieve media playback through the properties and methods provided by the MediaPlayer object, and one MediaPlayer object can only be bound to one player instance, and one player instance can only be bound to one MediaPlayer object at the same time. The relationship between the MediaPlayer object and the player instance is shown in Figure Q.1.

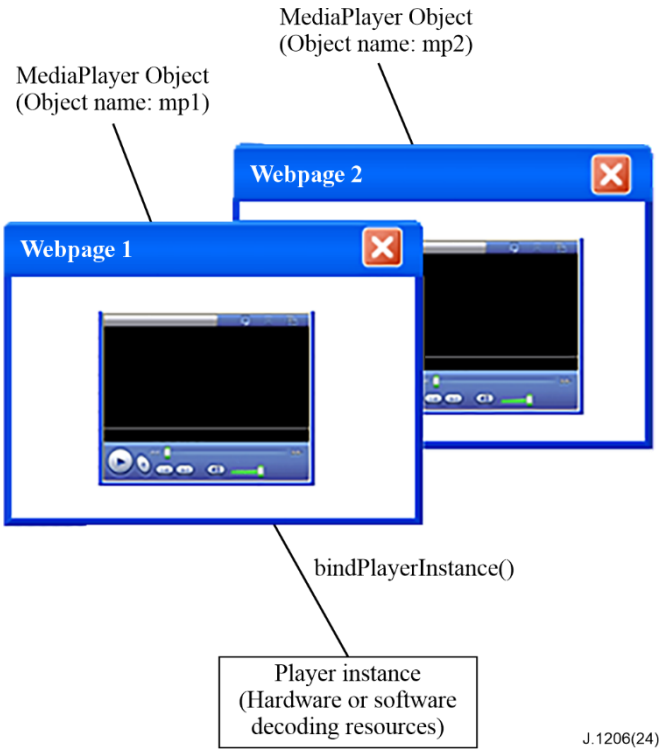


Figure Q.1 – Schematic diagram of the relationship between the MediaPlayer object and the media playback instance

Example:

The MediaPlayer object created in the web page can control the player instance on the receiving terminal, and the life cycle of the player instance is cross-page. Identify the currently bound player instance through the playerInstanceID property of the MediaPlayer object.

```
//In the first web page, create a MediaPlayer object mp1
var mp1 = new MediaPlayer();
//Read the local media player instance ID
var nativePlayerInstanceID = mp1.getPlayerInstanceID();
//Reserve this player instance identifier through global variables for cross-page use
GlobalVarManager.setItemValue("PLAYER_INSTANCE_ID", nativePlayerInstanceID);
//The MediaPlayer object is bound to the player instance
    mp1.bindPlayerInstance(nativePlayerInstanceID);
    mp1.setMediaSource(mediaURL); //Set media source
    mp1.play();//Start playing
...
mp1.setPace(2);  //fast forward, 2x speed playback
...
mp1.setPace(1);  //resume playback at normal speed
...
    mp1.pause(); //pause playback
    mp1.resume();//resume playback
```

```
mp1.stop(); //stop playback
mp1.unbindPlayerInstance(nativePlayerInstanceID);
//In the next web page, create a MediaPlayer object mp2
var mp2 = new MediaPlayer();
//Getting the player instance ID of the previous page through global variables
var nativePlayerInstanceID = GlobalVarManager.getItemValue("PLAYER_INSTANCE_ID");
//According to the player instance ID passed from the previous page, bind the MediaPlayer object and the media player instance
mp2.bindPlayerInstance(nativePlayerInstanceID);
mp2.setMediaSource(mediaURL); //Set media source
mp2.play(); //start playback
mp2.pause(); //pause playback
mp2.resume();//resume playback
mp2.stop(); //stop playback
mp2.unbindPlayerInstance(nativePlayerInstanceID);
```

Q.2.2.1 Media playback status

The media player has the following playback status: initialization, binding, stop, playing (forward/backward/fast forward/slow forward/fast backward/slow backward), pause and unbinding. The status transition diagram is shown in Figure Q.2.

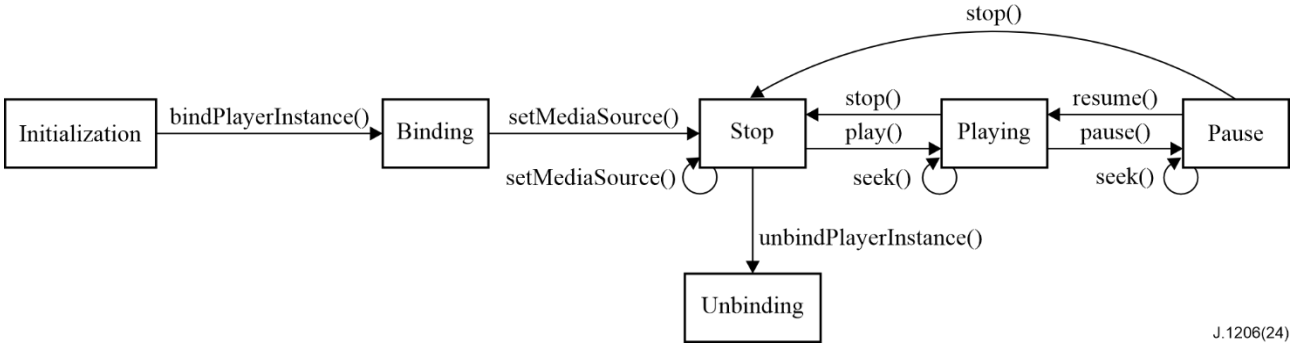


Figure Q.2 – Schematic diagram of media playback status transition

The various statuses of media playback in Figure Q.2 are described as follows:

– Initialization status: When the MediaPlayer object is created, it enters the initialization status.

– Binding status: When the MediaPlayer object calls the bindPlayerInstance() method, it enters the binding status.

– Stop status: When the MediaPlayer object calls the setMediaSource() or stop() method, it enters the stop status. At this time, the video and audio stop playing, and the video screen is in a hidden state.

– Playback status: When the MediaPlayer object calls the play() method, it enters the playback status, including forward, backward, fast forward, slow forward, fast backward, and slow backward. The relationship between the playback step size and the playback status is shown in Table Q.2.

Table Q.2 – Relationship between the playback step size and the playback status

Playback status	Step size (pace)	Description
Forward	1	When the play() method is used to start playback, pace = 1, it enters the forward status, and the default step size is 1.
Backward	−1	When the play() method is used to start playback, pace = −1, if the media is not played from the beginning, it will enter the backward status.
Fast forward	2, 4, 8, 16, 32	When the play() method is used to start playback, and pace = 2, 4, 8, 16, 32, it enters the fast forward status.
Slow forward	1/2, 1/4, 1/8	When the play() method is used to start playback, and pace = 1/2, 1/4, 1/8, it enters the slow forward status.
Fast backward	−2, −4, −8, −16, −32	When the play() method is used to start playback, and pace = −2, −4, −8, −16, −32, if the media is not played from the beginning, it will enter the fast backward status.
Slow backward	−1/2, −1/4, −1/8	When the play() method is used to start playback, and pace = −1/2, −1/4, −1/8, if the media is not played from the beginning, it will enter the slow backward status.

– Pause status: When the MediaPlayer object calls the pause() method, it can enter the pause status; in the pause status, the video picture continues to occupy the screen space, and it can be set to a static frame (default) or black field.

– Unbinding status: When the MediaPlayer object calls the unbindPlayerInstance() method, it can enter the unbinding status. Release resources.

The played media sources have the following forms:

– Broadcasting services, including TV services, audio broadcasting services, NVOD services, etc.;

– Interactive TV services, including video-on-demand, time-shifted TV and channel review, etc.;

– The code stream sent to the terminal through IP-UDP;

– Media files stored locally on the terminal.

The playback status and playback step size supported by different sources of media are shown in Table Q.3.

Table Q.3 – Table of media playback status and step size

Media source	Initialization	Binding	Ready	Playback						Pause	Stop	Unbinding
				Forward	Backward	Fast-forward	Fast-backward	Slow forward	Slow backward			
Broadcasting service	√	√	√	√	N/A	N/A	N/A	N/A	N/A	√	√	√
Interactive TV service	√	√	√	√	√	√	√	N/A	N/A	√	√	√
IP-UDP coding stream	√	√	√	√	N/A	N/A	N/A	N/A	N/A	√	√	√

Table Q.3 – Table of media playback status and step size

Media source	Initialization	Binding	Ready	Playback						Pause	Stop	Unbinding
				Forward	Backward	Fast-forward	Fast-backward	Slow forward	Slow backward			
Local media files	√	√	√	√	√	√	√	√	√	√	√	√

Q.2.2.2 Property

The definition of the media player object property is shown in Table Q.4.

Table Q.4 – Properties of the MediaPlayer object

Property name	Type	Read and write property	Description
Location	string	Read only	It indicates the locator of the media file.
playerInstanceID	number	Read only	It indicates the player instance ID bound to the current MediaPlayer object. If the value is −1, it indicates that the current MediaPlayer object has not been bound to any player instance. For example, before calling bindPlayerInstance() method, or after calling unbindPlayerInstance() method, read this property, it should return -1.

Q.2.2.3 Method

Q.2.2.3.1 MediaPlayer

Prototype: MediaPlayer()

Description: Construction method, create a default MediaPlayer object.

Parameter: None.

Q.2.2.3.2 getPlayerInstanceID

Prototype: number getPlayerInstanceID()

Description: Getting the player instance ID available locally on the receiving terminal, which is automatically assigned by the system.

Parameter: None.

Return: number type, returning 0-255 if it succeeds, and returning −1 if it fails.

Q.2.2.3.3 bindPlayerInstance

Prototype: number bindPlayerInstance(playerInstanceID)

Description: The MediaPlayer object is bound to the player instance.

Parameter: playerInstanceID − number type, indicating the local player instance ID, the value range being 0-255.

Return: number type, returning 0 if it succeeds, and returning −1 if it fails.

Q.2.2.3.4 unbindPlayerInstance

Prototype: number unbindPlayerInstance(playerInstanceID)

Description: The MediaPlayer object is unbound with the current player instance, and the related resources of the player are released.

Parameter: playerInstanceID – number type, indicating the local player instance ID, the value range being 0-255.

Return: number type, returning 0 if it succeeds, and returning −1 if it fails.

Q.2.2.3.5 setMediaSource

Prototype: number setMediaSource(mediaURL)

Description: An Asynchronous method, setting the URL address of the media to be played. After setting this parameter, the system automatically detects the legality of the set mediaURL.

– If the URL is legal, send the message MSG_MEDIA_URL_VALID to the page;

– If the URL is illegal, send the message MSG_MEDIA_URL_INVALID to the page.

Only after receiving the message MSG_MEDIA_URL_VALID, the page can call the play() method to play.

Parameter: mediaURL – string type, media path expressed in URL format, description of the URL format is shown in Table Q.5.

Table Q.5 – Format of media source

Media source	mediaURL	Description
Broadcasting service	dvb://<original_network_id>.<transport_stream_id>.<service_id>[.<component_tag>{&<component_tag>}][;event_id>]{/<path-element>}	It can be used to access: – Broadcasting services; – Service components, such as elementary streams such as video, audio, or subtitles; – Events of broadcasting service; – Files in OC/DC carousel.
Broadcasting service	dvbelement://frequency.symbolrate.modulation.serviceid.pmtpid.pcrpid.videotype.videopid.audiotype.audiopid	The program information such as frequency point is specified to play to meet the rapid start of broadcast. The frequency is measured in kilohertz, and all parameters are decimal. For multi-tuner scenes, expand to dvbelement://frequency.symbolrate.modulation.serviceid.pmtpid.pcrpid.videotype.videopid.audiotype.audiopid?tunerid=xxx.
Broadcasting service	dvbelement://frequency.symbolrate.modulation.serviceid.-1.-1.-1.-1.-1.-1	Play by specifying the frequency and other program information to meet the need for scrambled stream and multi-track playback when the program is not searched. The format is dvbelement://frequency.symbolrate.modulation.serviceid.-1.-1.-1.-1.-1.-1. In this scene, the media engine parses pmtpid from the code stream according to serviceid, and parses pcrpid, videotype, videopid, audiotype, and audiopid according to pmtpid. If pmtpid exists and is valid, it does not search for pmtpid, and parses pcrpid, videotype, videopid, audiotype, and

Table Q.5 – Format of media source

Media source	mediaURL	Description
		audiopid according to pmtpid. The frequency is measured in kilohertz, all numbers are decimal, and the invalid parameter is -1, such as dvbelement://131000.6875.64.3.-1.-1.-1.-1.-1.-1 or dvbelement://131000.6875.64.3.4.-1.-1.-1.-1.-1. For multi-tuner scenes, expand to dvbelement://frequency.symbolrate.modulation.serviceid.-1.-1.-1.-1.-1.-1?tunerid=xxx.
Interactive TV service	rtsp://<Session manager IP>:<Session manager port>/;EntitlementCode=<EntitlementCode>	It can be used to access: – Video on demand; – Time-shifted TV; – Channel review.
IP-UDP stream	udp://MulticastAddress:UDPPort:Programnumber:VideoPID:AudioPID	– When the front end uses UDP unicast, MulticastAddress is "0.0.0.0"; when the front end uses UDP multicast, MulticastAddress is the sending multicast address; – UDPPort is the UDP port number; – Programnumber is the service ID number; – VideoPID is the video PID; – AudioPID is the audio PID; VideoPID can be filled with 0, and the system defaults to the video stream with the smallest PID value under the service. If the service is a pure audio service, the value is 0; AudioPID can be filled with 0, and the system defaults to the audio stream with the smallest PID value under the service.
Media files	file://<host>/<path>	Access local media files.
HTTP	http://< remote host IP>:<remote host port>/<path>	Access remote resource files.

Return: number type, returning 0 if successful, otherwise returning others.

Q.2.2.3.6 play

Prototype: number play()

Description: An Asynchronous method, starting playing from the starting point of the media, and starting playing in real time for TV broadcasts.

– If the playback is successful, send the message MSG_MEDIA_PLAY_SUCCESS to the page;
– If the playback fails, send the message MSG_MEDIA_PLAY_FAILED to the page.

Return: number type, 1-indicating success, 0-indicating failure.

Q.2.2.3.7 seek

Prototype: number seek(type, timestamp)

Description: An Asynchronous method, starting to play from a certain point in time of the current media, this call is invalid for real-time broadcast TV broadcasts, but it is valid for TV broadcasts in a time-shifted state.

– If the call is successful, send the message MSG_MEDIA_SEEK_SUCCESS;

– If the call fails, send the message MSG_MEDIA_SEEK_FAILED.

NOTE – The seek() method can be called only in the play and pause states.

Parameter: type – number type, the values are:

– 1-indicating Normal Play Time;

– 2-indicating Absolute Time.

timestamp – two time type formats of Normal Play Time (NPT) and Absolute Time (ClockTime).

– For VOD, it is relative time calculated from the starting point of the media;

– For media with a time base such as time shift, it is absolute time.

It can indicate the start time (for example: "123-"), or it can indicate the time period (for example: "123-333"). If it is a time period, the playback should be in a pause state instead of in a stop state, when it reaches the end time.

Return: number type, returning 0 if successful, otherwise return others.

Q.2.2.3.8 setPace

Prototype: number setPace(pace)

Description: An Asynchronous method, setting the step size of the playback.

– If the setting is successful, send the message MSG_MEDIA_SET_PACE_SUCCESS to the page;

– If the setting fails, send the message MSG_MEDIA_SET_PACE_FAILED to the page.

NOTE – The setPace() method can be called only in the playing state.

Parameter: pace – number type, play the step size.

Return: number type, returning 0 if successful, otherwise return others.

Q.2.2.3.9 pause

Prototype: number pause()

Description: An Asynchronous method, pausing playback.

– If the pause is successful, send the message MSG_MEDIA_PAUSE_SUCCESS to the page;

– If the pause fails, send the message MSG_MEDIA_PAUSE_FAILED to the page.

Parameter: None.

Return: number type, returning 0 if successful, otherwise return others.

Q.2.2.3.10 resume

Prototype: number resume()

Description: An Asynchronous method, resuming playback.

– If resuming of the playback is successful, send the message MSG_MEDIA_RESUME_SUCCESS to the page;

– If the resuming of the playback fails, send the message MSG_MEDIA_RESUME_FAILED to the page.

Parameter: None.

Return: number type, returning 0 if successful, otherwise return others.

Q.2.2.3.11 stop

Prototype: number stop()

Description: An Asynchronous method, pausing playing.

– If the stop is successful, send the message MSG_MEDIA_STOP_SUCCESS to the page;

– If the stop fails, send the message MSG_MEDIA_STOP_FAILED to the page.

Parameter: None.

Return: number type, returning 0 if successful, otherwise return other.

Q.2.2.3.12 refresh

Prototype: number refresh()

Description: Calling setVideoDisplayMode(), setVideoDisplayArea() and other properties to adjust the video parameters does not take effect immediately, but takes effect only upon this method being called.

Parameter: None.

Return: number type, returning 0 if successful, otherwise return others.

Q.2.2.3.13 enableTrickMode

Prototype: number enableTrickMode(flag)

Description: Setting whether the player instance currently bound to the MediaPlayer object allows trick operations (including fast forward/fast backward/pause, etc.) during its life cycle, which is in a logical AND relationship with the trick mode property of the media itself.

For example, if the ad in the first 30 seconds of a stream cannot perform trick operations, enableTrickMode(0) can be called during this time period, and the trick is disabled; after 30 seconds, enableTrickMode (1) can be called again in the web page to start the trick function. If the media source being played does not support the trick mode, this method is invalid.

Parameter: flag – boolean type, false value indicating that the trick operations are prohibited (default value); true value indicating that the trick operations are allowed.

Return: number type, returning 0 if successful, otherwise return others.

Q.2.2.3.14 getTrickModeFlag

Prototype: boolean getTrickModeFlag()

Description: Getting the operation flag of the trick mode.

Parameter: None.

Return: boolean type, true value indicating the trick operations are allowed, and false value indicating that the trick operations are not allowed (default value).

Q.2.2.3.15 setVideoDisplayMode

Prototype: number setVideoDisplayMode(mode)

Description: Setting the display mode of the video window corresponding to the MediaPlayer object. Every time this method is called, the video display window will not immediately display the set mode, and the window isrefreshed to display the set mode only upon the refresh() method being called.

Parameter: mode – number type, video display mode, the values are as follows:

– 0-Display according to the position and size specified by the properties of height, width, left and top set in the setVideoDisplayArea() method;
– 1-Full screen display, display according to full screen height and width (default value);
– 2-Width display, which refers to the full-screen width display without changing the aspect ratio of the original image;
– 3-Height display, which refers to the full-screen height display without changing the aspect ratio of the original image;
– 255-The video display window will be closed. It will hide the video window while maintaining the media stream connection. If the media playback does not stop, it will continue to play.

Return: number type, returning 0 if successful, otherwise returning other values.

Q.2.2.3.16 getVideoDisplayMode

Prototype: number getVideoDisplayMode()

Description: Getting the display mode of the video window corresponding to the MediaPlayer object.

Parameter: None.

Return: number type, the values are as follows:

– 0-Display according to the position and size specified by the properties of height, width, left and top set in the setVideoDisplayArea() method;
– 1-Full screen display, display according to full screen height and width (default value);
– 2-Width display, which refers to the full-screen width display without changing the aspect ratio of the original image;
– 3-Height display, which refers to the full-screen height display without changing the aspect ratio of the original image;
– 255-The video display window will be closed. It will hide the video window while maintaining the media stream connection. If the media playback does not stop, it will continue to play.
– Other: The display mode is illegal.

Q.2.2.3.17 setVideoDisplayArea

Prototype: number setVideoDisplayArea(rect)

Description: Setting the window display area. Every time this method is called, the video display window is not displayed in the set area immediately, and the video is displayed in the set area only after the refresh() method is called. When the video needs to be displayed in the area set by the setVideoDisplayArea() method, setVideoDisplayMode(0) needs to be called; but there is no order in which the two are called, just before calling the refresh() method.

Parameter: rect – Rectangle object, describing the location information of the display area.

Return: number type, returning 0 if successful, otherwise return others.

Q.2.2.3.18 getVideoDisplayArea

Prototype: Rectangle getVideoDisplayArea()

Description: Getting the position information of the display area of the video window.

Parameter: None.

Return: A Rectangle object, indicating the location information of the display area of the video window.

Q.2.2.3.19 setVolume

Prototype: number setVolume(volume)

Description: Setting the current volume. If the broadcast service is currently being played, the increment of the channel relative to the global volume should be memorized.

NOTE – Volume increment = current volume-global volume.

Parameter: volume − number type, the value range being 0-100, 0 indicating minimum volume (mute), 100 indicating maximum volume.

Return: number type, returning 0 if successful, otherwise returning others.

Q.2.2.3.20 getVolume

Prototype: number getVolume()

Description: Getting the current volume.

Parameter: None.

Return: number type, the value range being 0-100, 0 indicating the minimum volume (mute), and 100 indicating the maximum volume.

Q.2.2.3.21 getCurrentLanguage

Prototype: string getCurrentLanguage()

Description: Getting the current language being used.

Parameter: None.

Return: string type, indicating the audio language, and the three-letter code of the language follows the GB/T 4880.2-2000 standard. If it cannot be obtained, undefined is returned.

Q.2.2.3.22 listAvailableLanguages

Prototype: string[] listAvailableLanguages()

Description: Getting all currently available audio languages.

Parameter: None.

Return: A String type array, indicating a list of audio languages. The three-letter language codes follow the GB/T 4880.2-2000 standard. If it cannot be obtained, an array of length 0 is returned.

Q.2.2.3.23 selectDefaultLanguage

Prototype: string selectDefaultLanguage()

Description: Setting the audio language as the default language.

Parameter: None.

Return: string type, indicating the default audio language, and the three-letter language code follows the GB/T 4880.2-2000 standard.

Q.2.2.3.24 selectLanguage

Prototype: selectLanguage(language)

Description: Setting the language of the current audio.

Parameter: language – string type, indicating the audio language, and the three-letter code of the language follows the GB/T 4880.2-2000 standard.

Return: None.

Q.2.2.3.25 getMediaDuration

Prototype: string getMediaDuration()

Description: Getting the total duration of the currently media playing.

Parameter: None.

Return: string type, the format is "hh:mm:ss".

– When the property of the media source is a local media file, this method returns the total duration of the media being played;
– When the media source is broadcast service, interactive TV service, or IP-UDP coding stream, this method returns undefined.

Q.2.2.3.26 getCurrentPlayTime

Prototype: string getCurrentPlayTime()

Description: Getting the current time point of media playback.

Parameter: None.

Return: string type, indicating the current time point of media playback, with two time types: Normal Play Time and Absolute Time.

– Normal Play Time format for on-demand;
– Absolute Time format for time shift.

Q.2.2.3.27 getPlaybackMode

Prototype: string getPlaybackMode()

Description: Getting the current play mode of the player.

Parameter: None.

Return: string type, indicating JSON character string, which includes at least two types of information: "PlayMode" and "Speed".

For the values, PlayMode can be Normal, Pause, Trickmode, when it is Trickmode, it must be 2x/-2x, 4x/-4x, 8x/-8x, 16x/-16x, 32x/-32x parameters to indicate the speed parameters of fast forward/fast backward. Examples of return values:

{"PlayMode":"Trickmode", "Speed":"2x"}.

Q.2.2.3.28 getServiceLocation

Prototype: string getServiceLocation(flag)

Description: Getting the locator of the specified program according to the program flag object.

Parameter: flag – number type, the value range is shown in Table Q.6.

Table Q.6 – Table of flag value

Flag	Description
0	It indicates the program being played.
1	It indicates the previous program played.
2	It indicates the previous television program played.
3	It indicates the previous broadcast program played.

Return: string type, indicating the locator (URL) of the service.

Q.2.2.3.29 setPauseMode

Prototype: setPauseMode(mode)

Description: Setting the output mode of the video in the pause state.

Parameter: mode – number type, 0 indicating a black screen, and 1 indicating a still frame.

Return: None.

Q.2.2.3.30 getPauseMode

Prototype: number getPauseMode()

Description: Getting the video output mode in the pause state.

Parameter: None.

Return: number type, 0 indicating a black screen, and 1 indicating a still frame.

Annex R

JavaScript-Aplication management unit

(This annex forms an integral part of this Recommendation.)

R.1 Overview

This annex defines the functional modules related to application management: application management module, the detailed definition of the application management module functions is shown in Table R.1.

Table R.1 – Definition of application management module function

Function	Description
Application download	1. The application can be downloaded to the local storage through multiple data channels such as http, https, DVB OC data push, etc. from the website or specified URL, APPSTORE; 2. Copy applications directly from other storage devices to local storage.
Application installation	1. The installation package can only be installed through the signature verification; 2. Decompress the installation package to the specified location in the local storage.
Application uninstall	1. Delete the installation package; 2. Delete the buffered data files stored locally by the application; 3. Delete the decompressed directory.
Application upgrade	1. Get the application list on the server; 2. Check the updated version on the server, and upgrade if there is a new version.
Application permission management	1. The application's access to system resources; 2. Application permissions need to be classified and prioritized; 3. The settings of application permissions can be modified and configured through the configuration file of the installation package; 4. Application startup and operation need to be carried out in the sandbox and also need permission for control.
Application information query	1. Query of application operating status information; 2. Access authority information for system resources owned by the application; 3. Version information of the application.
Application security	1. Ensure the legitimacy of the application source through digital signature; 2. The generated installation package database requires root permission control to prevent unauthorized users from modifying the installation package database and ensure the security of the file; 3. After decompressing the application files and directory tree, only the application manager has read and write permissions to prevent applications from reading each other and ensure the isolation between applications; 4. Startup and operation, and access to resources need to be controlled through permissions to ensure the safety of application operation.

Table R.1 – Definition of application management module function

Function	Description
Application startup and switching	1. Analyze the size, icon, authorization, multi-language, localization, startup of the HTML file, characteristics, text encoding, and resource location of each application through the configuration file; 2. Switch between application and launcher; 3. Switch between applications; 4. Management of focus; 5. Processing of independent applications.

R.2 Application management module

This module defines the JS object related to application management: widget.

R.2.1 window object extended property widget

```
interface partial Window : EventTarget {
  readonly attribute Widget widget;
};
```

R.2.2 widget object

The widget object is a built-in object and is used directly.

R.2.2.1 Overview

The widget object contains read-only properties, methods, and message feedback.

```
interface Widget {
readonly attribute DOMString   author;   //corresponding to the author element of config.xml
readonly attribute DOMString   description; //corresponding to the description element of config.xml
readonly attribute DOMString   name;   //corresponding to the name element of config.xml
readonly attribute DOMString   shortName;  //corresponding to the name short property of config.xml
readonly attribute DOMString   version;   //corresponding to the widget version property of config.xml
readonly attribute DOMString   id;   //corresponding to the widget id property of config.xml
readonly attribute DOMString   authorEmail; //corresponding to the author email property of config.xml
readonly attribute DOMString   authorHref; //corresponding to the author href property of config.xml
readonly attribute Storage   preferences; //corresponding to the preference element of config.xml
readonly attribute unsigned long height;   //corresponding to the widget height property of config.xml
readonly attribute unsigned long width;   //corresponding to the widget width property of config.xml
void launchWidget(DOMString widgetname, DOMString src, DOMString type);
```

void installWidget(DOMString widgetname, DOMString id, DOMString url); //Install widget

void updateWidget(DOMString widgetname, DOMString id, DOMString url); // Update widget

DOMString checkWidget(DOMString widgetname, DOMString id, DOMString url); //Check widget, return version number

void uninstallWidget(DOMString widgetname, DOMString id); //Uninstall widget

void deletePackage(DOMString widgetname, DOMString id); //Delete application package

//The following are several messages that the application manager feeds back to the application

attribute EventHandler onstart;

attribute EventHandler onresume;

attribute EventHandler onpause;

attribute EventHandler onstop;

attribute EventHandler onterminate;

attribute EventHandler ondownloadsuccess;

attribute EventHandler ondownloadfail;

attribute EventHandler ondownloadsize;

};

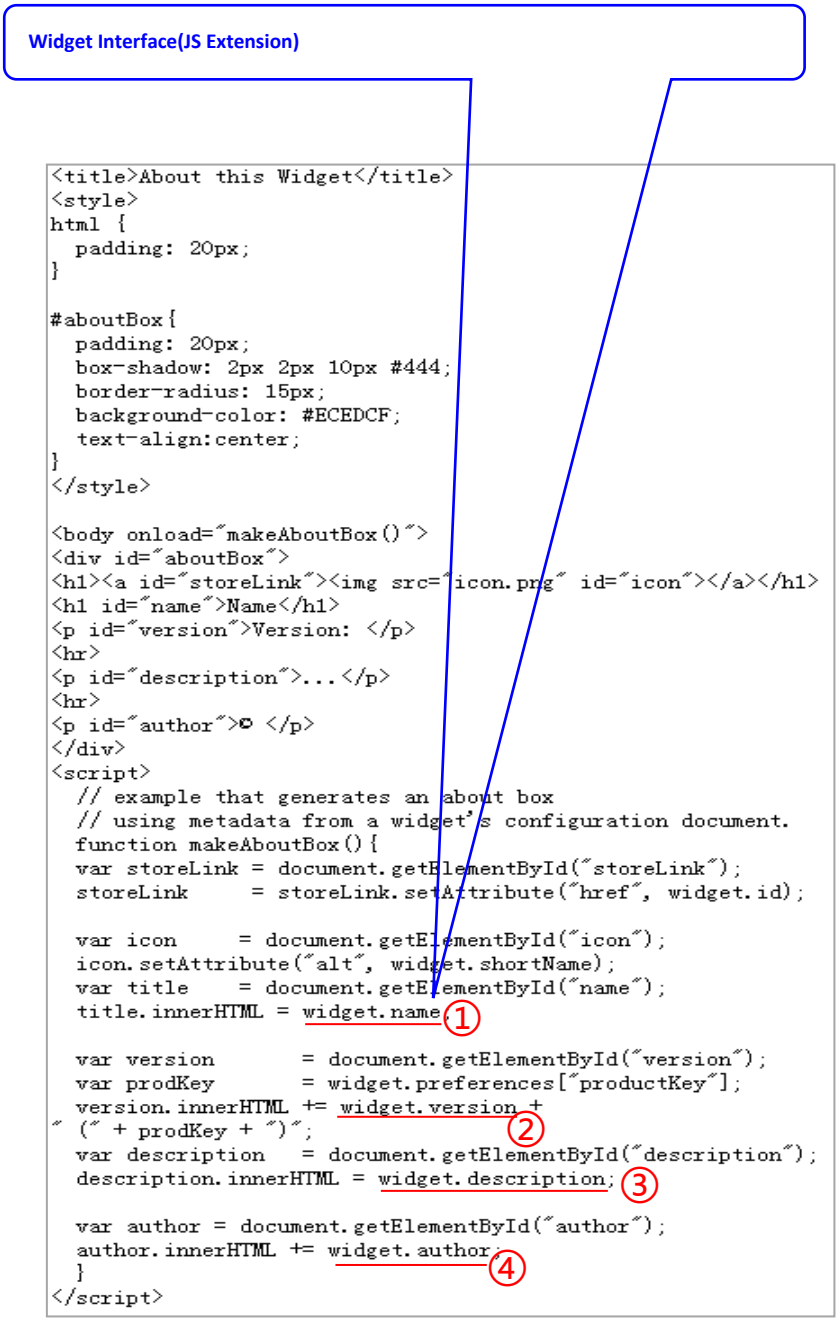
Example of use

The widget's config.xml configuration file is as follows:

Packaged Web Apps (Widgets) – Packaging and XML Configuration (Second Edition)

```
<widget xmlns      = "http://www.w3.org/ns/widgets"
        id         = "http://example.org/exampleWidget"
        version    = "2.0 Beta"
        height     = "200"
        width      = "200"
        viewmodes = "floating">

<name short="Example 2.0">The example Widget!</name>

<description>A sample widget to demonstrate some of the possibilities.</description>

<author href  = "http://foo-bar.example.org/"
       email = "foo-bar@example.org">Foo Bar Corp</author>

<preference name     = "apikey"
            value    = "ea31ad3a23fd2f"
            readonly = "true"/>

</widget>
```

The page code for accessing config.xml through JS widget API is as follows:



R.2.2.2 Property

The definition of the widget object property is shown in Table R.2.

Table R.2 – Properties of widget object

Property name	Type	Read/write property	Description
Author	string	Read only	corresponding to the author element of config.xml
description	string	Read only	corresponding to the description element of config.xml
Name	string	Read only	corresponding to the name element of config.xml
shortName	string	Read only	corresponding to the name short property of config.xml

Table R.2 – Properties of widget object

Property name	Type	Read/write property	Description
Version	string	Read only	corresponding to the widget version property of config.xml
Id	string	Read only	corresponding to the widget id property of config.xml
authorEmail	string	Read only	corresponding to the author email property of config.xml
authorHref	string	Read only	corresponding to the author href property of config.xml
preferences	Storage	Read only	corresponding to the preference element of config.xml
Height	Number unsigned long	Read only	corresponding to the widget height property of config.xml
width	Number unsigned long	Read only	corresponding to the widget width property of config.xml

R.2.2.3 Method

R.2.2.3.1 launchWidget

Prototype: void launchWidget(DOMString widgetname, DOMString src, DOMString type)

Description: Start the widget on the launcher or return the widget to the launcher.

Parameter: widgetname – DOMString type, which is the start of the widget, which is the character string, if the widgetname is widget.launcher.home, it indicates that it must return to the launcher;

src – DOMString type, it is obtained through the src of the content element of the widget's config.xml file;

type – DOMString type, it is obtained by the type of the content element of the widget's config.xml file.

Return: None.

R.2.2.3.2 installWidget

Prototype: void installWidget(DOMString widgetname, DOMString id, DOMString url)

Description: Download and install the widget. If the widget does not exist, download it from the location specified by the url first.

Parameter: widgetname – DOMString type, which is widget name, which is a character string;

id – DOMString type, which is widget identifier, which is a character string;

url – DOMString type, which is the download location of the specified widget, which is a character string.

Return: None.

R.2.2.3.3 updateWidget

Prototype: void updateWidget(DOMString widgetname, DOMString id, DOMString url)

Description: update widget.

Parameter: widgetname – DOMString type, which is widget name, which is a character string;

id – DOMString type, which is widget identifier, which is a character string;

url – DOMString type, which is the download location of the specified widget, which is a character string.

Return: None.

R.2.2.3.4 checkWidget

Prototype: DOMString checkWidget(DOMString widgetname, DOMString id, DOMString url)

Description: detect widget.

Parameter: widgetname – DOMString type, which is widget name, which is a character string;

id – DOMString type, which is widget identifier, which is a character string;

url – DOMString type, which is the download location of the specified widget, which is a character string.

Return: version number.

R.2.2.3.5 uninstallWidget

Prototype: void uninstallWidget(DOMString widgetname, DOMString id)

Description: Uninstall the widget.

Parameter: widgetname – DOMString type, which is widget name, which is a character string;

id – DOMString type, which is widget identifier, which is a character string;

Return: None.

R.2.2.3.6 deletePackage

Prototype: void deletePackage(DOMString widgetname, DOMString id)

Description: Delete the application package.

Parameter: widgetname – DOMString type, which is widget name, which is a character string;

id – DOMString type, which is widget identifier, which is a character string;

Return: None.

R.2.2.4 Message callback

Message callback, used to feed back the current application process status and download status, the callback details are shown in Table R.3.

Table R.3 – Message callback of widget object

Callback name	Description
onstart	Feedback status when the application process starts
onresume	Feedback status when the application process is running
onpause	Feedback status when the application process is paused
onstop	Feedback status when the application process is stopped
onterminate	Feedback status when the application process is interrupted
ondownloadsuccess	Feedback status when the application is downloaded successfully
ondownloadfail	Feedback status when downloading the application fails

Table R.3 – Message callback of widget object

Callback name	Description
ondownloadsize	The feedback status when the application is being downloaded, including the total size and the current number of bytes, see: interface WidgetEvent: Event { readonly attribute long widgetEvenTtype; readonly attribute long download_totoalsize; readonly attribute long download_currSize;};

Annex S

JavaScript-System management unit

(This annex forms an integral part of this Recommendation.)

S.1 Overview

This annex defines the functional modules related to system management: data management, storage device management, file management, multimedia files, OTA software upgrades, system tools, and software and hardware information query.

S.2 Data management module

This module defines the JS object related to data management: DataConfig.

S.2.1 Message

The definition of the message that the data management module may send to the application layer is shown in Table S.1.

Table S.1 – Definition table of data management module message

Message name	event.which	event.modifiers	Message description
MSG_SAVE_DATA_SUCCESS	14001	–	Data writing is completed.
MSG_SAVE_DATA_FAILED	14002	–	Data writing is not completed.
MSG_REMOVE_DATA_SUCCESS	14003	–	Data deletion is completed.
MSG_REMOVE_DATA_FAILED	14004	–	Data deletion is not completed.
MSG_REVERT_DATA_SUCCESS	14005	–	Data recovery is completed.
MSG_REVERT_DATA_FAILED	14006	–	Data recovery is not completed.
MSG_RESTORE_TO_DEFAULT_SUCCESS	14007	–	Data restoration to factory settings is completed.
MSG_RESTORE_TO_DEFAULT_FAILED	14008	–	Data restoration to factory settings is not completed.
Reserved	14009～14100		
NOTE – The value of event.modifiers is automatically given by the system, and its data type: – "Number" indicating that the value is the ID of the message description string, which can be obtained through the Utility.getEventInfo() method. If the "message description" defines the message character string JSON format, the message content will be retrieved according to the format. – "–", indicating event.modifiers is undefined.			

S.2.2 DataConfig object

The DataConfig object is a built-in object.

System parameters and user parameters are stored in the receiving terminal NVM in the form of a data table, and the data table is accessed in the form of "key name + key value". The key value is stored in JSON format in RAM and NVM. When the application reads the data, it should be parsed according to the application scenario corresponding to the key name.

System parameters are provided by the system and stored in the system data table; user parameters are set by the application program and stored in the user data table. The system data table is used to store the global configuration information set by the system. Key names and key values are publicly disclosed in JSON format. Authorized applications can set/read parameters, but cannot delete data items; user data tables are maintained by the application itself, the JSON formats of the key names and key values are not publicly available.

After the receiving terminal is turned on, the data table is automatically read from the NVM to the RAM; after the user sets the parameters, the data in the RAM can be saved to the NVM.

NOTE – The system saves the setting values of the following functions. The saved values will be used next time the system is turned on, and will not be displayed in the system information table.

Volume setting, setting S/PDIF output interface data format, channel setting, video resolution, graphics layer transparency, display aspect ratio.

The JSON formats of the key names and key values of the system data table are shown in Table S.2.

Table S.2 – Definition table of data management module message

<table>
<tr><th>Key name</th><th colspan="3">Key value</th></tr>
<tr><td rowspan="5">DVBMainFrequency Info</td><td>Use</td><td colspan="2">It describes DVB main frequency point information.</td></tr>
<tr><td>JSON format</td><td colspan="2">[
{
"deliveryType":1,
"deliveryParams":[
{"frequency":626000, "symbolRate":6875, "modulation":3},
{"frequency":634000, "symbolRate":6875, "modulation":3},
{"frequency":642000, "symbolRate":6875, "modulation":3}
]
},
...
]
NOTE – The JSON format of the key value of the DVBMainFrequencyInfo key can describe multiple delivery systems, and each delivery system can describe multiple main frequency point information. In this example, the above-mentioned JSON character string describes a DVB-C delivery system, and the delivery system includes 3 main frequency point information.</td></tr>
<tr><td rowspan="3">Description</td><td>deliveryType</td><td>Int type, indicating the type of the delivery system, for the values, please see the constant field definition of the DeliverySystemType interface "Delivery System Type".</td></tr>
<tr><td>frequency</td><td>Int type, indicating the tuning frequency, and is measured in a unit related to the value of the deliveryType field:
– If deliveryType=1, it indicates DVB-C delivery system, in kilohertz (kHz);
– If deliveryType is other value, this key is meaningless.</td></tr>
<tr><td>symbolRate</td><td>Int type, indicating the symbol rate, in thousand symbols per second (ksymbol/s).</td></tr>
</table>

Table S.2 – Definition table of data management module message

Key name	Key value		
		modulation	Int type, indicating the modulation mode, and the value is related to the value of the deliveryType field: – If deliveryType=1, it indicates DVB-C delivery system. For the values, please see the constant field definition of "modulation mode" of the DvbcTunningParameters class; – If deliveryType is other value, this key is meaningless.
EPGSetting	Use	It describes EPG search setting information.	
	JSON format	{"search_start_date":0, "search_days":7, "program_event_maxcount":255}	
	Description	search_start_date	Int type, indicating the start date of the system searching the program table. The default is 0, indicating that the current day will be the start date of the search, 1 indicating the second day, 2 indicating the third day,..., and so on.
		search_days	Int type, indicating how many consecutive days the system will search for the program table.
		program_event_max_count	Int type, indicating the maximum number of program events searched by EPG. If the value is -1, it means unlimited.
AudioSetting	Use	It describes audio settings information.	
	JSON format	{ "enableGlobalVolume":0 }	
	Description	enableGlobalVolume	Int type, indicating whether to enable unified volume control, the value is: – indicating whether the application allows the user to set the volume of each channel separately; – the application allows to indicate that the volume of all live TV, audio broadcasting, NVOD, and mosaic is unified as the outputVolume value.
UserPreference	Use	It describes user preference setting information.	
	JSON format	{ "audioLang":"zho", "osdLang":"eng" }	
	Description	audioLang	Character string, indicating the audio language preferred by the user. The three-letter language code follows GB/T 4880.2-2000.

Table S.2 – Definition table of data management module message

Key name	Key value		
		osdLang	Character string, indicating the interface language preferred by the user. The three letter language code follows GB/T 4880.2-2000.
Portal	Use	It describes the portal server address information.	

	JSON format	{
		"address":"http://ngb.com",
		"port":8080,
		"returnChannel":"modulator"
		}	

	Description	address	Character string, indicating the portal server address.
		port	Int type, indicating the access port of the portal server.
		returnChannel	Character string, indicating a return channel, the values modulator and ethernet indicate the network modulator and Ethernet respectively.
NTP	Use	It describes the NTP server address information.	

	JSON format	{
		"address":"http://ntp.com",
		"port":8080
		}	

	Description	address	Character string, indicating the NTP server address.
		port	Int type, indicating the NTP server access port.
VODChannel	Use	It describes the VOD server address information.	
	JSON format	{ "MD5":"5A8B6493", "VODParams":[{ "frequency":634000, "symbolRate":6875, "modulation":3, "QAMName":1234 }, { "frequency":642000, "symbolRate":6875, "modulation":3, "QAMName":4321 }, {	

Table S.2 – Definition table of data management module message

<table>
<tr><th>Key name</th><th colspan="3">Key value</th></tr>
<tr><td rowspan="6"></td><td></td><td colspan="2">"frequency":650000,
"symbolRate":6875,
"modulation":3,
"QAMName":8888
}
]
}
NOTE – Only applicable to cable digital TV.</td></tr>
<tr><td rowspan="5">Description</td><td>MD5</td><td>Character string, indicating the MD5 code of the configuration file.</td></tr>
<tr><td>frequency</td><td>Int type, indicating the frequency of the IPQAM frequency, in kilohertz (kHz).</td></tr>
<tr><td>symbolRate</td><td>Int type, indicating the symbol rate of the IPQAM frequency point, in thousand symbols per second (ksymbol/s).</td></tr>
<tr><td>modulation</td><td>Int type, indicating the IPQAM frequency point modulation mode. For the value, please refer to the constant field definition of "modulation mode" of the DvbcTunningParameters class.</td></tr>
<tr><td>QAMName</td><td>Int type, indicating the VOD area code.</td></tr>
<tr><td rowspan="3">UserInfo</td><td>Use</td><td colspan="2">It describes end user information.</td></tr>
<tr><td>JSON format</td><td colspan="2">{
"adminPassword":"12345"
}</td></tr>
<tr><td>Description</td><td>adminPassword</td><td>Character string, indicating the administrator password, used to enter the system setting interface and watch locked channels.</td></tr>
<tr><td rowspan="2">Autodeployer</td><td>Use</td><td colspan="2">It describes the automatic deployment information of the application.</td></tr>
<tr><td>JSON format</td><td colspan="2">{
"mode":"auto-ip",
"ocPath":[
{
"deliveryType":1,
"deliveryParams":[
{"frequency":626000, "symbolRate":6875, "modulation":3},
{"frequency":634000, "symbolRate":6875, "modulation":3},
{"frequency":642000, "symbolRate":6875, "modulation":3}
]
},
...
],
"ipPath":[
{
"udpPath":"192.168.1.12",
"udpPort":8080,
"downloadTimeOut":60</td></tr>
</table>

Table S.2 – Definition table of data management module message

Key name	Key value		
		}, ...] } NOTE – The one-way broadcast channel supports multiple delivery systems and multiple frequency points to issue XML signaling files; the two-way broadband channel supports multiple UDP servers to issue XML signaling files.	
Autodeployer	Description	mode	Character string, indicating the method of obtaining the XML signaling file for automatic deployment, the value is: – The method of obtaining the file indicates obtaining the XML signaling file from the two-way broadband channel; – File; Broadband channel indicates obtaining XML signaling file from one-way broadcast channel; – File; Broadcast channel acquisition value: it indicates self-adaptation most of the time, and preferentially obtains XML signaling file from two-way broadband channel; – File; it indicates self-adaptation preferentially from two-way broadband communication, and preferentially obtains XML signaling file from one-way broadband channel.
		ocPath	JSON object type indicates OC download path for automatic deploying the xml signaling file. – Path. It indicates the automatic deployment of broadcast communication: int type, indicating the type of delivery system, same as the explanation of the DVBMainFrequencyInfo key; – Explanation; inFrequency: JSON object, same as the explanation of DVBMainFrequencyInfo key.
		ipPath	JSON object type, indicating the ip download path for automatic deploying the xml signaling file. – Path. It indicates automation: character string, indicating the UDP server address; – Device address; It indicates automation: Int type, indicating UDP service port; – Port; It indicates automatic deployment of uencyI: Int type, indicating application download timeout time, in second.
SeaChangeVOD	Use	It describes SeaChange VOD-related information.	

	JSON format	{
		"bakerFreq": {
		"frequency": 626000,
		"symbolRate": 6875,
		"modulation": 3	

Table S.2 – Definition table of data management module message

Key name	Key value		
		}, "serviceGroupId": 1 }	
		frequency	Int type, indicating the tuning frequency, – in kilohertz (kHz);
		symbolRate	Int type, indicating the symbol rate, in thousand symbols per second (ksymbol/s).
		modulation	Int type, indicating the modulation mode, for the value, please refer to the constant field definition of "modulation mode" of the DvbcTunningParameters class;
		serviceGroupId	Service Group Id value

S.2.2.1 Method

S.2.2.1.1 createUserPropertyTable

Prototype: number createUserPropertyTable(name)

Description: Creating a new user data table.

Parameter: name – string type, indicating the name of the user data table.

Return: number type, the value being:

– If the user data table is created successfully, the return value is greater than 0, which is used to indicate the globally unique data table identifier;

– If the creation of the user data table fails due to unknown reasons, the value 0 will be returned;

– If the user data table to be created already exists, the value −1 will be returned;

– If the name of the user data table to be created is invalid (for example, it is a null or empty character string), the value −2 will be returned.

S.2.2.1.2 getUserPropertyTable

Prototype: number getUserPropertyTable(name)

Description: Getting the user data table.

Parameter: name − string type, indicating the name of the user data table.

Return: number type, the value being as follows:

– If the user data table is successfully obtained, the return value is greater than 0, which is used to indicate the globally unique data table identifier;

– If the user data table fails to be obtained due to unknown reasons, the value 0 will be returned;

– If the user data table to be obtained does not exist, the value −1 is returned;

– If the name of the user data table to be obtained is invalid (for example, it is a null or empty character string), the value −2 is returned.

S.2.2.1.3 deleteUserPropertyTable

Prototype: number deleteUserPropertyTable(tableID)

Description: Delete the user data table.

Parameter: tableID − number type, user data table ID.

Return: number type, the value being as follows:

– If the user data table is deleted successfully, the return value is greater than 0, which is used to indicate the globally unique data table identifier;

– If the user data table fails to be deleted due to unknown reasons, the value 0 will be returned;

– If the user data table to be deleted does not exist, the value −1 is returned;

– If the name of the user data table to be deleted is invalid (for example, it is a null or empty character string), the value −2 will be returned.

S.2.2.1.4 getSystemPropertyTable

Prototype: number getSystemPropertyTable()

Description: Obtain the system data table.

Parameter: None.

Return: number type, the value being as follows:

– If the system data table is successfully obtained, the return value is greater than 0, which is used to indicate the globally unique data table identifier;

– If the system data table fails to be obtained due to unknown reasons, the value 0 will be returned;

– If the system data table to be obtained does not exist, the value −1 is returned.

S.2.2.1.5 createItem

Prototype: createItem(tableID,strItem,strValue)

Description: Creating a data item in the user data table and assign an initial value to the content of this data item. The operation on the system data table is invalid.

Parameter: tableID – number type, user data table ID.

strItem – string type, key name of the newly created data item.

strValue – string type, JSON format character string, indicating the key value of the newly created data item.

Return: number type, the value being as follows:

– If the data item is created successfully, the return value is greater than 0;

– If the creation of the data item fails due to unknown reasons, the value 0 will be returned;

– If the input data table ID does not exist, the value −1 will be returned;

– If the name of the input data item already exists in the data table, the value −2 will be returned;

– If the name of the input data item is invalid (for example, null or empty character string), the value -3 is returned.

S.2.2.1.6 deleteItem

Prototype: deleteItem(tableID,strItem)

Description: Delete a data item in the user data table. The operation on the system data table is invalid.

Parameter: tableID – number type, data table ID.

strItem – string type, name of the data item.

Return: number type, the value being as follows:

– If the deletion is successful, the return value is greater than 0;

– If the deletion of the data item fails due to unknown reasons, the value 0 will be returned;
– If the data item does not exist, the value −1 is returned.

S.2.2.1.7 getProperty

Prototype: string getProperty(tableID, strItem)

Description: Getting the value of a data item (read from memory). Both system data table and user data table can be operated.

Parameter: tableID – number type, data table ID.

strItem – string type, name of the data item.

Return: string type, value of the data item; if the data item does not exist, null is returned.

S.2.2.1.8 setProperty

Prototype: number setProperty(tableID,strItem,strValue)

Description: Setting the value of a data item (write to memory). Both system data table and user data table can be operated.

Parameter: tableID – number type, indicating the data table ID;

strItem – string type, indicating the name of the data item;

strValue – string type, indicating the value of the data item.

Return: number type, the value being as follows:
– If the modification is successful, the return value is greater than 0;
– If the modification of the content fails due to unknown reasons, the value 0 will be returned;
– If the data item does not exist, the value −1 is returned.

S.2.2.1.9 saveToNvm

Prototype: boolean saveToNvm(tableID)

Description: An Asynchronous method, writing the data table in RAM to NVM, overwrite the data table in NVM. Both system data table and user data table can be operated.
– If the data is written successfully, send the message MSG_SAVE_DATA_SUCCESS to the page;
– If the data writing fails, send the message MSG_SAVE_DATA_FAILED to the page.

Parameter: tableID – number type, data table ID.

Return: boolean type, true indicating the start of the execution, false indicating unexecution, and whether the execution result is successful is known by capturing the message.

S.2.2.1.10 removeFromNvm

Prototype: boolean removeFromNvm(tableID)

Description: An Asynchronous method, deleting the user table specified by the tableID parameter in the NVM.
– If the data is deleted successfully, send the message MSG_REMOVE_DATA_SUCCESS to the page;
– If the data deletion fails, send the message MSG_REMOVE_DATA_FAILED to the page.

Parameter: tableID – number type, indicating the data table ID.

Return: boolean type, true indicating the start of the execution, false indicating unexecution, and whether the execution result is successful is known by capturing the message.

S.2.2.1.11 revertFromNvm

Prototype: boolean revertFromNvm(tableID)

Description: An Asynchronous method, importing the data table in NVM into RAM, overwrite the current data table in memory. Both system data table and user data table can be operated.

– If the data coverage is successful, send the message MSG_REVERT_DATA_SUCCESS to the page;
– If the data coverage fails, send the message MSG_REVERT_DATA_FAILED to the page.

Parameter: tableID – number type, indicating the data table ID.

Return: boolean type, true indicating the start of the execution, false indicating unexecution, and whether the execution result is successful is known by capturing the message.

S.2.2.1.12 restoreDefault

Prototype: boolean restoreDefault()

Description: An Asynchronous method, restoring the system data table in NVM to the factory setting state, and update the system data table in RAM synchronously.

– If the data is restored successfully, send the message MSG_RESTORE_TO_DEFAULT_SUCCESS to the page;
– If the data recovery fails, send the message MSG_RESTORE_TO_DEFAULT_FAILED to the page.

Parameter: None.

Return: boolean type, true indicating the start of the execution, false indicating unexecution, and whether the execution result is successful is known by capturing the message.

S.3 External storage device management module

This module defines JS objects related to external storage device management: StorageDeviceManager, StorageDevice, StoragePartition.

S.3.1 Message

The definition of message that the external storage device management module may send to the application layer is shown in Table S.3.

Table S.3 – Definition table of message of external storage device management module

Message name	event.which	event.modifiers	Message format
MSG_DEVICE_UNINSTALL_SUCCESS	14101	–	The device is successfully uninstalled.
MSG_DEVICE_UNINSTALL_FAILED	14102	–	The uninstallation of the device failed.
Reserved	14103～14200		
NOTE – The value of event.modifiers is automatically given by the system, and its data type: – "Number", indicating that the value is the ID of the message description character string, which can be obtained through the Utility.getEventInfo() method. If the "message description" defines the message character string JSON format, the message content will be retrieved according to the format. – "–", indicating event.modifiers is undefined.			

S.3.2 StorageDeviceManager object

The StorageDeviceManager object is a built-in object, which completes the management and setting operations of external storage devices.

S.3.2.1 Method

S.3.2.1.1 uninstallDeviceByID

Prototype: public boolean uninstallDeviceByID(id)

Description: An Asynchronous method, uninstalling the device based on the device ID.

– If the uninstallation is successful, it will return "MSG_UNINATALL_DEVICE_SUCCESS" message;

– If the uninstallation fails, it will return "MSG_UNINATALL_DEVICE_FAILED" message.

Parameter: id – number type, device ID.

Return: boolean type, true value indicating the start of the execution, false indicating unexecution, and whether the execution result is successful is known by capturing the message.

S.3.2.1.2 getAllStorageDevices

Prototype: public StorageDevice[] getAllStorageDevices()

Description: Obtain all storage device information.

Parameter: None.

Return: Array of StorageDevice objects.

S.3.3 StorageDevice object

The StorageDevice object is a local object, which describes storage device information, including name, size, free space, etc.

Example:

```
//Create StorageDevice object through the StorageDeviceManager method.
  var storageDeviceArray = StorageDeviceManager.getAllStorageDevices();
  var storageDevice = storageDeviceArray[0];
```

S.3.3.1 Property

The definition of the property of the StorageDevice object is shown in Table S.4.

Table S.4 – Table of StorageDevice property

Property name	Type	Read and write property	Description
Id	number	Read only	Get the unique identifier of the storage device.
Name	string	Read only	Get the name of the storage device.

Table S.4 – Table of StorageDevice property

Property name	Type	Read and write property	Description
status	number	Read only	Get the status information of the storage device, the value is: – 0-indicating that the storage device has been uninstalled; – 1-indicating that the storage device has been found and the type has been determined; – 2-indicating that the storage device is ready; – 3-indicating that the storage device is not available.
serialNumber	string	Read only	Returns the serial number of the storage device.

S.3.3.2 Method

S.3.3.2.1 getAllPartitions

Prototype: public StoragePartition[] getAllPartitions()

Description: Obtain all partition information of the device.

Parameter: empty.

Return: Array of StoragePartition objects.

S.3.3.2.2 getPartitionByID

Prototype: public StoragePartition getPartitionByID(id)

Description: Getting the partition object of the specified ID.

Parameter: id – num ber type, specify the partition ID.

Return: StoragePartition object.

S.3.4 StoragePartition object

The StoragePartition object is a local object.

This object describes the partition information of the storage device, including name, size, idle state, access path, partition type, etc.

Example:

```
//Create StoragePartition object through the StorageDevice method.
var storagePartitionArray= storageDevice.getAllParitions();
var storagePartition = storagePartitionArray[0];
```

S.3.4.1 Property

The definition of the property of the StoragePartition object is shown in Table S.5.

Table S.5 – Table of property of StoragePartition

Property name	Type	Read and write property	Description
Id	number	Read only	Partition ID, that is, the unique identifier of the partition.
Name	string	Read only	Get the partition name of the storage device.
totalSize	number	Read only	Get the size of the partition space of the storage device, in KB.
freeSize	number	Read only	Get the free space size of the partition of the storage device, in KB.
Path	string	Read only	Get the access path of the partition of the storage device.
fsType	string	Read only	Get the system type of the partition file, that is partition format, the main values are E2FS, FAT32, NTFS, etc.
fsStatus	string	Read only	It indicates the partition status of the storage device, such as "good", "unformatted", etc.

S.4 File management module

This module defines JS objects related to file management: FileManager, FileObj and Directory. The relationship between the objects is shown in Figure S.1.

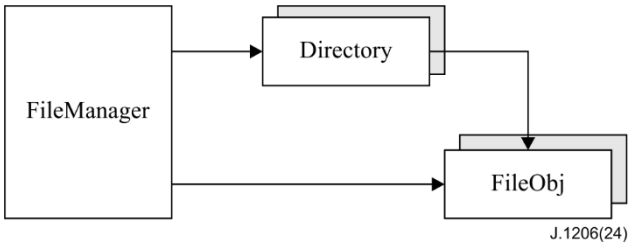


Figure S.1 – Relationship between file resource management objects

1) Local storage file system

The receiving terminal assigns an area in the local storage to the system (currently the local storage of the receiving terminal is usually implemented by flash memory, and may support other local storage media in the future) for the system to store the downloaded web main application, other applications, local configuration information, PSI/ SI information, user files and other data. For systems and applications, only this area is writable in the local storage, which is maintained and used by the system and applications. This area is uniformly identified with /storage/storage0 in the application to identify its root path. In the future, based on service needs, the second and third writable areas may be added, which are identified by /storage/storage1, /storage/storage2, and currently the terminal only needs to support /storage/storage0.

Example:

– Access the data stored in the boot application with the path /storage/storage0/startapp/;

– Access the data stored in the advertising service with the path /storage/storage0/adv/.

The corresponding relationship between the /storage/storage0 path of the receiving terminal and the Linux file system path (for example, /mnt/hd/HDD0) is implemented by the system itself.

2) USB file system

The path of the storage device connected to the USB interface of the receiving terminal is obtained by using the related objects of the external storage device management module interface in this annex.

S.4.1 Message

The message of the information of the file management module is shown in Table S.6.

Table S.6 – Message of file management module

Message name	event.which	event.modifiers	Message format
MSG_COPYFILE_SUCCESS	14201	–	The file is copied successfully.
MSG_COPYFILE_FAILED	14202	–	The copy of the file fails.
MSG_FILE_NOT_EXIST	14203	–	The source file does not exist.
MSG_SPACE_SHORTAGE	14204	–	The target storage space is insufficient.
MSG_MOVEFILE_SUCCESS	14205	–	The file moves successfully.
MSG_MOVEFILE_FAILED	14206	–	File moves failed.
MSG_CANNOT_DELETE_FILE	14207	–	Unable to delete source file.
MSG_DELETEFILE_SUCCESS	14208	–	The file is deleted successfully.
MSG_DELETEFILE_FAILED	14209	–	File deletion fails.
Reserved	14210～14300	–	
MSG_COPYDIRECTORY_SUCCESS	14301	–	The directory is copied successfully.
MSG_COPYDIRECTORY_FAILED	14302	–	The directory copy fails.
MSG_DIRECTORY_NOT_EXIST	14303	–	The source directory does not exist.
MSG_MOVEDIRECTORY_SUCCESS	14305	–	The directory moves successfully.
MSG_MOVEDIRECTORY_FAILED	14306	–	The directory moves failed.
MSG_CANNOT_DELETE_DIRECTORY	14307	–	Unable to delete the source directory.
MSG_DELETEDIRECTORY_SUCCESS	14308	–	The directory is deleted successfully.
MSG_DELETEDIRECTORY_FAILED	14309	–	Directory deletion fails.
MSG_DOWNLOAD_REMOTE_FILE_SUCCESS	14310	number	The front-end file has been downloaded to the memory.

Table S.6 – Message of file management module

Message name	event.which	event.modifiers	Message format
MSG_DOWNLOAD_REMOTE_FILE_NOT_EXIST	14311	number	The file to be downloaded does not exist on the front end.
MSG_DOWNLOAD_REMOTE_FILE_FAILED	14312	number	The front-end file download fails.
MSG_DOWNLOAD_REMOTE_FILE_TIMEOUT	14313	number	The front-end file download times out.
MSG_REMOTE_FILE_EXIST	14314	number	The specified file exists on the front end.
MSG_REMOTE_FILE_NOT_EXIST	14315	number	The specified file does not exist on the front end.
MSG_REMOTE_FILE_SEARCH_TIMEOUT	14316	number	The search for front-end file times out.
Reserved	14317～14400		

S.4.2 FileManager object

The FileManager object is a built-in object, used to manage local directories and files.

S.4.2.1 Method

S.4.2.1.1 copyFile

Prototype: boolean copyFile(path1, path2)

Description: An Asynchronous method, copying the file specified by path1 to path2 without deleting the source file.

– If the copy is successful, send the message MSG_COPYFILE_SUCCESS to the page;

– If the source file does not exist, send the message MSG_FILE_NOT_EXIST to the page;

– If the storage area where the target path is located is insufficient, send the message MSG_SPACE_SHORTAGE to the page;

– If the copy fails, send the message MSG_COPYFILE_FAILED to the page.

Parameter: path1-string type, the path of the source file (including the file name), there are two ways:

– Local storage file path similar to /storage/storage0/xx/xx.txt;

– USB external storage file path obtained through related objects of the device management interface.

path2-string type, the path of the target file (including the file name), there are two ways:

– Local storage file path similar to /storage/storage0/xx/xx.txt;

– USB external storage file path obtained through related objects of the device management interface.

Return: boolean type, true value indicating the start of the execution, false indicating the unexecution, and the execution result is known by capturing message.

S.4.2.1.2 moveFile

Prototype: boolean moveFile(path1, path2)

Description: An Asynchronous method, copying the file specified by path1 to path2, and deleting the source file.

– If the move is successful, send the message MSG_MOVEFILE_SUCCESS to the page;
– If the source file does not exist, send the message MSG_FILE_NOT_EXIST to the page;
– If the storage area where the target path is located is insufficient, send the message MSG_SPACE_SHORTAGE to the page;
– If the source file cannot be deleted, send the message MSG_CANNOT_DELETE_FILE to the page;
– If the move fails, send the message MSG_MOVEFILE_FAILED to the page.

Parameter: path1 – string type, the path of the source file (including the file name), there are two ways:

– Local storage file path similar to /storage/storage0/xx/xx.txt;
– The path of the USB external storage file obtained through the related object of the device management interface.

path2 – string type, the path of the target file (including the file name), there are two ways:

– Local storage file path similar to /storage/storage0/xx/xx.txt;
– The path of the USB external storage file obtained through the related object of the device management interface.

Return: boolean type, true value indicating the start of the execution, false indicating the unexecution, and the execution result is known by capturing message.

S.4.2.1.3 deleteFile

Prototype: boolean deleteFile(path)

Description: An Asynchronous method, deleting a local file.

– If the deletion is successful, send the message MSG_DELETEFILE_SUCCESS to the page;
– If the deletion fails, send the message MSG_DELETEFILE_FAILED to the page.

Parameter: path – string type, the path of the file to be deleted (including the file name), there are two ways:

– Local storage file path similar to /storage/storage0/xx/xx.txt;
– The path of the USB external storage file obtained through the related object of the device management interface.

Return: boolean type, true value indicating the start of the execution, false indicating the unexecution, and the execution result is known by capturing message.

S.4.2.1.4 existLocalFile

Prototype: boolean existLocalFile(path)

Description: Determine whether a local file exists.

Parameter: path – string type, the path of the local file (including the file name), there are two ways:

– Local storage file path similar to /storage/storage0/xx/xx.txt;
– USB external storage file path obtained through related objects of the device management interface.

Return: boolean type, true value indicating the file exists, false value indicating the file does not exist.

S.4.2.1.5 copyDirectory

Prototype: boolean copyDirectory(path1, path2)

Description: Copy the directory specified by path1 and all the contents of the directory to the path2 path, without deleting the source directory and its contents

– If the copy is successful, send the message MSG_COPYDIRECTORY_SUCCESS to the page;
– If the source file does not exist, send the message MSG_DIRECTORY_NOT_EXIST to the page;
– If the storage area where the target path is located is insufficient, send the message MSG_SPACE_SHORTAGE to the page;
– If the copy fails, send the message MSG_COPYDIRECTORY_FAILED to the page.

Parameter: path1 – string type, the path of the source directory, it can have the following two ways:

– The local storage directory path similar to /storage/storage0/xx;
– The USB external storage directory path obtained through the related objects of the device management interface.

path2 – string type, the path of the target directory, it can have the following two ways:

– The local storage directory path similar to /storage/storage0/xx;
– The USB external storage directory path obtained through the related objects of the device management interface.

Return: boolean type, true value indicating the start of the execution, false value indicating the unexecution, and the execution result is known by capturing message.

S.4.2.1.6 moveDirectory

Prototype: boolean moveDirectory(path1, path2)

Description: Copy the directory specified by path1 and all the contents of the directory to the path2 path, and delete the source directory and its contents

– When the move is successful, send the message MSG_MOVEDIRECTORY_SUCCESS to the page;
– If the source file does not exist, send the message MSG_DIRECTORY_NOT_EXIST to the page;
– If the storage area where the target path is located is insufficient, send the message MSG_SPACE_SHORTAGE to the page;
– If the source file cannot be deleted, send the message MSG_CANNOT_DELETE_DIRECTORY to the page;
– If the move fails, send the message MSG_MOVEDIRECTORY_FAILED to the page.

Parameter: path1 – string type, the path of the source directory, it can have the following two ways:

– The local storage directory path similar to /storage/storage0/xx;
– The USB external storage directory path obtained through the related objects of the device management interface.

path2 – string type, the path of the target directory, it can have the following two ways:

– The local storage directory path similar to /storage/storage0/xx;
– The USB external storage directory path obtained through the related objects of the device management interface.

Return: boolean type, true value indicating the start of the execution, false indicating the unexecution, and the execution result is known by capturing message.

S.4.2.1.7 deleteDirectory

Prototype: boolean deleteDirectory(path)

Description: Delete a directory and all its contents.

– If the deletion is successful, send the message MSG_DELETEDIRECTORY_SUCCESS to the page;
– If the deletion fails, send the message MSG_DELETEDIRECTORY_FAILED to the page.

Parameter: path – string type, indicating the path of the directory to be deleted, it can have the following two ways:

– Local storage directory path similar to /storage/storage0/xx;
– The USB external storage directory path obtained through the related objects of the device management interface.

Return: boolean type, true value indicating the start of the execution, false indicating the unexecution, and the execution result is known by capturing message.

S.4.2.1.8 existDirectory

Prototype: boolean existDirectory(path)

Description: Determine whether a local directory exists.

Parameter: path – string type, indicating the path of the local directory to be detected, it can have the following two ways:

– Local storage directory path similar to /storage/storage0/xx;
– The USB external storage directory path obtained through the related objects of the device management interface.

Return: boolean type, true value indicating that the file exists, false value indicating the file does not exist.

S.4.2.1.9 downloadRemoteFile

Prototype: downloadRemoteFile(path,endureTime)

Description: Download a file from the front end, and generate the corresponding FileObj object in the local memory.

Parameter: path – string type, indicating the path of the front-end file to be downloaded. There are two ways:

– OC address similar to dvb://onid.tsid.serviceid.componenttag/a/b.txt;
– http address similar to http://a.b.c/d/e.txt.

endureTime – number type, download timeout time, in seconds; if the value is 0, it indicates that there is no time limit for this asynchronous download process.

Return: number type, the system assigns a unique maskId for this asynchronous process.

Remarks: This method is an asynchronous operation: if the download is successful, the system sends the message MSG_DOWNLOAD_REMOTE_FILE_SUCCESS "The front-end file has been downloaded to the memory" to the application, at this time the application can obtain the FileObj object through the FileManager.getRemoteFile(maskId) method; If the file to be downloaded does not exist in the front end, the system sends the message MSG_DOWNLOAD_REMOTE_FILE_NOT_EXIST "the file to be downloaded does not exist in

the front end" to the application; If the file download fails, the system sends the message MSG_DOWNLOAD_REMOTE_FILE_FAILED "File download fails" to the application; If the time specified by endureTime has elapsed and the file download has not yet been completed, the system will send the message MSG_DOWNLOAD_FILE_TIMEOUT "Timeout expired, file download has not been completed" to the application, and the download operation will be terminated.

S.4.2.1.10 getRemoteFile

Prototype: getRemoteFile(maskId)

Description: Getting front-end files. After the application receives the message MSG_DOWNLOAD_REMOTE_FILE_SUCCESS "The front-end file has been downloaded to the memory", it can retrieve the FileObj object in the memory through this interface. The got FileObj object is in the closed state, and the file must be opened through the FileObj.open(mode) method in order to perform the file reading operation.

Parameter: maskId – number type, when calling the FileManager.downloadRemoteFile() method, the system automatically assigns a unique identifier for this asynchronous process.

Return: FileObj type, file object.

S.4.2.1.11 killObject

Prototype: killObject(obj)

Description: Clear File/Directory objects in memory, reclaim memory space.

Parameter: obj – FileObj or Directory type, indicating the memory file/directory object to be cleared.

Return: number type, 1 indicating clearing succeeded; 0 indicating clearing failed.

S.4.3 Directory object

The Directory object is a local object, which is used to manage the directory of the local disk.

Example:

```
//Create Directory objects through the construction method.
var dir = new Directory(path);
//The getting is performed through the method of the Directory object.
var directoryArray = dir.getDirList();
var directory = directoryArray[0];
```

S.4.3.1 Property

The definition of the property of the Directory object is shown in Table S.7.

Table S.7 – Table of property of Directory object

Property name	Type	Read and write property	Description
name	String	Read only	Get the name of the directory.
path	String	Read only	Get the absolute path of the directory. 1) Local storage directory path similar to /storage/storage0/xx; 2) The USB external storage directory path obtained through the related objects of the device management interface.

S.4.3.2 Method

S.4.3.2.1 Directory

Prototype: Directory(path)

Description: Construction method, which creates a new Directory object instance based on the specified path name.

Parameter: path – string type, indicating the character string of the directory path.

S.4.3.2.2 getFileList

Prototype: FileObj[] getFileList()

Description: Getting the files contained in the current directory.

Parameter: None.

Return: File object array, if the directory does not contain any files, the length of the returned array is 0.

S.4.3.2.3 getDirList

Prototype: Directory[] getDirList()

Description: Getting the subdirectories contained in the current directory.

Parameter: None.

Return: Directory object array, if the directory does not contain any subdirectories, the length of the returned array is 0.

S.4.4 FileObj object

FileObj is a local object, used to manage files. This object is equivalent to a mirror image of the local storage area file, USB external storage file or front-end file in the memory. Unless the saveAs(path) method is called to save the file, the actual file will not be changed.

S.4.4.1 Property

The definition of the property of the FileObj object is shown in Table S.8.

Table S.8 – Table of the property of the FileObj object

Property name	Type	Read and write property	Description
size	number	Read only	Get the size of the file, in bytes.
name	String	Read only	Get the name of the file (including extension).
path	String	Read only	Get the absolute path (including the file name) where the file is located in the following four ways: 1) Local storage file path similar to /storage/storage0/xx/xx.txt; 2) The USB external storage file path obtained through the related objects of the device management interface; 3) oc address path of dvb://onid.tsid.serviceid.component_tag/xx/xx.txt; 4) http address path similar to http://x.x.x.x/xx/xx.txt.

S.4.4.2 Method

S.4.4.2.1 FileObj

Prototype: FileObj(locator)

Description: Construction method, which creates a file object according to the specified file path name.

Parameter: locator – string type, indicating the locator of the file. See the path property description of the table of the property of the FileObj object for the format.

S.4.4.2.2 open

Prototype: open(mode)

Description: Open the file for various editing operations.

Parameter: mode – number type, 1 indicating open in text mode; 0 indicating open in binary mode.

Return: number type, 1 indicating opening is successful; 0 indicating opening failed.

S.4.4.2.3 close

Prototype: close()

Description: Close the file for the save operation.

Parameter: None.

Return: number type, 1 indicating closing is successful; 0 indicating closing failed.

S.4.4.2.4 readFile

Prototype: readFile(len)

Description: Read len bytes of data from the file and return it as a character string.

Parameter: len – number type, indicating reading the content of len byte length.

Return: string type, returning the read content as a character string.

S.4.4.2.5 readAllFile

Prototype: readAllFile()

Description: Read all the contents of the file and return it as a character string.

Parameter: None.

Return: string type, returning the read content as a character string.

S.5 Multimedia file module

This module defines JS objects related to multimedia files: AudioFile, VideoFile and ImageFile.

S.5.1 AudioFile object

The AudioFile object is a local object. This object describes the detailed information of an audio file.

Example:

```
//Create an AudioFile object.
var audioFile = new AudioFile(path);
```

S.5.1.1 Property

The definition of the property of the AudioFile object is shown in Table S.9.

Table S.9 – Table of the property of the AudioFile

Property name	Type	Read and write name	Description
Type	string	Read only	Get the audio file type, refer to MIME.
location	string	Read only	Get the locator (resource locator) of the audio file.
fileName	string	Read only	Get the audio file name.
fileSize	number	Read only	Get the size of the audio file, in bytes.
sampleRate	number	Read only	Sampling rate, such as 32000, 44100, 48000, in Hz.
numberOfChannels	number	Read only	The number of channels.
duration	number	Read only	The playback time of the audio file, in seconds.
Date	Date	Read only	The release date of the audio.
Title	string	Read only	The title of the audio.
Artist	string	Read only	Artist/performer.
album	string	Read only	The album name.

S.5.1.2 Method

S.5.1.2.1 AudioFile

Prototype: AudioFile(path)

Description: Construction method, create a new audio file object according to the specified path.

Parameter: path − string type, indicating the audio file path.

S.5.2 VideoFile object

The VideoFile object is a local object. This object describes the detailed information of a video file.

Example:

```
//Create a VideoFile object.
var videoFile = new VideoFile(path);
```

S.5.2.1 Property

The definition of the property of the VideoFile object is shown in Table S.10.

Table S.10 – Table of the property of the VideoFile

Property name	Type	Read and write property	Description
typeContainer	string	Read only	Encapsulation format, such as "AVI", "MP4", "MKV", etc.
typeVideo	string	Read only	Video type, refer to MIME.
typeAudio	string	Read only	Audio type of the accompanying sound, refer to MIME.
location	string	Read only	The locator (resource locator) of the video file.
fileName	string	Read only	The name of the video file.
fileSize	number	Read only	The size of the video file, in bytes.
bitRate	number	Read only	Play rate, in kilobits per second (kb/s)
duration	number	Read only	The playback time of the file, in seconds.

Table S.10 – Table of the property of the VideoFile

Property name	Type	Read and write property	Description
width	number	Read only	The width of the video, in pixels.
height	number	Read only	The height of the video, in pixels.
aspectRatio	string	Read only	Video aspect ratio, such as "16:9".
fps	number	Read only	Video frame rate.
sampleRate	number	Read only	Audio sampling rate of the accompanying sound, such as 32000, 44100, 48000, in Hz.
numberOfChannels	number	Read only	The number of channels.

S.5.2.2 Method

S.5.2.2.1 VideoFile

Prototype: VideoFile(path)

Description: Construction method, create a new video file object according to the specified path.

Parameter: path − string type, indicating the path of the video file.

S.5.3 ImageFile object

The ImageFile object is a local object. This object describes the detailed information of a picture file.

Example:

//Create an ImageFile object.

var imageFile = new ImageFile(path);

S.5.3.1 Property

The property of the ImageFile object is shown in Table S.11.

Table S.11 – Table of the property of the ImageFile

Property name	Type	Property	Description
type	string	Read only	For the type of image file, refer to MIME.
location	string	Read only	The locator (resource locator) of the image file.
fileName	string	Read only	The name of the image file.
width	number	Read only	The width of the picture file, in pixels.
height	number	Read only	The height of the picture file, in pixels.

S.5.3.2 Method

S.5.3.2.1 ImageFile

Prototype: ImageFile(path)

Description: Construction method, create a new image file object according to the specified path.

Parameter: path – string type, image file path.

S.6 OTA software upgrade module

This module defines the JS object related to OTA software upgrade: Upgrade.

S.6.1 Message

The definition of the message that the OTA software upgrade module may send to the application layer is shown in Table S.12.

Table S.12 – Message definition table of OTA software upgrade module

Message name	event.which	event.modifiers	Message description
MSG_FORCE_OTA	22101	–	It indicates the OTA upgrade is mandatory, the system automatically downloads the set-top box software image and upgrades the set-top boxNote.
MSG_OPTIONAL_OTA	22102	–	It indicates that the OTA upgrade is not mandatory, and the user can choose whether to upgrade this time.
Reserved	22103～22200		

S.6.2 Upgrade object

The Upgrade object is a built-in object that provides JS methods for OTA or local upgrade operations.

S.6.2.1 Method

S.6.2.1.1 checkOTA

Prototype: boolean checkOTA()

Description: Determine whether the front end deploys a new OTA upgrade. This method is mainly used to manually detect OTA upgrade information.

Parameter: None.

Return: boolean type, true indicating there is OTA upgrade, false indicating there is no OTA upgrade.

S.6.2.1.2 startOTA

Prototype: startOTA()

Description: Start the OTA upgrade.

Parameter: None.

Return: None.

S.6.2.1.3 getOTAName

Prototype: string getOTAName()

Description: Getting the name of the OTA upgrade event, which is different from the name of the OTA provider and refers to the text description of the OTA upgrade event.

Parameter: None.

Return: string type, indicating the name of the OTA upgrade event.

S.7 System tool module

This module defines tool JS objects: Utility, GlobalVarManager, Rectangle and SysTool.

S.7.1 Utility object

The Utility object is a built-in object, which can complete the acquisition of event messages and the printing of character strings.

S.7.1.1 Method

S.7.1.1.1 getEventInfo

Prototype: string getEventInfo(id)

Description: Getting a detailed description of the message.

Parameter: id – number type, indicating the message description character string ID.

Return: string type, indicating the detailed description information of the message.

S.7.1.1.2 println

Prototype: println(str)

Description: Print the character string, the parameter supports the addition operation of the character string, and the printing will automatically wrap. This method outputs the print information to the serial port, which is convenient for developers to check debugging information.

Parameter: str – string type, indicating the character string to be printed.

Return: None.

S.7.2 GlobalVarManager object

This object is a built-in object, used for global variable management.

S.7.2.1 Method

S.7.2.1.1 setItemValue

Prototype: setItemValue(key, value)

Description: Setting global variable parameters.

Parameter: key − string type, indicating the global variable keyword, which is set by the application;

value – number type, indicating the global variable value.

Return: None.

S.7.2.1.2 getItemValue

Prototype: number getItemValue(key)

Description: Getting global variable parameters.

Parameter: key − string type, indicating the global variable keyword, which should correspond to the key in the GlobalVarManager.setItemValue() method.

Return: number type, returning the value of the global variable specified by the keyword key; if the global variable specified by the keyword key does not exist, it returns null.

S.7.2.1.3 setItemStr

Prototype: setItemStr(key, str)

Description: Setting global variable parameters.

Parameter: key − string type, indicating the global variable keyword, which is set by the application;

str − string type, indicating the global variable value.

Return: None.

S.7.2.1.4 getItemStr

Prototype: string getItemStr(key)

Description: Getting global variable parameters.

Parameter: key − string type, indicating the global variable keyword, which should correspond to the key in the GlobalVarManager.setItemStr() method.

Return: string type, returning the global variable value specified by the keyword key; if the global variable specified by the keyword key does not exist, it returns null.

S.7.2.1.5 removeItem

Prototype: removeItem(key)

Description: Delete a global variable parameter.

Parameter: key – string type, global variable keyword; if the global variable specified by the keyword key does not exist, no operation is performed.

Return: None.

S.7.2.1.6 clearAll

Prototype: clearAll()

Description: Delete all global variable parameters.

Parameter: None.

Return: None.

S.7.3 Rectangle object

The Rectangle object is a local object.

S.7.3.1 Property

The property definition of the Rectangle object is shown in Table S.13.

Table S.13 – Table of the property of Rectangle object

Property name	Type	Read and write property	Description
left	number	Read/write	The X-axis coordinate of the upper left corner of the Rectangle object, in pixels.
top	number	Read/write	The Y-axis coordinate of the upper left corner of the Rectangle object, in pixels.
width	number	Read/write	The width of the Rectangle object, in pixels.
height	number	Read/write	The height of the Rectangle object, in pixels.

S.7.3.2 Method

S.7.3.2.1 Rectangle

Prototype: Rectangle(left, top, width, height)

Description: Construction method, which creates a new rectangular object according to the specified coordinates and width and height.

Parameter: left – number type, the X-axis coordinate of the upper left corner of the rectangle, in pixel;

right – number type, the Y-axis coordinate of the upper left corner of the rectangle, in pixel;

width – number type, the width of the rectangle, in pixel;

height – number type, the height of the rectangle, in pixel.

S.7.4 SysTool object

This object is a built-in object and provides methods for system standby, sleep, and restart operations. The system should verify the permissions of the application, and only privileged applications can call the methods provided by this class. Standby – The CPU of the receiving terminal is still working, used to run some background programs, such as push downloads, software upgrades, etc., for the turning off all audio and video output, from the appearance, the receiving terminal has stopped working. Sleep – The CPU of the receiving terminal stops working, and the power supply of the CPU and the motherboard is completely cut off. Then, depending on the external single-chip or other means to monitor the activation command issued by the remote control, start the switching power supply to supply power to the CPU and the main board to realize the remote start-up. This method consumes less energy. The relevant methods of this object should be implemented according to the actual capabilities of the receiver. For example, if sleep is not supported, the implementation of the sleep() method is empty or the same as the standBy() method.

S.7.4.1 Method

S.7.4.1.1 standBy

Prototype: standBy()

Description: Control the receiver to enter the standby state. The CPU is still running.

Parameter: None.

Return: None.

S.7.4.1.2 sleep

Prototype: sleep()

Description: Control the receiver to enter the sleep state, and the CPU will power off and stop working.

Parameter: None.

Return: None.

S.7.4.1.3 reboot

Prototype: reboot()

Description: Restart the receiver.

Parameter: None.

Return: None.

S.7.4.1.4 wakeUp

Prototype: wakeUp()

Description: Wake up the receiver.

Parameter: None.

Return: None.

S.7.4.1.5 getStandByState

Prototype: number getStandByState()

Description: Getting the standby state.

Parameter: None.

Return: number 1-true standby/sleep; 2-false standby/standBy; 3-normal work.

S.8 Software and hardware information query module

This module defines JS objects related to receiving terminal software and hardware information query: HardwareInfo, SoftwareInfo.

S.8.1 HardwareInfo object

HardwareInfo is a built-in object, which is used to describe the hardware parameter information of the receiving terminal.

S.8.1.1 Property

The property definition of the HardwareInfo object is shown in Table S.14.

Table S.14 – Table of the property of HardwareInfo object

Property name	Type	Read and write property	Description
Flash.size	string	Read only	Get the size of the receiving terminal's flash memory, in MB.
RAM.size	string	Read only	Get the size of the receiving terminal's memory, in MB.
RAM.type	string	Read only	Get the memory type of the receiving terminal, which can take the values "SDRAM", "DDR", "DDR2", etc.
SOC.model	string	Read only	Get the model of the main chip of the receiving terminal.
SOC.frequency	string	Read only	Get the operating frequency of the main chip of the receiving terminal, in MHz.
SOC.provider	string	Read only	Get the name of the provider of the main chip of the receiving terminal.
HW.version	string	Read only	Get the hardware version number of the receiving terminal.
STB.TPtype	string	Read only	Get the transmission mode type of the receiving terminal, the value can be a combination of "DVB-C", "DVB-S", "DVB-T", "ABS-SS", and "DTMB".
STB.definition	string	Read only	Get the definition type of the receiving terminal, which can take the values "HD" and "SD".
STB.provider	string	Read only	Get the name of the provider of the receiving terminal.
STB.brand	string	Read only	Get the brand name of the receiving terminal.
STB.model	string	Read only	Get the model of the receiving terminal.
STB.serialnumber	string	Read only	Get the serial number of the receiving terminal.
STB.returnPath	string	Read only	Get the property of the return channel of the receiving terminal. The value "modulator" indicates cableModem; the value "ethernet" indicates Ethernet.

S.8.2 SoftwareInfo object

SoftwareInfo is a built-in object, which is used to describe the parameter information of the receiving terminal software.

S.8.2.1 Property

The property definition of the SoftwareInfo object is shown in Table S.15.

Table S.15 – Table of the property of SoftwareInfo

Property name	Type	Read and write property	Description
OS.name	string	Read only	Get the name of the operating system software.
OS.version	string	Read only	Get the version number of the operating system software.
OS.provider	string	Read only	Get the name of the provider of the operating system software.
middleware.name	string	Read only	Get the name of the system software.
middleware.version	string	Read only	Get the version number of the system software.
middleware.provider	string	Read only	Get the name of the provider of the system software.
middleware.releaseDate	string	Read only	Get the release date of the system software.
middleware.copyright	string	Read only	Get the copyright information of the system software.
middleware.RAMSize	string	Read only	Get the memory space occupied by the system software, in KB.
middleware.NVMSize	string	Read only	Get the NVM space occupied by the system software, in KB.
middleware.platform_profile	string	Read only	Get the platform grade supported by the system software.
loader.name	string	Read only	Get the name of the software update module (loader).
loader.version	string	Read only	Get the version number of the software update module (loader).
loader.provider	string	Read only	Get the provider of the software update module (loader).
loader.size	string	Read only	Get the size of the software update module (loader), in KB.
CA.name	string	Read only	Get the name of the CA software.
CA.version	string	Read only	Get the version number of the CA software.
CA.provider	string	Read only	Get the provider of the CA software.
Driver.vision	string	Read only	Version number of the receiving terminal driver.

Annex T

JavaScript-Message management unit

(This annex forms an integral part of this Recommendation.)

T.1 Overview

This annex defines functional modules related to message management.

T.2 Message management module

This module defines JS object related to message processing: event.

The messages captured by the application layer have the following two sources:

– System messages, such as successful frequency lock, channel scan completed, network signal interruption, etc.;

– Key messages, such as remote control, keyboard, mouse, front panel and other key trigger messages;

The application layer captures messages in the following ways:

– System message: The application captures system messages through document.onsystemevent;

– Key message:

  • Keyboard: The application captures keyboard messages through document.onkeydown, document.onkeyup and document.onkeypress, and the message code is consistent with the PC mode;

  • Mouse: The application captures mouse messages through document.onmousedown, document.onmouseup, document.onmousemove, etc., and the message code is consistent with the PC mode;

  • Remote control: The application captures remote control messages through document.onkeydown and document.onkeyup, and the message code is compatible with the PC mode. The definition of the extended key message code is shown in Table T.1;

  • Front panel: The application captures front panel messages through document.onkeydown and document.onkeyup, and the message code is consistent with the remote control mode.

T.2.1 event object

The event object is a built-in object.

T.2.1.1 Property

The property of the event object is shown in Table T.1.

Table T.1 – Property of event object

Property name	Type	Read and write property	Description
event.type	number	Read only	It indicates the type of event that occurred, that is, the name of the event represented by the current event object, which has the same name as the registered event handle, such as

Table T.1 – Property of event object

Property name	Type	Read and write property	Description
			"onclick"; or the prefix "on" is deleted for the property of the event handle, such as "click". Same as W3C definition.
event.source	number	Read only	It indicates the source of the message.
event.which	number	Read only	It indicates the code value of the message.
event.modifiers	number	Read only	It indicates the extended property of the message. If the extended property of the message is empty, modifiers returns 0; if the extended property of the message is number type, then modifiers returns the value; if the extended property of the message is a character string, then modifiers returns an ID value, which is generated internally by the system, as a pointer to the specific character string content, the application can call the Utility.getEventInfo(ID) method to retrieve the character string content.

T.2.1.2 Message source

The definition of the message source (event.source) is shown in Table T.2.

Table T.2 – event.source definition

event.source	Description
1001	It indicates system messages.
1002	It indicates the remote control key message.
1003	It indicates the front panel key message.

T.2.1.3 System message

For system messages, see the definitions of messages in Annexes L, M, N, Q, S, U, V and other annexes.

T.2.1.4 Key message

Refer to clause D.2.1.5 for the key message definition.

Annex U

JavaScript-Application engine unit

(This annex forms an integral part of this Recommendation.)

U.1 Overview

This annex defines some commonly used application engine modules: channel management, electronic program guide, scheduled reminder management, information search, etc.

U.2 Channel management module

This module defines JS objects related to channel management: ChannelManager, Channel.

U.2.1 Message

The definitions of messages that the channel management module may send to the application layer are shown in Table U.1.

Table U.1 – Channel management module mMessage

Message name	event.which	event.modifiers	Message description
MSG_CHANNEL_RAM_TO_NVM_SUCCESS	19001	–	It indicates that writing channel data from RAM to NVM is successful.
MSG_CHANNEL_RAM_TO_NVM_FAILED	19002	–	It indicates that writing channel data from RAM to NVM failed.
MSG_CHANNEL_NVM_TO_RAM_SUCCESS	19003	–	It indicates that restoring channel data from NVM to RAM is successful.
MSG_CHANNEL_NVM_TO_RAM_FAILED	19004	–	It indicates that restoring channel data from NVM to RAM failed.
Reserved	19005～19500		
NOTE – The value of event.modifiers is automatically given by the system, and its data type: – "Number", indicating that the value is the ID of the message description character string, which can be obtained through the Utility.getEventInfo() method. If the "message description" defines the message character string JSON format, the message content will be retrieved according to the format. – "–", indicating event.modifiers is undefined.			

U.2.2 ChannelManager object

This object is a built-in object that describes how users manage channels.

U.2.2.1 Method

U.2.2.1.1 getChannelByChannelID

Prototype: Channel getChannelByChannelID(channelId)

Description: Getting the channel object based on the channel ID.

Parameter: channelId – number type, indicating the channel ID, it must be a decimal integer greater than 0.

Return: Channel object.

U.2.2.1.2 getChannelByLogicalID

Prototype: Channel getChannelByLogicalID(logicalId)

Description: Getting the channel object according to the logical channel number.

Parameter: logicalId – number type, indicating the logical channel number, it must be a decimal integer greater than 0.

Return: Channel object.

U.2.2.1.3 getChannelByServiceID

Prototype: Channel getChannelByServiceID(serviceId)

Description: Getting the channel object based on the service ID.

Parameter: serviceId – number type, indicating the service ID corresponding to the channel, and the value range being 0-65535.

Return: Channel object.

U.2.2.1.4 getLastChannel

Prototype: Channel getLastChannel()

Description: Getting the previously opened channel.

Parameter: None.

Return: Channel object.

U.2.2.1.5 getLastChannel

Prototype: Channel getLastChannel(serviceType)

Description: Getting the previously opened channel of the specified type.

Parameter: serviceType – number type, indicating the type of the service.

Return: Channel object.

U.2.2.1.6 getShutDownChannel

Prototype: Channel getShutDownChannel()

Description: Getting shutdown channel.

Parameter: None.

Return: Channel object.

NOTE – After the user turns on a certain channel, or switches channels, the system will immediately set the just now opened channel as the shutdown channel. In any case, the shutdown channel can be obtained. If the system has set the shutdown channel, it will get the set value; otherwise, the system decides the shutdown channel by itself.

U.2.2.1.7 getShutDownChannel

Prototype: Channel getShutDownChannel(serviceType)

Description: Getting the shutdown channel of the specified type.

Parameter: seviceType – number type, indicating the service type.

Return: Channel object.

NOTE – After the user turns on a certain channel, or switches channels, the system will immediately set the just now opened channel as the shutdown channel. In any case, the shutdown channel can be obtained. If the system has set the shutdown channel, the value obtained is the set value; otherwise, in the service type specified by serviceType, the system determines the shutdown channel by itself.

U.2.2.1.8 delChannel

Prototype: number delChannel(obj)

Description: Delete the specified channel from the channel list.

NOTE – Operate on the channel list in RAM.

Parameter: obj – Channel object.

Return: number type, 1 indicating the deletion is successful, 0 indicating the deletion failed.

U.2.2.1.9 deleteAll

Prototype: number deleteAll()

Description: Delete all channels in the channel list.

NOTE – Operate on the channel list in RAM.

Parameter: None.

Return: number type, 1 indicating success, 0 indicating failure.

U.2.2.1.10 deleteAllDelMarked

Prototype: number deleteAllDelMarked()

Description: Delete all channel objects marked for deletion in the channel list.

NOTE – Operate on the channel list in RAM.

Return: number type, 1 indicating success, 0 indicating failure.

U.2.2.1.11 deleteAllFavorites

Prototype: number deleteAllFavorites()

Description: Delete all favorite channels.

NOTE – Operate on the channel list in RAM.

Parameter: None.

Return: number type, 1 indicating success, 0 indicating failure.

U.2.2.1.12 resetProperties

Prototype: resetProperties()

Description: Reset all the channels that the user has set as favorite, locked, hidden, etc., and all channels are changed to non-favorite, non-locked, and non-hidden.

NOTE – Operate on the channel list in RAM.

Parameter: None.

Return: number type, 1 indicating success, 0 indicating failure.

U.2.2.1.13 swap

Prototype: swap(obj1,obj2)

Description: Exchange the positions of the channel object obj1 and the channel object obj2 in the channel list.

NOTE – Operate on the channel list in RAM.

Parameter: obj1 – indicating the Channel object;

obj2 – indicating the Channel object.

Return: None.

U.2.2.1.14 sort

Prototype: sort(sortTypeArray[], sortOrderArray[])

Description: Sort the channels according to the specified method.

NOTE – Operate on the channel list in RAM.

Parameter: sortTypeArray – number type array, indicating the sorting basis. There can be one or more sorting basis, but they cannot be repeated. The priority of the sorting basis is related to the order of the array members. The smaller the subscript of the array member, the higher the priority.

sortOrderArray – number type array, indicating the sorting mode.

Return: None.

NOTE – The hidden and locked channels must also be sorted. If two or more channels are sorted on the same basis, the previous sorting order between these channels will be maintained.

Example:

```
//Sort the channels in ascending order by service_id, and the channels with the same service_id are sorted in the order before this time,
sort([DVB_SORT_TYPE_SERVICE_ID], [DVB_SORT_ORDER_ASC]);
//Sort the channels according to whether they are encrypted first, non-encrypted channels are first, and encrypted channels are second; channels that are both non-encrypted or encrypted are then sorted in ascending order by service_name, and channels with the same service_name are sorted in the order before this sorting
sort([DVB_SORT_TYPE_FTA_SCR, DVB_SORT_TYPE_SERVICE_NAME],
[DVB_SORT_ORDER_ASC,DVB_SORT_ORDER_DESC]);
```

U.2.2.1.15 filter

Prototype: Channel[] filter(filterTypeArray[], valueArray[])

Description: Filter out the new channel list with specified conditions in the current channel list.

Parameter: filterTypeArray – number type array, indicating the filtering basis. The value of each member of the array is shown in Table U.2. There can be one or more sorting basis, the members cannot be repeated, and the filtering basis has no priority.

Table U.2 – Filter filting-conditions

filterTypeArray value	Description	Value corresponding to valueArray
const FILTER_TYPE_SERVICETYPE = 1000;	Filter out the channels of the specified service type.	See the definition of "Service Type Constant" for the value.
const FILTER_TYPE_FAV = 1001;	Filter out the channels with the specified favorite level.	Value: – 0-indicating filtering non-favorite channels; – 1-indicating filtering favorite channels.

Table U.2 – Filter filting-conditions

filterTypeArray value	Description	Value corresponding to valueArray
const FILTER_TYPE_BAT = 1002;	Filter out the channels under the specified service group.	It indicates bouquet_id.
const FILTER_TYPE_FTA = 1003;	Filter out encrypted or non-encrypted channels.	Value: – 0-indicating filtering out non-encrypted channels; – 1-indicating filtering out encrypted channels.
const FILTER_TYPE_HIDE = 1004;	Filter out hidden or non-hidden channels.	Value: – 0-indicating only filtering out non-hidden channels; – 1-indicating only filtering out hidden channels.

valueArray – number type array, which has the same length as filterTypeArray. The members of the array should correspond to the members of the filterTypeArray one-to-one, respectively indicating the value of each filterTypeArray member.

Return: Channel object array.

Example:

```
//It indicates filtering out the favorite channels among all TV broadcast channels
filter([FILTER_TYPE_SERVICETYPE, FILTER_TYPE_FAV],
    [DVB_SERVICE_TYPE_DTV, 1]);
//It indicates filtering out all encrypted channels in the service group with bouquet_id=10
filter([FILTER_TYPE_BAT, FILTER_TYPE_FTA],[10,1]);
```

U.2.2.1.16 save

Prototype: save()

Description: An Asynchronous method, saving the channel list data in RAM to NVM.

– If the save is successful, send the message MSG_CHANNEL_RAM_TO_NVM_SUCCESS;

– If the save fails, send the message MSG_CHANNEL_RAM_TO_NVM_FAILED.

Parameter: None.

Return: None.

U.2.2.1.17 restore

Prototype: restore()

Description: An Asynchronous method, restoring the channel list data in NVM to RAM.

– If the restoration is successful, send the message MSG_CHANNEL_NVM_TO_RAM_SUCCESS;

– If the restoration fails, send the message MSG_CHANNEL_NVM_TO_RAM_FAILED.

Parameter: None.

Return: None.

U.2.3 Channel object

The Channel object is a local object, used to describe the properties and methods related to the behavior of users and operators.

U.2.3.1 Property

The properties of the Channel object are defined in Table U.3.

Table U.3 – Table of Channel object properties

Property name	Type	Property	Description
channelId	number	Read/write	It indicates the channel number assigned by the receiving terminal. NOTE – The channel number assigned by the receiving terminal refers to the number assigned to the channel by the terminal system software, and the assignment strategy is determined by the terminal software. After the channelId is generated, it remains unchanged during the life cycle of the Channel object, and channel ordering and switching operations may modify the value of channelId.
logicalId	number	Read only	It indicates the logical channel number of the channel. If the channel does not have a logical channel number, this property returns the value -1. The acquisition of the logical channel number is related to the operator and is implemented by the receiving terminal itself.
isDeleted	number	Read/write	It indicates the delete flag of the channel. The value is: – 0-indicating no delete flag is set; – 1-indicating that the delete flag is set.
isFavorite	number	Read/write	It indicates the favorite level of the channel in the favorite class, value: – 0-indicating dislike; – 1-indicating like.
isLocked	number	Read/write	It indicates the locked state of the channel, value: – 0-indicating the channel is not locked; – 1-indicating the channel is locked. When a channel is locked, the user needs to enter the password before watching the program of the channel.

Table U.3 – Table of Channel object properties

Property name	Type	Property	Description
isHided	number	Read/write	It indicates the hidden state of the channel, value: – 0 indicating not hidden; – 1 indicating hidden. Once a channel is "hidden", it will not appear in the channel list, nor can it be watched by the up/down button on the remote control. It can only be watched after entering the "System Settings" to cancel its hidden state.
deltVolume	number	Read/write	It indicates the value-added (±) relative to the global volume, namely: Channel actual audio = global volume + deltVolume NOTE – The modification takes effect immediately.
supportPlayback	boolean	Read only	It indicates whether the channel supports time-shifted lookback service, value: – true-indicating support; – false-indicating not supported.

U.2.3.2 Method

U.2.3.2.1 getService

Prototype: DvbService getService()

Description: Getting DvbService object corresponding to the current channel object.

Parameter: None.

Return: DvbService object.

U.3 Electronic Program Guide Module

This module defines JS objects related to the electronic program guide: EPGManager, ProgramEvent, ReferenceEvent and TimeShiftEvent.

U.3.1 Message

The message definition that the electronic program guide module may send to the application layer is shown in Table U.4.

Table U.4 – Electronic Program Guide module message

Message name	event.which	event.modifiers	Message description
MSG_EPG_SEARCH_SUCCESS	18001	number	EPG search is completed successfully.
MSG_EPG_SEARCH_EXCEED_MAX_COUNT	18002	number	When the search result reaches the maximum value, the search automatically stops. The maximum number of ProgramEvent is specified by EPGSetting.program_event_max

Table U.4 – Electronic Program Guide module message

Message name	event.which	event.modifiers	Message description
			count in the data management module.
MSG_EPG_SEARCH_REFRESH	18003	number	EPG data update. When EPG has searched part of the data, the message is sent.
MSG_EPG_SEARCH_TIMEOUT	18004	number	Search EPG timed out. When no program information is searched within the time period specified by the interface, the message is sent.
MSG_EPG_RECEIVE_NVODREFERENCE_SUCCESS	18005	number	Successfully received NVOD reference event.
MSG_EPG_RECEIVE_ALL_NVODREFERENCE_SUCCESS	18006	number	The data of all NVOD frequency points has been received.
MSG_EPG_RECEIVE_NVODREFERENCE_TIMEOUT	18007	number	NVOD reference event search timed out.
MSG_EPG_RECEIVE_NVODTIMESHIFT_SUCCESS	18008	number	Successfully received a time-shift event under a reference event.
MSG_EPG_RECEIVE_NVODTIMESHIFT_TIMEOUT	18009	number	NVOD time-shift event receiving timed out.
Reserved	18010～18500		
NOTE – The value of event.modifiers is automatically given by the system, and its data type: – "Number", indicating the maskId of the search session. – "–", indicating event.modifiers is undefined.			

U.3.2 EPGManager object

The EPGManager object is a built-in object. This object provides a method to obtain an array of events under a specified class.

U.3.2.1 Constant

The definition of program type constants is shown in Table U.5.

Table U.5 – Program type constants

Constants	Description
Movie/drama	
const CONTENT_TYPE_MOVIE=0x10;	Movie/Drama (general)
const CONTENT_TYPE_MOVIE_DETECTIVE=0x11;	Detective/Thrill
const CONTENT_TYPE_MOVIE_ADVENTURE=0x12;	Expedition/Western/War
const CONTENT_TYPE_MOVIE_FANTASY=0x13;	Sci-Fi/Magic/Horror
const CONTENT_TYPE_MOVIE_COMEDY=0x14;	Comedy
const CONTENT_TYPE_MOVIE_SOAP=0x15;	Series/Situation/Folklore
const CONTENT_TYPE_MOVIE_ROMANCE=0x16;	Romantic

Table U.5 – Program type constants

Constants	Description
const CONTENT_TYPE_MOVIE_CLASSIC=0x17;	Serious/Classic/Religious/Historical Film/Drama
const CONTENT_TYPE_MOVIE_ADULT=0x18;	Adult film/drama
News/Real Time	
const CONTENT_TYPE_NEWS=0x20;	News/Current Affairs (general)
const CONTENT_TYPE_NEWS_WEATHER=0x21;	News/Weather
const CONTENT_TYPE_NEWS_MAGAZINE=0x22;	News Magazine
const CONTENT_TYPE_NEWS_DOCUMENTARY=0x23;	Documentary TV
const CONTENT_TYPE_NEWS_DISCUSSION=0x24;	Discussion/Interview/Debate
Performance/Entertainment	
const CONTENT_TYPE_SHOW=0x30;	Performance/Entertainment (general)
const CONTENT_TYPE_SHOW_GAME=0x31;	Recreation/Guess/Athletics
const CONTENT_TYPE_SHOW_VARIETY=0x32;	Various performances
const CONTENT_TYPE_SHOW_TALK=0x33;	Talk show
Sports	
const CONTENT_TYPE_SPORTS=0x40;	Sports (general)
const CONTENT_TYPE_SPORTS_SPECIAL=0x41;	Specific events (Olympics, World Cup, etc.)
const CONTENT_TYPE_SPORTS_MAGAZINE=0x42;	Sports magazine
const CONTENT_TYPE_SPORTS_SOCCER=0x43;	Football/rugby
const CONTENT_TYPE_SPORTS_TENNIS=0x44;	Tennis/Squash
const CONTENT_TYPE_SPORTS_TEAM=0x45;	Team events (including football)
const CONTENT_TYPE_SPORTS_ATHLETIC=0x46;	Track and field
const CONTENT_TYPE_SPORTS_MOTOR=0x47;	Racing car
const CONTENT_TYPE_SPORTS_WATER=0x48;	Water sports
const CONTENT_TYPE_SPORTS_WINTER=0x49;	Winter sports
const CONTENT_TYPE_SPORTS_EQUESTRAIN=0x4A;	Horse racing
const CONTENT_TYPE_SPORTS_MARTIAL=0x4B;	Martial arts
Children's/Youth Program	
const CONTENT_TYPE_CHILDREN=0x50;	Children/youth program (general)
const CONTENT_TYPE_CHILDREN_PRESCHOOL=0x51;	Programs for preschoolers
const CONTENT_TYPE_CHILDREN_6TO14=0x52;	Entertainment programs suitable for 6 to 14 years old
const CONTENT_TYPE_CHILDREN_10TO16=0x53;	Entertainment programs suitable for 10 to 16 years old
const CONTENT_TYPE_CHILDREN_SCHOOL=0x54;	Information/education/teaching program
const CONTENT_TYPE_CHILDREN_CARTOON=0x55;	Cartoon/Puppet
Music/ballet/dance	

Table U.5 – Program type constants

Constants	Description
const CONTENT_TYPE_MUSIC=0x60;	Music/ballet/dance (general)
const CONTENT_TYPE_MUSIC_POP=0x61;	Rock/pop
const CONTENT_TYPE_MUSIC_CLASSICAL=0x62;	Serious music/classical music
const CONTENT_TYPE_MUSIC_FOLK=0x63;	Folk/Traditional Music
const CONTENT_TYPE_MUSIC_JAZZ=0x64;	Jazz
const CONTENT_TYPE_MUSIC_OPERA=0x65;	Musical/opera
const CONTENT_TYPE_MUSIC_BALLET=0x66;	Ballet
Art/Culture (excluding music)	
const CONTENT_TYPE_ARTS=0x70;	Art/Culture (excluding music, general)
const CONTENT_TYPE_ARTS_PERFORMANCE=0x71;	Performance art
const CONTENT_TYPE_ARTS_ARTS=0x72;	Fine arts
const CONTENT_TYPE_ARTS_RELIGION=0x73;	Religion
const CONTENT_TYPE_ARTS_CULTURE=0x74;	Pop culture/traditional art
const CONTENT_TYPE_ARTS_LITERATURE=0x75;	Literature
const CONTENT_TYPE_ARTS_FILM=0x76;	Film/movie
const CONTENT_TYPE_ARTS_VIDEO=0x77;	Experimental movie/video
const CONTENT_TYPE_ARTS_PRESS=0x78;	Broadcast/News
const CONTENT_TYPE_ARTS_NEW=0x79;	New media
const CONTENT_TYPE_ARTS_MAGAZINE=0x7A;	Art/Cultural Magazine
const CONTENT_TYPE_ARTS_FASHION=0x7B;	Fashion
Social/Political Hotspot/Economy	
const CONTENT_TYPE_SOCIAL=0x80;	Social/Political Hotspot/Economy (General)
const CONTENT_TYPE_SOCIAL_MAGAZINE=0x81;	Magazine/Report/Documentary
const CONTENT_TYPE_SOCIAL_ECONOMICS=0x82;	Economic/Social Consulting
const CONTENT_TYPE_SOCIAL_PEOPLE=0x83;	Special character
Science/Education/Fact Topics	
const CONTENT_TYPE_EDUCATION=0x90;	Science/education/fact topics (general)
const CONTENT_TYPE_EDUCATION_NATUAL=0x91;	Nature/animal/environment
const CONTENT_TYPE_EDUCATION_TECNOLOGY=0x92;	Technology/Natural Science
const CONTENT_TYPE_EDUCATION_MEDICAL=0x93;	Medicine/Physiology/Psychology
const CONTENT_TYPE_EDUCATION_FOREIGN=0x94;	Abroad/expedition
const CONTENT_TYPE_EDUCATION_SOCIAL=0x95;	Social/Spiritual Science
const CONTENT_TYPE_EDUCATION_EDUCATION=0x96;	Re-education
const CONTENT_TYPE_EDUCATION_LANGUAGE=0x97;	Language
Leisure and entertainment	
const CONTENT_TYPE_LEISURE=0xA0;	Leisure and entertainment (general)

Table U.5 – Program type constants

Constants	Description
const CONTENT_TYPE_LEISURE_TRAVLE=0xA1;	Sightseeing/travel
const CONTENT_TYPE_LEISURE_HANDICRAFT=0xA2;	Handicraft
const CONTENT_TYPE_LEISURE_MOTOR=0xA3;	Self-driving
const CONTENT_TYPE_LEISURE_HEALTH=0xA4;	Fitness/health
const CONTENT_TYPE_LEISURE_COOKING=0xA5;	Cooking
const CONTENT_TYPE_LEISURE_SHOPPING=0xA6;	Advertising/shopping
const CONTENT_TYPE_LEISURE_GARDENING=0xA7;	Gardening
Describe special characteristics	
const CONTENT_TYPE_DESCRIPTION_LANGUAGE=0xB0;	Original language
const CONTENT_TYPE_DESCRIPTION_COLOR=0xB1;	Black/white
const CONTENT_TYPE_DESCRIPTION_UNPUBLISHED=0xB2;	Unreleased
const CONTENT_TYPE_DESCRIPTION_LIVE=0xB3;	Live
NOTE – The upper 4 bits of the constant value are the first-level program content classification, and the lower 4 bits are the second-level program content classification, such as 0xB1, where 0xB indicates the first-level program content classification, and 0x1 indicates the second-level program content classification.	

U.3.2.2 Method

U.3.2.2.1 searchProgramEvent

Prototype: number searchProgramEvent(tsList, mask, endureTime)

Description: An Asynchronous method, within the date range defined by the global parameters EPGSetting.search_start_date and EPGSetting.search_days, searching for the program event message specified by the parameter mask according to the frequency list specified by the parameter tsList.

– Send the message MSG_EPG_SEARCH_SUCCESS after the search is completed;

– If some data is received during the search, the message MSG_EPG_SEARCH_REFRESH can be sent to notify the page that some data has been searched, and the corresponding data can be obtained through the EPGManager.getProgramsByService() method;

– When the search time reaches the timeout value specified by the parameter endureTime, the system automatically stops the search and sends the message MSG_EPG_SEARCH_TIMEOUT.

The event.modifiers property of the message object should carry the search process identifier (maskId).

Parameter: tsList – DvbTS object array, specify the list of frequency points for searching program information.

mask – number type, indicating the search mask. The basic masks are:

– 0x01 – indicating searching for actual PF;

– 0x02 – indicating searching for actual schedule;

– 0x04 – indicating searching for other PF;

– 0x08 – indicating searching for other schedule.

The mask value can be composed of one or more basic masks, such as 0x03=(0x01|0x02), which indicates searching for the data of actual PF and actual schedule.

endureTime – number type, indicating the timeout period for searching for EPG message, in seconds.

Return: number type, indicating the unique identifier (maskId) assigned by the system for this asynchronous process.

U.3.2.2.2 searchProgramEventByService

Prototype: number searchProgramEventByService(serviceLocator, mask, endureTime)

Description: An Asynchronous method, within the date range defined by the global parameters EPGSetting.search_start_date and EPGSetting.search_days, searching for the program event message specified by the parameter mask according to the service specified by the parameter serviceLocator.

Parameter: serviceLocator – string type, indicating the broadcast service locator.

mask – number type, indicating the search mask. The basic masks are:

– 0x01 – indicating searching for actual PF;
– 0x02 – indicating searching for actual schedule;
– 0x04 – indicating searching for other PF;
– 0x08 – indicating searching for other schedule.

The mask value can be composed of one or more basic masks, such as 0x03=(0x01|0x02), which indicates searching for the data of actual PF and actual schedule.

endureTime – number type, indicating the timeout period for searching for EPG message, in seconds.

Return: number type, indicating the unique identifier (maskId) assigned by the system for this asynchronous process.

U.3.2.2.3 searchNVODRefEvents

Prototype: number searchNVODRefEvents(endureTime)

Description: An Asynchronous method, notifying the system to start receiving NVOD reference event data.

– If the NVOD data on a certain frequency point is successfully found, send the message MSG_EPG_RECEIVE_NVODREFERENCE_SUCCESS to the page, and all the received data can be obtained through the getReferenceEvents() method;
– If the data of all NVOD frequency points is received, send the message MSG_EPG_RECEIVE_ALL_NVODREFERENCE_SUCCESS to the page, and the search results are saved in the memory, and all the data can be obtained through getReferenceEvents();
– If the search time reaches endureTime, send the message MSG_EPG_RECEIVE_NVODREFERENCE_TIMEOUT to the page. At this time, the getReferenceEvents() method can also be called to obtain the information that has been searched.

Use the exitNVODMode() method to exit the receiving of the reference event.

Parameter: endureTime: number type, indicating the timeout period for searching for NVOD message, in seconds.

Return: number type, indicating the unique identifier (maskId) assigned by the system for this asynchronous process.

U.3.2.2.4 searchNVODRefEvents

Prototype: number searchNVODRefEvents(tsArray, endureTime)

Description: An Asynchronous method, notifying the system to start receiving NVOD reference event data at the specified TS frequency.

– If the NVOD data on a certain frequency point is successfully found, send the message MSG_EPG_RECEIVE_NVODREFERENCE_SUCCESS to the page, and all the received data can be obtained through the getReferenceEvents() method;
– If the data of all NVOD frequency points is received, send the message MSG_EPG_RECEIVE_ALL_NVODREFERENCE_SUCCESS to the page, and the search results are saved in the memory, and all the data can be obtained through getReferenceEvents();
– If the search time reaches endureTime, send the message MSG_EPG_RECEIVE_NVODREFERENCE_TIMEOUT to the page. At this time, the getReferenceEvents() method can also be called to obtain the information that has been searched.

Use the exitNVODMode() method to exit the receiving of the reference event.

Parameter: tsArray – DvbTS object array, search for the NVOD data of the specified TS frequency point.

endureTime – number type, indicating the timeout period for searching for NVOD message, in seconds.

Return: number type, indicating the unique identifier (maskId) assigned by the system for this asynchronous process.

U.3.2.2.5 getPresentProgram

Prototype: programEvent getPresentProgram(serviceLocator)

Description: Getting the current program of the specified service.

Parameter: serviceLocator – string type, indicating the broadcast service locator.

Return: ProgramEvent object.

U.3.2.2.6 getPresentProgramsByContentType

Prototype: programEvent[] getPresentProgramsByContentType(contentType)

Description: According to the program content classification value specified in the parameter, search for the current program message that meets the conditions in the current EPG database.

Parameter: contentType – number type, indicating the type of program content classification.

Return: ProgramEvent object array.

U.3.2.2.7 getPresentProgramsByName

Prototype: programEvent[] getPresentProgramsByName(str)

Description: According to the program name specified in the parameter, search for the current program message that meets the conditions in the current EPG database.

Parameter: str – string type, indicating search keyword.

Return: ProgramEvent object array.

U.3.2.2.8 getFollowingProgram

Prototype: programEvent getFollowingProgram(serviceLocator)

Description: Getting the following program of the specified service.

Parameter: serviceLocator – string type, indicating the broadcast service locator.

Return: ProgramEvent object.

U.3.2.2.9 getFollowingProgramsByContentType

Prototype: programEvent[] getFollowingProgramsByContentType(contentType)

Description: A Synchronous method, according to the program content classification value specified in the parameter, searching for the following program information that meets the conditions in the current EPG database.

Parameter: contentType – number type, indicating the type of program content classification.

Return: ProgramEvent object array.

U.3.2.2.10 getFollowingProgramsByName

Prototype: programEvent[] getFollowingProgramsByName(str)

Description: A Synchronous method, according to the program name specified in the parameter, searching the current program information that meets the conditions in the current EPG database.

Parameter: str – string type, indicating search keyword.

Return: ProgramEvent object array.

U.3.2.2.11 getProgramsByService

Prototype: programEvent[] getProgramsByService(serviceLocator)

Description: Getting all program information of a specified service.

Parameter: serviceLocator – string type, indicating the broadcast service locator.

Return: ProgramEvent object array.

U.3.2.2.12 getProgramsByDate

Prototype: programEvent[] getProgramsByDate(serviceLocator, beginDate, endDate)

Description: A Synchronous method, according to the start date and end date specified in the parameter, getting the program information that meets the conditions in the specified service.

Parameter: serviceLocator – string type, indicating the broadcast service locator.

beginDate – Date type object, indicating the start date.

endDate – Date type object, indicating the end date.

Return: ProgramEvent object array.

U.3.2.2.13 getProgramsByDirection

Prototype: programEvent[] getProgramsByDirection(serviceLocator, beginDate, count, isForward)

Description: A Synchronous method, according to the start date and search direction specified in the parameters, getting the specified number of program message in the specified service.

Parameter: serviceLocator – string type, indicating the broadcast service locator.

beginDate – Date type object, start date.

count – number type, indicating the number of program message to be obtained.

isForword – number type, 0 indicating backward search; 1 indicating forward search.

Return: ProgramEvent object array.

U.3.2.2.14 getProgramsByContentType

Prototype: programEvent[] getProgramsByContentType(contentType)

Description: A Synchronous method, according to the program content classification value specified in the parameter, searching for the program information that meets the conditions in the current EPG database.

Parameter: contentType – number type, program content classification type.

Return: ProgramEvent object array.

U.3.2.2.15 getProgramsByName

Prototype: programEvent[] getProgramsByName(str)

Description: A Synchronous method, according to the program name specified in the parameter, searching the program information that meets the conditions in the current EPG database.

Parameter: str – string type, indicating search keyword.

Return: ProgramEvent object array.

U.3.2.2.16 getReferencePrograms

Prototype: ReferenceEvent[] getReferencePrograms(serviceLocator)

Description: A Synchronous method, getting the reference program on the designated reference service.

Parameter: serviceLocator – string type, indicating the reference service locator.

Return: ReferenceEvent object array.

U.3.2.2.17 getReferenceEvents

Prototype: ReferenceEvent[] getReferenceEvents(sortType, sortOrder)

Description: Getting the searched ReferenceEvent object array.

Parameter: sortType – number type, indicating sorting basis, value:

– 1-indicating sorting according to the reference event ID;

– 2-indicating sorting according to the reference event name.

sortOrder-number type, indicating the sorting method, value:

– 0-indicating descending sort;

– 1-indicating ascending sort.

Return: ReferenceEvent object array.

U.3.2.2.18 exitNVODMode

Prototype: exitNVODMode()

Description: Exit the receiving of NVOD data.

Parameter: None.

Return: None.

U.3.3 ProgramEvent object

The ProgramEvent object is a local object, used to store program event information related to user behavior.

U.3.3.1 Property

The property definition of the ProgramEvent object is shown in Table U.6.

Table U.6 – Table of ProgramEvent property

Property name	Type	Property	Description
channelObj	Channel object	Read only	Channel object to which the program event belongs.
eventObj	DvbEvent object	Read only	DvbEvent object corresponding to the program event.
isBooked	number	Read/write	Whether the program is reserved to be marked, value: – 0-indicating not reserved; – 1-indicating reserved.

U.3.4 ReferenceEvent object

ReferenceEvent is a local object, used to store reference event message related to user behavior.

U.3.4.1 Property

The property definitions of the ReferenceEvent object are shown in Table U.7.

Table U.7 – Table of ReferenceEvent property

Property name	Type	Property	Description
channelObj	Channel object	Read only	Reference service Channel object to which the reference event belongs.
eventObj	DvbEvent object	Read only	DvbEvent object corresponding to the reference event.

U.3.4.2 Method

U.3.4.2.1 searchSchedules

Prototype: number searchSchedules(endureTime)

Description: An Asynchronous method, searching all time-shift event message corresponding to the reference event.

Parameter: endureTime – number type, allowable receiving timeout time, in seconds.

Return: number type,

– 1-indicating that there is no data in the cache, and the bottom layer will automatically lock the frequency point. If the data is received successfully, the message MSG_EPG_RECEIVE_NVODTIMESHIFT_SUCCESS will be sent; when the search time reaches the endureTime, the system will automatically stop the search and send the message MSG_EPG_RECEIVE_NVODTIMESHIFT_TIMEOUT, the getSchedules() function can also be called at this time to obtain the information that has been searched.

– 2-indicating that the NVOD time-shift event message that you want to collect already exists in the cache and is complete and can be obtained directly. At this time, no more messages will be returned to the page, and the time-shifted event message can be directly obtained when the return value is determined to be 2 on the page.

U.3.4.2.2 getSchedules

Prototype: TimeShiftEvent[] getSchedules()

Description: Getting a list of all time-shifted events within a few days from this moment on the reference event. Time-shifted events that have been played that day will be discarded, and the time-shifted events will be sorted according to the start time.

Parameter: None.

Return: TimeShiftEvent object array.

U.3.4.2.3 getPresentSchedules

Prototype: TimeShiftEvent[] getPresentSchedules()

Description: Getting all the time-shift event objects currently being played carried in the reference event, and the elements in the array are sorted according to the playback start time.

Parameter: None.

Return: TimeShiftEvent object array.

U.3.4.2.4 getFollowingSchedules

Prototype: TimeShiftEvent[] getFollowingSchedules()

Description: Getting all the time-shift event objects for the next playback carried in the reference event, and the elements in the array are sorted according to the playback start time.

Parameter: None.

Return: TimeShiftEvent object array.

U.3.5 TimeShiftEvent object

The TimeShiftEvent object is a local object, used to save time-shift event information.

U.3.5.1 Property

The properties of the TimeShiftEvent object are shown in Table U.8.

Table U.8 – Table of TimeShiftEvent object property

Property name	Type	Property	Description
channelObj	Channel object	Read only	Channel object to which the time-shift event belongs.
refChannelObj	Channel object	Read only	Reference service Channel object corresponding to the time-shift event.
eventObj	DvbEvent object	Read only	DvbEvent object corresponding to the time-shift event.
refEventObj	DvbEvent object	Read only	Reference DvbEvent object corresponding to the time-shift event.
status	number	Read only	Get the time-shift event status, value: – -1-indicating that the playback has been completed; – 0-indicating it is broadcasting; – 1-indicating it has not been broadcast yet. This property is a real-time value.

Table U.8 – Table of TimeShiftEvent object property

Property name	Type	Property	Description
orderIndex	number	Read only	It indicates the position of the time-shifted event in the reservation list, value: – -1-indicating not reserved; – ≥0-indicating the position of the time-shift event in the reservation list.
preEvent	TimeShiftEvent object	Read only	Get the previous time-shift event of the time-shift event. If null is returned, it indicates that the object is the first time-shift event.
nextEvent	TimeShiftEvent object	Read only	Get the next time-shift event of the time-shift event, if it returns null, it indicates that the object is the last time-shift event.

U.4 Reservation reminder module

U.4.1 Message

The message definition that the reservation reminder module may send to the application layer is shown in Table U.9.

Table U.9 – Reservation reminder module messages

Message name	event.which	event.modifiers	Message description
MSG_REMIND_RAM_TO_NVM_SUCCESS	20001	–	It indicates that writing channel data from RAM to NVM is successful.
MSG_REMIND_RAM_TO_NVM_FAILED	20002	–	It indicates that writing channel data from RAM to NVM failed.
MSG_REMIND_NVM_TO_RAM_SUCCESS	20003	–	It indicates that restoring channel data from NVM to RAM is successful.
MSG_REMIND_NVM_TO_RAM_FAILED	20004	–	It indicates restoring channel data from NVM to RAM failed.
Reserved	20005～20500		
NOTE – The value of event.modifiers is automatically given by the system, and its data type: – "Number", indicating that the value is the ID of the message description character string, which can be obtained through the Utility.getEventInfo() method. If the "message description" defines the message character string JSON format, the message content will be retrieved according to the format. – "–", indicating that event.modifiers is undefined.			

U.4.2 OrderManager object

The OrderManager object is a built-in object, used to describe the management mode of user reservations.

U.4.2.1 Property

The property definition of the OrderManager object is shown in Table U.10.

Table U.10 – OrderManager object properties

Property name	Type	Property	Description
advanceRemind	number	Read/write	This interface is used to set the pop-up reminder time of the reserved program, that is, how long the reserved program will be played, andthat the page reminds the user how long it will start to play the reserved program. The default value is 60, in seconds.
conflictInterval	number	Read/write	This interface is used to set the conflict threshold of reserved programs. If the user wants to reserve program A, within the range of the set minutes of the start time of program A, no other programs that have been reserved before can start to play, otherwise the system will not add a program reservation, and a reservation conflict program list will be generated in the memory (Which programs that have been reserved have conflict with the programs in this reservation on the start time), the default value for calling getConflictEvents() method is 5, in minute.
remindType	number	Read/write	0-indicating that the conflict Interval value is set as the conflict determination condition; 1-indicating that the entire duration of the program is used as the conflict determination condition.

U.4.2.2 Method

U.4.2.2.1 getPlayingList

Prototype: Order[] getPlayingList()

Description: Getting all the Order objects currently being played in the reservation list.

Parameter: None.

Return: Order object array.

U.4.2.2.2 getRemindList

Prototype: Order[] getRemindList()

Description: Getting all Remind objects to be played in the reservation list.

Parameter: None.

Return: Order object array.

U.4.2.2.3 getOrders

Prototype: Order[] getOrders(type)

Description: Getting all reserved programs according to the specified type.

Parameter: type – number type, indicating the type of program reservation, value:

– 0 or no parameter indicating obtaining all types of program reservations;
– 1 indicating obtaining the program reservation of the video type;
– 2 indicating obtaining the program reservation of the audio type;
– 3 indicating obtaining the program reservation of the NVOD type.

Return: Order object array.

U.4.2.2.4 getOrderByID

Prototype: Order getOrderByID(orderID)

Description: Obtain the reservation object according to the reservation identifier.

Parameter: orderID – number type, indicating the identifier of the reservation, and the value range being 0-65535.

Return: Order object.

U.4.2.2.5 getOrderByEvent

Prototype: getOrderByEvent(type, eventObj)

Description: Obtain the reservation object according to the event object corresponding to the reservation.

Parameter: type – indicating the eventObj parameter object type, value:

– 0-indicating eventObj is ProgramEvent object;

– 1-indicating that eventObj is TimeShiftEvent object.

eventObj-ProgramEvent object or TimeshiftEvent object, the type of the object is determined by the type parameter.

Return: Order object, if there is no corresponding Order object, null is returned.

U.4.2.2.6 getConflictOrders

Prototype: Order[] getConflictOrders()

Description: Getting the conflict reservation list, when the user wants to reserve a new program (the application calls the OrderManager.addEvent(obj) method), if the start time of the new program and the start time of the reserved program have conflict according to OrderManager.conflictInterval, the system will not add the program reservation, and a reservation conflict program list will be generated in the memory (which programs that have been reserved have conflict with the programs in this reservation on the start time), the reservation conflict program list can be accessed and retrieved by this method.

Parameter: None.

Return: Order object array.

U.4.2.2.7 getDelMarkedList

Prototype: Order[] getDelMarkedList()

Description: Getting the reservation program list that are currently marked for deletion.

Parameter: None.

Return: Order object array.

U.4.2.2.8 addEvent

Prototype: number addEvent(type, eventObj)

Description: Add reservation program information to the user reservation list.

NOTE – Only operate on the reservation list in RAM.

Parameter: type – indicating the type of the obj parameter object, the value is:

– 0-indicating that the eventObj parameter is of ProgramEvent type;

– 1-indicating that the eventObj parameter is of TimeShiftEvent type.

eventObj – ProgramEvent or TimeShiftEvent object to be reserved, the type is specified by the type parameter.

Return: number type, the value being:

– 1-indicating the addition is successful;
– 0-indicating the program has been broadcast;
– -1-indicating adding conflict;
– -2-indicating there is no corresponding service;
– -3-indicating that the reserved space is full.

U.4.2.2.9 deleteOrder

Prototype: deleteOrder(orderObj)

Description: Delete the specified reserved program from the reservation program list.

NOTE – Only operate on the reservation list in RAM.

Parameter: orderObj – Order object.

Return: number type, 1 indicating successful deletion of the reserved program, 0 indicating failure to delete the reserved program.

U.4.2.2.10 deleteAll

Prototype: deleteAll()

Description: Delete all reserved programs in the program reservation list.

NOTE – Only operate on the reservation list in RAM.

Parameter: None.

Return: None.

U.4.2.2.11 deleteAllDelMarked

Prototype: deleteAllDelMarked()

NOTE – Only operate on the reservation list in RAM.

Description: Delete all reserved programs of the specified type from the user reservation program list, and clear related information from the NVM.

Parameter: None.

Return: None.

U.4.2.2.12 delConflictOrders

Prototype: number delConflictOrders()

NOTE – Only operate on the reservation list in RAM.

Description: Delete all conflicting reserved programs.

Parameter: None.

Return: number type, value 0 indicating deletion failed, 1 indicating deletion succeeded.

U.4.2.2.13 restore

Prototype: restore()

Description: Import the program reservation list saved by the current NVM into RAM.

Parameter: None.

Return: number type, value 1 indicating success in restoring the reservation data in the NVM to the memory, and 0 indicating failure.

U.4.2.2.14 save

Prototype: save()

Description: Saving the program reservation list in RAM to the NVM of the receiving terminal.

Parameter: None.

Return: None.

U.4.3 Order object

The Order object is a local object, used to describe the program reservation information.

U.4.3.1 Property

The properties of the Order object are shown in Table U.11.

Table U.11 – Table of Order object property

Property name	Type	Property	Description
orderID	number	Read only	Get the identifier of the current reservation. When a user reserves a program on the terminal interface, the system generates an Order object and assigns an orderID. It must be ensured that the orderID is unique among the Order objects of all users that currently exist.
channelObj	Channel	Read only	Get the channel object to which the current reservation belongs.
deleteFlag	number	Read/write	Mark the reserved program with the deletion mark, the value is: – 0-indicating there is no deletion mark; – 1-indicating that there is a deletion mark.
Type	number	Read only	It indicates the eventObj object type, the value being: – 0-indicating that the eventObj property is ProgramEvent object; – 1-indicating that the eventObj property is TimeShiftEvent object.
eventObj	ProgramEvent/ TimeShiftEvent	Read only	Get the event object corresponding to the current reservation, the type is specified by the type property.

U.5 Message search module

The message search module defines JS objects related to global search and matching search: SearchManager, GlobalSearchSession, AutoCompleteSearchSession, GloalSearchFilter, AutoCompleteSearchFilter, GlobalSearchResultItem, AutoCompleteSearchItem and SearchHistoryItem.

Definition of terms:

– Global Search – Find SI, PVR and other content according to the search conditions set by the user, and return meaningful search results to improve the user experience;

– Matching search – According to the user's input, the character string that can be matched in the current data source is given to shorten the time for the user to enter the querying keyword and reduce the difficulty for the user to enter the querying keyword.

The message search module is divided into the following components:

– Search Manager (SearchManager) – the entry class of the message search module.

– Global Search Session (GlobalSearchSession)-session associated with the global search process.

– Matching search session (AutoCompleteSearchSession) – the session associated with the ongoing matching search process.

U.5.1 Constants

The constant definitions of this module are shown in Table U.12.

Table U.12 – Constant definitions of message search module

Constants	Value	Description
Data source type		
SourceType.ALL	0	It indicates that the data source of the search is two parts: broadcast data and local recording data.
SourceType.BROADCAST	1	It indicates that the data source of the search is broadcast data.
SourceType.RECORDED	2	It indicates that the searched data source is local recording data.
Search field		
SearchFields.ALL_STRING_FIELDS	0	It indicates searching all character string fields.
SearchFields.SYNOPSIS	4	It indicates searching profile field.
SearchFields.TITLE	5	It indicates searching title field.
Data content type		
SearchContentType.ALL	0	It indicates searching for data in both audio and video formats.
SearchContentType.AUDIO_ONLY	1	It indicates searching for data only in the audio format.
SearchContentType.VIDEO_ONLY	2	It indicates searching for data only in the video format.
Get paging direction		
RetrieveDirection.FIRST_PAGE	0	It indicates that the data on the first page is got according to the getting results by paging.
RetrieveDirection.NEXT_PAGE	1	It indicates that the data on the next page is got according to the getting results by paging.
RetrieveDirection.PREVIOUS_PAGE	2	It indicates that the data on the previous page is got according to the getting results by paging.
Filter condition		
SearchCriteriaFlags.FLAG_NONE	0	Filter condition flag-empty.
SearchCriteriaFlags.FLAG_SD_EVENT	1	Filter condition tag-filter standard definition content.
SearchCriteriaFlags.FLAG_HD_EVENT	2	Filter condition tag-filter high definition content.
SearchCriteriaFlags.FLAG_3D_EVENT	4	Filter condition tag-filter 3D content.
SearchCriteriaFlags.FLAG_CLEAR	32	Filter condition tag-filter unscrambled content.
SearchCriteriaFlags.FLAG_SCRAMBLED	64	Filter condition tag-filter scrambled content.

Table U.12 – Constant definitions of message search module

Constants	Value	Description
Search status		
SearchStatus.INITIATED	0	Search status-initialization is complete.
SearchStatus.IN_PROGRESS	1	Search status-in progress.
SearchStatus.COMPLETED	2	Search status-end.
SearchStatus.INTERRUPTED	3	Search status-interrupted.
SearchStatus.TIMEOUT	4	Search status-search stops after timeout.
SearchStatus.TIMEOUT_STOP_FAILED	5	Search status-search stop failed after timeout.
SearchStatus.FAILED	6	Search status-search failed.
SearchStatus.STOP_SUCCESS	7	Search status—successfully stop the search.
SearchStatus.STOP_FAILED	8	Search status-failed to stop the search.
SearchStatus.DISPOSE_SUCCESS	9	Search status-successfully close the search.
SearchStatus.DISPOSE_FAILED	10	Search status-close the search failed.
SearchStatus.RETRIEVAL_SUCCESS	11	Search status—successfully get the search results.
SearchStatus.RETRIEVAL_FAILED	12	Search status-failed to get the search results.
SearchStatus.RETRIEVAL_INSUFFICIENT	13	Search status-not enough search results are available.

U.5.2 SearchManager object

SearchManager is a built-in object, the manager of global search and matching search, and the entry class of the information search module. At the same time, the historical search records can be got through it.

U.5.2.1 Method

U.5.2.1.1 getAutoCompleteSearchSession

Prototype: AutoCompleteSearchSession getAutoCompleteSearchSession(autoCompleteFiter)

Description: Getting the matching search session object. Each matching search object is an independent search session. The system can only have one matching search session at a time.

Parameter: autoCompleteFilter – An AutoCompleteSearchFilter object, indicating a matching search filter.

Return: An AutoCompleteSearchSession object, indicating an instance of the matching search session.

U.5.2.1.2 getGlobalSearchSession

Prototype: GlobalSearchSession getGlobalSearchSession(globalFilter,sortCriteria)

Description: Getting the global search session object. Each global search session object is an independent search session. In order to have multiple global search sessions at the same time, the application needs not only to obtain different objects, but also to handle the corresponding callback function. According to the terminal capabilities, the system decides itself whether to support multiple global search sessions at the same time.

Parameter: globalFilter – A GlobalSearchFilter object, indicating the global search filter.

sortCriteria – A SortCriteria object, indicating the criteria for sorting search results.

Return: A GlobalSearchSession object, indicating an instance of the global search session.

U.5.2.1.3 getSearchHistory

Prototype: SearchHistoryItem[] getSearchHistory()

Description: Getting a list of historical search records.

Parameter: None.

Return: SearchHistoryItem array, indicating historical search records. If there is a successful search record before, and the application explicitly calls the method for saving search result, this method returns the history record of the successful search; if the history record is empty, the returned array length is 0.

U.5.2.1.4 clearSearchHistory

Prototype: void clearSearchHistory()

Description: Clear historical search records. At any time, the application can call this method to clear historical search records to protect personal privacy.

Parameter: None.

Return: None.

U.5.3 GlobalSearchSession object

GlobalSearchSession describes a global search session, provides interfaces for the global search session and related control.

U.5.3.1 Constants

The constant definitions of the GlobalSearchSession object are shown in Table U.13.

Table U.13 – Table of GlobalSearchSession object constant definition

Constant	Value	Description
DEFAULT_PAGE_SIZE	6	The default page size of the global search results, that is, the number of search result items contained in each page.

U.5.3.2 Property

The property definition of the GlobalSearchSession object is shown in Table U.14.

Table U.14 – GlobalSearchSession object properties

Property name	Type	Read and write property	Description
pageSize	number	Read and write	When returning search results by page, specify the number of results contained in each page. The default value is GlobalSearchSession.DEFAULT_PAGE_SIZE.
onGlobalSearchStart	Function pointer	Read and write	Global search session start event handling method.
onGlobalSearchStop	Function pointer	Read and write	Global search session closing event handling method.

Table U.14 – GlobalSearchSession object properties

Property name	Type	Read and write property	Description
onGlobalSearchDestroy	Function pointer	Read and write	Global search session destruction event handling method.
onGlobalSearchRetrieval	Function pointer	Read and write	Global search session obtains the search result event handling method.
onGlobalSearchError	Function pointer	Read and write	Global search session error event handling method.

Examples of using the callback method properties of the GlobalSearchSession object are as follows:

```
var globalSearchSession = SearchManager.getGlobalSearchSession(globalFilter, sortCriteria);
globalSearchSession.onGlobalSearchStart = eventHandle;
globalSearchSession.onGlobalSearchStop = eventHandle;
globalSearchSession.onGlobalSearchDestroy = eventHandle;
globalSearchSession.onGlobalSearchRetrieval = eventHandle;
globalSearchSession.onGlobalSearchError = eventHandle;
globalSearchSession.pageSize = 10;
......
globalSearchSession.startSearch("The Founding of A Party");
......
//definition of callback method
function eventHandle(aEvent) {
  ......
switch (aEvent){
  case COMPLETED：
    break;
  }
  ......
}
......
```

U.5.3.3 Method

U.5.3.3.1 startSearch

Prototype: void startSearch(searchStr)

Description: Start a new global search request. Once the search is completed, the system will notify the application through an event to get the search results.

Parameter: searchStr – string type, indicating the character string to be searched entered by the user.

Return: None.

U.5.3.3.2 stopSearch

Prototype: void stopSearch()

Description: Terminate the global search request that has been started before.

Parameter: None.

Return: None.

U.5.3.3.3 getSearchResultList

Prototype: GlobalSearchResultItem[] getSearchResultList()

Description: To get the global search results, the application needs to wait for the event trigger before calling this method. The results are stored in the same cache. Once a new search is initialized, this interface can no longer be used to obtain the results of the last search.

Parameter: None.

Return: A GlobalSearchResultItem array, indicating global search results. If there is no search result, the length of the returned array is 0.

U.5.3.3.4 getResultCount

Prototype: number getResultCount()

Description: Getting the number of search result items. If the search has not been completed and is partially updated, the number of results will also be updated according to the change of the search element. For each threshold limit, the application will receive a notification of SearchStatus.IN_PROGRESS.

Parameter: None.

Return: number type, indicating the number of search result items.

U.5.3.3.5 dispose

Prototype: void dispose()

Description: Destroy global search session objects that are no longer needed.

NOTE – After the global search is completed, once the session is no longer needed, this method must be called to release the corresponding resources.

Parameter: None.

Return: None.

U.5.3.3.6 retrievePage

Prototype: void retrievePage(retrieveDirection)

Description: Getting the data on the first page, the next page or the previous page from the search result list, an asynchronous method, waiting for event processing mechanism.

Parameter: retrieveDirection – number type, used to specify the direction of getting the result.

Return: None.

U.5.3.3.7 saveRecentSearchQuery

Prototype: void saveRecentSearchQuery()

Description: Saving the latest search results to the history records.

Parameter: None.

Return: None.

U.5.3.4 Callback method

The event description of the GlobalSearchSession object is shown in Table U.15, and the value of the event code is shown in the "Search Status" constant definition in Table U.12.

Table U.15 – GlobalSearchSession object related events

Trigger event	Callback disposal method
COMPLETED IN_PROGRESS INITIATED	onGlobalSearchStart
STOP_SUCCESS	onGlobalSearchStop
DISPOSE_SUCCESS	onGlobalSearchDestroy
RETRIEVAL_SUCCESS	onGlobalSearchRetrieval
FAILED INTERRUPTED TIMEOUT_STOP_FAILED TIMEOUT DISPOSE_FAILED RETRIEVAL_FAILED RETRIEVAL_INSUFFICIENT STOP_FAILED EXPAND_RESULT_FAILED	onGlobalSearchError

U.5.3.4.1 onGlobalSearchStart

Description: The properties of the GlobalSearchSession object, the callback method of the global search session start event. It is used to monitor the global search start event, the application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating the global search event.

Return: None.

U.5.3.4.2 onGlobalSearchStop

Description: The properties of the GlobalSearchSession object, the callback method of the global search session stop event. It is used to monitor the global search stop event, the application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating the global search event.

Return: None.

U.5.3.4.3 onGlobalSearchDestroy

Description: The properties of the GlobalSearchSession object, the callback method of the global search session destruction event. It is used to monitor the global search destruction event, the application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating the global search event.

Return: None.

U.5.3.4.4 onGlobalSearchRetrieval

Description: The properties of the GlobalSearchSession object, the global search session gets the search result event callback method. It is used to monitor the global search to get some search result events, the application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating the global search event.

Return: None.

U.5.3.4.5 onGlobalSearchError

Description: The properties of the GlobalSearchSession object, the callback method of the global search session error event. It is used to monitor the global search error event, the application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating the global search event.

Return: None.

U.5.4 AutoCompleteSearchSession object

The matching search session object provides a matching search session control method.

U.5.4.1 Property

The property definition of the AutoCompleteSearchSession object is shown in Table U.16.

Table U.16 – AutoCompleteSearchSession object properties

Property name	Type	Read and write property	Description
onAutoCompleteSearchStart	Function pointer	Read and write	Matching search session start event handling method.
onAutoCompleteSearchStop	Function pointer	Read and write	Matching search session closing event handling method.
onAutoCompleteSearchDestroy	Function pointer	Read and write	Matching search session destruction event handling method.
onAutoCompleteSearchRetrieval	Function pointer	Read and write	Matching search session gets the search result event handling method.
onAutoCompleteSearchError	Function pointer	Read and write	Matching search session error event handling method.

Examples of using the callback method properties of the AutoCompleteSearchSession object are as follows:

```
var autoCompleteSearchSession = SearchManager.getAutoCompleteSearchSession(autoCompleteFilter);
autoCompleteSearchSession.onAutoCompleteSearchStart = eventHandle;
autoCompleteSearchSession.onAutoCompleteSearchStop = eventHandle;
autoCompleteSearchSession.onAutoCompleteSearchDestroy = eventHandle;
```

```
autoCompleteSearchSession.onAutoCompleteSearchRetrieval = eventHandle;
autoCompleteSearchSession.onAutoCompleteSearchError = eventHandle;
......
autoCompleteSearchSession.startSearch("The Founding of A Party");
......
//Definition of callback method
function eventHandle(aEvent) {
  ......
switch (aEvent){
  case COMPLETED：
    break;
  }
  ......
}
......
```

U.5.4.2 Method

U.5.4.2.1 startSearch

Prototype: void startSearch(searchStr)

Description: Start a new matching search request. Once the search is completed, the search engine will notify the application to get the search results. After starting the search, the application needs to wait for the notification of onAutoCompleteSearchStart() event and then get the results.

NOTE 1 – If the timeout of AutoCompleteSearchFilter is set to 0, the search will stop only when the stopSearch() method is explicitly called.

NOTE 2:

1 Get matching search results by calling the getSearchResultList() method;

2 Terminate the previous search request that is still in progress by calling the stopSearch() method;

3 If the matching search session is no longer used, release it by calling the dispose() method;

4 At the same time, only one matching search session can be active. During this period, the startSearch() and stopSearch() methods can be called repeatedly, but when the matching search session is no longer needed, dispose () method should be called to close the search session and release resources; the search results are stored in the same cache. Once a new search is started, the old search results will no longer be valid.

Parameter: searchStr – string type, indicating the keyword character string entered by the user to be searched.

Return: None.

U.5.4.2.2 stopSearch

Prototype: void stopSearch()

Description: Terminate previously initiated matching search requests.

NOTE – This operation only terminates the matching search, but the matching search session is still valid, and the corresponding resources are not released. This method should be forced to be called in the following scenarios:

– If the matching search has been started and the results are waiting to be returned, the user must call the stopSearch() method to terminate the search when the user enters another character before the search engine returns the results;
– The timeout limit of the matching search filter is set to 0, so that it will always wait for the search result. If there is no corresponding matching result, the stopSearch() method must be called to terminate the search before starting another search.

Tip: If the matching search has been completed, there is no need to call this method.

Parameter: None.

Return: None.

U.5.4.2.3 dispose

Prototype: void dispose()

Description: Destroy matching search session objects that are no longer needed.

NOTE – After the matching search is completed, once the session is no longer needed, this method must be called to release the corresponding resources.

Parameter: None.

Return: None.

U.5.4.2.4 getSearchResultList

Prototype: string[] getSearchResultList()

Description: Getting the results of a matching search.

Parameter: None.

Return: string array, indicating all the searched character string arrays that match the querying keywords entered by the user.

U.5.4.3 Callback method

The event description of the AutoCompleteSearchSession object is shown in Table U.17.

Table U.17 – AutoCompleteSearchSession object related events

Search event	Callback disposal method
COMPLETED	onAutoCompleteSearchStart
STOP_SUCCESS	onAutoCompleteSearchStop
DISPOSE_SUCCESS	onAutoCompleteSearchDestroy
FAILED TIMEOUT STOP_FAILED DISPOSE_FAILED	onAutoCompleteSearchError

U.5.4.3.1 onAutoCompleteSearchStart

Description: The properties of the AutoCompleteSearchSession object, the matching search session start event callback method. It is used to monitor the matching search start event. The application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating matching search event.

Return: None.

U.5.4.3.2 onAutoCompleteSearchStop

Description: Properties of the AutoCompleteSearchSession object, the matching search session stop event callback method. It is used to monitor the matching search stop event. The application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating matching search event.

Return: None.

U.5.4.3.3 onAutoCompleteSearchDestroy

Description: properties of the AutoCompleteSearchSession object, the matching search session destruction event callback method. It is used to monitor the matching search destruction event. The application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating matching search event.

Return: None.

U.5.4.3.4 onAutoCompleteSearchError

Description: properties of the AutoCompleteSearchSession object, the matching search session error event callback method. It is used to monitor the matching search error event. The application should assign it an event handling function handle to handle the corresponding event.

Prototype: function eventHandle(aEvent)

Parameter: aEvent – number type, indicating matching search event.

Return: None.

U.5.5 GlobalSearchFilter object

The global search filter object provides methods for setting and obtaining global search filter conditions. The global search filter conditions can be:

– Search data source;

– Search field;

– The language of the text information;

– The maximum number of returned results;

– Search timeout time limit.

NOTE – The application using GlobalSearchFilter can obtain the following effects: According to the specified search data source and search field, the search result list that matches the keyword input by the user is listed within the timeout limit, and the number of search results is less than or equal to the maximum value set by the application.

U.5.5.1 Constants

The constant definitions of the GlobalSearchFilter object are shown in Table U.18.

Table U.18 – Table of GlobalSearchFilter object constant definition

Constant	Value
DEFAULT_MAX_GLOBAL_SEARCH_RESULTS	50

U.5.5.2 Property

The properties of the GlobalSearchFilter objects are defined in the following table.

Table U.19 – Definitions of GlobalSearchFilter object properties

Property name	Type	Read and write property	Description
source	number	Read and write	Specify the search source, the default is SourceType.ALL.
contentType	number	Read and write	It indicates the content type, the default is SearchContentType.ALL.
contentNibble	number	Read and write	It indicates the class of the program to be filtered.
searchField	number	Read and write	Search fields, the default is SearchFields.ALL_STRING_FIELDS.
searchLanauage	string	Read and write	Preset system language on the receiving terminal, the default is "zho".
criteriaFlags	number	Read and write	It indicates filter conditions.
maxResults	number	Read and write	It indicates the maximum number of returned results, the default value is GlobalSearchFilter.DEFAULT_MAX_GLOBAL_SEARCH_RESULTS.
threshold	number	Read and write	Set the threshold of search query results, the default is 0.
timeLimit	number	Read and write	Timeout setting, the default is 0, in milliseconds.

U.5.5.3 Method

U.5.5.3.1 GlobalSearchFilter

Prototype: GlobalSearchFilter()

Description: Construction method, creates a GlobalSearchFilter object.

Parameter: None.

U.5.6 AutoCompleteSearchFilter object

Matching search filter object, provides methods for setting and obtaining matching search filter conditions. The matching search filter conditions can be:

– Search data source;

– Search field;

– Language of the text information;

– Maximum number of returned results;

– Search timeout time limit.

NOTE – The application uses AutoCompleteSearchFilter to obtain the following effects: According to the specified search data source and search field, a list of prompt character strings that match the characters input by the user are listed within the timeout limit, and the number of character strings is less than or equal to the maximum value set by the application.

U.5.6.1 Constants

The constant definitions of the AutoCompleteSearchFilter object are shown in the following table.

Table U.20 – AutoCompleteSearchFilter object constant

Constant	Value
DEFAULT_MAX_AUTO_COMPLETE_SEARCH_RESULTS	10

U.5.6.2 Property

The property definitions of the AutoCompleteSearchFilter object are shown in the following table.

Table U.21 – AutoCompleteSearchFilter Object Properties

Property name	Type	Property	Description
source	number	Read and write	Specify the search source, the default is SourceType.ALL.
searchField	number	Read and write	Search fields, the default value is SearchFields.ALL_STRING_FIELDS.
searchLanauage	string	Read and write	It indicates the language of the search text, following the three-letter code agreed by the GB/T 4880.2-2000 standard, and the default value is "zho".
maxResults	number	Read and write	It indicates the maximum number of returned results, the default value is AutoCompleteSearchFilter. DEFAULT_MAX_AUTO_COMPLETE_SEARCH_ RESULTS.
timeLimit	number	Read and write	Timeout setting, the default value is 0, in milliseconds.

U.5.6.3 Method

U.5.6.3.1 AutoCompleteSearchFilter

Prototype: AutoCompleteSearchFilter()

Description: Construction method, creates an AutoCompleteSearchFilter object.

Parameter: None.

U.5.7 SortCriteria object

The SortCriteria object describes the sorting mechanism of the search results, including two rules, one is to sort according to the specified field, and the other is to use it in ascending or descending order. The two rules need to be used in combination, that is, sort a field in ascending or descending order.

U.5.7.1 Constants

The constant definitions of the SortCriteria object are shown in Table U.22.

Table U.22 – Constants of SortCriteria object

Constants	Value
Sorting method (ascending/descending)	
SORT_ORDER_NONE	0
SORT_ORDER_ASCENDING	1
SORT_ORDER_DESCENDING	2
Sorting basis	

Table U.22 – Constants of SortCriteria object

Constants	Value
SORT_TYPE_NONE	0
SORT_TYPE_TITLE	1
SORT_TYPE_START_TIME	2
SORT_TYPE_CONTENT_NIBBLE	15

U.5.7.2 Property

The property definitions of the SortCriteria object are shown in Table U.23.

Table U.23 – Table of SortCriteria object property definitions

Property name	Type	Read and write property	Description
sortOrder	number	Read and write	The possible values are the "sorting method" constants defined in Table W.22.
sortType	number	Read and write	The possible values are the "sorting basis" constants defined in Table W.22.

U.5.7.3 Method

U.5.7.3.1 SortCriteria

Prototype: SortCriteria()

Description: Construction method, creates a default SortCriteria object for sorting search results.

Parameter: None.

U.5.7.3.2 SortCriteria

Prototype: SortCriteria(field, order)

Description: A construction method, creating a SortCriteria object according to the specified parameters, which is used to sort the search results.

Parameter: field – number type, which specifies which field to be sorted, which can be the "sorting basis" constants defined in Table W.22 for the value.

order – number type, which specifies the ascending or descending type of sorting, which can be the "sorting method" constant defined in Table W.22 for the value.

U.5.8 GlobalSearchResultItem object

U.5.8.1 Constant

The constant definitions of the GlobalSearchResultItem object are shown in Table U.24.

Table U.24 – GlobalSearchResultItem object constants

Constants	Value	Description
CONTENT_DVBEVENT	0	It indicates that the content type is DvbEvent type.
CONTENT_PVREVENT	1	It indicates that the content type is PVREvent type.

U.5.8.2 Property

The property definition of the GlobalSearchResultItem object is shown in Table U.25.

Table U.25 – GlobalSearchResultItem object properties

Property name	Type	Read and write property	Description
contentType	number	Read only	The value can be CONTENT_DVBEVENT or CONTENT_PVREVENT.

U.5.8.3 Method

U.5.8.3.1 getContent

Prototype: object getContent()

Description: Getting the object associated with this search result. The object may be of type DvbEvent or PVREvent, indicated by the contentType property.

Parameter: None.

Return: Object object type, the specific type is determined according to the value of the contentType property:

– If contentType=GlobalSearchResultItem.CONTENT_DVBEVENT, then return DvbEvent object;

– If contentType=GlobalSearchResultItem.CONTENT_PVREVENT, then return the PVREvent object;

U.5.9 SearchHistoryItem object

The SearchHistoryItem object describes a search history record and provides methods to obtain various information about the search history record.

U.5.9.1 Method

U.5.9.1.1 getContentType

Prototype: number getContentType()

Description: Getting the content type of this search history record.

Parameter: None.

Return: number type, indicating the content type of the search history record.

U.5.9.1.2 getCriteriaFlags

Prototype: number getCriteriaFlags()

Description: Getting the filter character of the search history record.

Parameter: None.

Return: number type, indicating the filter condition of the search history record.

U.5.9.1.3 getSearchField

Prototype: number getSearchField()

Description: Getting the search field of this search history record.

Parameter: None.

Return: number type, indicating the search field of the search history record.

U.5.9.1.4 getSearchString

Prototype: string getSearchString()

Description: Getting the search keyword character string of the search history record.

Parameter: None.

Return: string type, indicating the search keyword character string entered by the user.

U.5.9.1.5 getSortCriteria

Prototype: SortCriteria getSortCriteria()

Description: Getting the sorting information of the search record.

Parameter: None.

Return: SortCriteria object, indicating the sorting method and sorting basis of the search results.

U.5.9.1.6 getSource

Prototype: number getSource()

Description: Getting the search source of this search history record.

Parameter: None.

Return: number type, indicating the search source.

Annex V

JavaScript-Broadcast Information Service Management Unit

(This annex forms an integral part of this Recommendation.)

V.1 Overview

This annex defines the relevant JavaScript interface of the broadcast information service management unit.

V.2 Broadcast information service management module

This module defines JS objects related to broadcast information service management: DthManager, Ad, AdService.

V.2.1 Message

The message definition of the broadcast information service management module is shown in Table V.1.

Table V.1 – Broadcast Information service management module message definition

Message name	event.which	event.modifiers	Message description
DTH_EVENT_EMBD_TRIG	15001	number	Emergency broadcast event trigger event, the message character string JSON format is: {"service_id":param1[Note 1], "ts_id":param2[Note 2], "orig_net_id":param3[Note 3]}
DTH_EVENT_EMBD_CANCEL	15002	–	Emergency broadcast event cancel event.
DTH_EVENT_OSD_UPDATE	15003	number	Prompt application, there is a new OSD prompt message
DTH_EVENT_SERVICE_UPDATE	15004	number, a parameter of 0 indicates that the program does not need to be updated immediately, a parameter of 1 indicates that the program information is forced to be updated	NIT service update
DTH_EVENT_RESET_DATA	15005	When the parameter is 01, the set-top box data is required to be erased immediately (dth processing,	Erase messages reported by the data

Table V.1 – Broadcast Information service management module message definition

Message name	event.which	event.modifiers	Message description
		just reporting a message to the application), when the parameter is 0, it indicates that the state of erasing the set-top box is restored to the normal state.	
DTH_EVENT_GPRS_SEND_STATUS	15006	A message of 01 indicates a successful sending, and a message of 00 indicates a failed sending	Determining whether the data is sent successfully
DTH_EVENT_BOUQUET_ID_UPDATE	15007	number	Bouquet id update message
DTH_EVENT_UPGRADE_TRIG	15008	A message of 01 indicates a mandatory upgrade, and a message of 00 indicates a non-mandatory upgrade	Software upgrade message
DTH_EVENT_FINGERPRINT_TRIG	15009	number	Fingerprint trigger event.[Note 4]
DTH_EVENT_GPRS_STATUS	15010	number	Current available status.[Note 5]
DTH_EVENT_GPRS_BASE_STATION	15011	number	GPRS current base station information.[Note 6]
Reserved	15012–15100	–	

The value of event.modifiers is automatically given by the system, and its data type:

– "number", indicating that the value is the ID of the message description character string, which can be obtained through the Utility.getEventInfo() method. If the "message description" defines the message character string JSON format, the message content will be retrieved according to the format.

– "–", indicating event.modifiers is undefined.

NOTE 1 – param1: number type, program service ID;

NOTE 2 – param2: number type, transport stream ID.

NOTE 3 – param3: number type, network ID.

NOTE 4 – Fingerprint trigger event, the message character string JSON format is: {

char_color: number //Text color

reg_hight: number //Background area height

reg_widgth: number //Background area width

reg_color: number //Background area color

X_Reg_Offset: number // Background area x coordinate

Table V.1 – Broadcast Information service management module message definition

Message name	event.which	event.modifiers	Message description

Y_Reg_Offset: number // Background area y coordinate
X_Text_Offset: number // Text x coordinate
Y_Text_Offset: number // Text y coordinate
Duration: number //duration
fs_text: String //Text content
};
NOTE 5 – Currently available status: {
0, //Module is normal and currently available
-1, //Module is being initialized
-2, //Module SIM is abnormal
-3, //Module network is abnormal
-4, //Module is in a busy status, and need to be obtained later.
-5, //No response from the module
-6, //Other errors
}			

NOTE 6 – Base station information report event, the JSON format contained in the array is: [{ LAC:number /*location area code location area code*/ Cell_ID:number /* Cell (base station) number*/ Bsic_ID:number /* Base station identification code (C network)*/ ucMNC:number /* Network number*/ strength:number /*Signal strength*/ }, …]			

V.2.2 DthManager object

DthManager object is a built-in object that provides management methods for DTH components.

V.2.2.1 Method

V.2.2.1.1 startServer

Prototype: number startServer()

Description: The DTH component is called in the live broadcast star application. The server side of the DTH component is started. You need to wait for the live broadcast star application to start before calling this interface to start each function of the DTH component.

Parameter: None.

Return: number type, value: 0: indicating success. Non-0: indicating failure.

V.2.2.1.2 stopServer

Prototype: number stopServer()

Description: After the live broadcast star application exits, call this interface to stop each function of the DTH component.

Parameter: None.

Return: number type, value: 0: indicating success. Non-0: indicating failure.

V.2.2.1.3 getADFinishStatus

Prototype: number getADFinishStatus()

Description: Obtain the processing status of the boot advertisement, that is, obtain whether the advertisement is ready.

Parameter: None.

Return: number type, value: 1 indicating processing is completed and the application can work normally, 0 indicating processing is in progress, and the application needs to continue to wait.

V.2.2.1.4 getAllAds

Prototype: Ad[] getAllAds(epgInfoType,serviceid)

Description: Getting the advertisement image information object.

Parameter: epgInfoType – number type, indicating various types of advertising space, defined as follows:

```
enum {
  EPGINFO_TYPE_EPG_CONFIG_DATA = 0xF0F0,
  EPGINFO_TYPE_BOOTLOGO = 1,
  EPGINFO_TYPE_MAINMENU,
  EPGINFO_TYPE_CHANNEL_LIST,
  EPGINFO_TYPE_FAV_CHANNEL_LIST,
  EPGINFO_TYPE_EPG_LIST,
  EPGINFO_TYPE_PFBAR,
  EPGINFO_TYPE_VOLUMEBAR,
  EPGINFO_TYPE_AUDIOLOGO,
  EPGINFO_TYPE_ERROR,
};
```

serviced – number type, when it is EPGINFO_TYPE_PFBAR or EPGINFO_TYPE_VOLUMEBAR, serviceid is the serviceid of the channel corresponding to the advertisement, otherwise it is 0.

Return: Array of Ad objects.

V.2.2.1.5 getOsdXmlFile

Prototype: String getOsdXmlFile()

Description: Getting the save address of the XML file updated by OSD text.

Parameter: None.

Return: String type, the storage location of the XML file updated by OSD text in the box.

V.2.2.1.6 getOsdInfo

Prototype: String getOsdInfo(tag)

Description: Obtain osd event message that needs to be displayed.

Parameter: tag – String type. It is the prompt message code corresponding to osd.

Return: String type, returning the corresponding osd event message.

V.2.2.1.7 startAdPCT

Prototype: number startAdPCT(transport_stream_id)

Description: Getting the first picture of the boot advertisement. Real-time ad filtering for a certain tp should be turned on instead.

Parameter: transport_stream_id – number type, transport stream id.

Return: number type, value: 0 indicating successful acquisition, non-0 indicating failed acquisition, and the application displaying the default picture.

V.2.2.1.8 stopEMBDAction

Prototype: number stopEMBDAction()

Description: The live broadcast star receives the event information of canceling the emergency broadcast, and calls this interface to make the DTH component stop sending the current emergency broadcast.

Parameter: None.

Return: number type, value: 0: successful operation, non-0: failed operation.

V.2.2.1.9 recordAVBEvent

Prototype: recordAVBEvent(event_id, event_param)

Description: Detect user events and save them.

Parameter: event_id – number type, event number;

event_param – number type, setting the event parameters according to the event number.

Return: None.

V.2.2.1.10 dataBDStart

Prototype: number dataBDStart(file_path)

Description: Provide the storage path of the information service file, and start the information service receiving.

Parameter: file_path – String type, the storage path of the message service file. If the parameter is empty, the default path is /data/db/dth/datadb.

Return: number type, value: 0: successful operation, non-0: failed operation.

V.2.2.1.11 dataBDStop

Prototype: number dataBDStop()

Description: Suspension of message service receiving.

Parameter: None.

Return: number type, value: 0: successful operation, non-0: failed operation.

V.2.2.1.12 getDATABDFinishPercent

Prototype: number getDATABDFinishPercent()

Description: Getting the percentage of download completion.

Parameter: None.

Return: number type, value: 0-100: percentage of download completion, other values: failed operation.

V.2.2.1.13 getDataBDXmlFileData

Prototype: String getDataBDXmlFileData(file_name)

Description: Obtain the corresponding message service data.

Parameter: file_name – String type, the file name of the message service file. If the parameter is empty, the default path is /data/db/dth/datadb.

Return: String type, returning the corresponding message service data.

V.2.2.1.14 GPRSTransmit

Prototype: String GPRSTransmit(conn_type, addr, port, data, timeout,retry_count)

Description: Sending data through GPRS module and receive returned data. (Asynchronous interface)

Parameter: conn_type – number type, the type of GPRS connection server is 0 or 1, 0: TCP; 1: UDP;

addr – String type, the domain name or IP address of the server;

port – number type, the port number of the server;

data – String type, transmitted data;

timeout – number type, the timeout period for waiting for the return of transmission data;

retry_count – number type, the number of retries for transmission failure.

Return: String type: If it is empty, it indicates failure, and if it is not empty, it indicates the data returned by the receiving module.

V.2.2.1.15 getGPRSStatus

Prototype: number getGPRSStatus()

Description: Getting whether GPRS is currently in a normal available state.

Parameter: None.

Return: number type, value: 0: GPRS status is normal, non-0: GPRS status is abnormal.

V.2.2.1.16 getGprsBaseStationInfo

Prototype: number getGprsBaseStationInfo()

Description: Asynchronous interface, obtains current base station information.

Parameter: None.

Return: 0 indicating success, others indicating failure.

V.2.2.1.17 SaveNITServiceUpdateVersion

Prototype: number SaveNITServiceUpdateVersion()

Description: After the application updates the program information, notify dth to save the current service update descriptor version.

Parameter: None.

Return: number type, value: 0: successful save, non-0: failed save.

V.2.2.1.18 GetBouquetId

Prototype: number GetBouquetId()

Description: Getting the current bouquet id.

Parameter: None.

Return: number type, the current bouquetid.

V.2.2.1.19 SaveNITServiceUpdateVersion

Prototype: number SaveNITServiceUpdateVersion()

Description: After the application updates the program information, notify dth to save the current service update descriptor version.

Parameter: None.

Return: number type, 0 indicating success, others indicating failure.

V.2.2.1.20 dataBDdeleteFiles

Prototype: number dataBDdeleteFiles()

Description: Delete all downloaded information service files (called when exiting the information service).

Parameter: None.

Return value: number type, 0 indicating success, others indicating failure.

V.2.3 Ad object

V.2.3.1 Property

The property definition of the Ad object is shown in Table V.2.

Table V.2 – Ad object properties

Property name	Type	Read and write property	Description
adPath	string	Read only	It indicates the storage address of the advertisement.
startDate	string	Read only	It indicates the start date of playing for the advertisement, the format is "YYYY-MM-DD".
startTime	string	Read only	It indicates the start time of playing for the advertisement, the format is "hh:mm:ss".
endDate	string	Read only	It indicates the end date of playing for the advertisement, the format is "YYYY-MM-DD".
endTime	string	Read only	It indicates the ending time of playing for the advertisement, the format is "hh:mm:ss".
Property name	Type	Read and write property	Description
duration	number	Read only	It indicates the playing time of the advertisement, in seconds.
tableExtId	number	Read only	It indicates the type of program-associated advertisement of the advertisement.

V.2.3.2 Method

V.2.3.2.1 getAllAdServices

Prototype: AdService[] getAllAdServices()

Description: Getting all relevant channel parameters and AdService objects of channel-related advertisements.

Parameter: None.

Return value: AdService object array.

V.2.4 AdService object

V.2.4.1 Property

The property definition of the AdService object is shown in Table V.3.

Table V.3 – AdService object properties

Property name	Type	Read and write property	Description
onId	number	Read only	It indicates the onid of the channel-related advertisement.
tdId	number	Read only	It indicates the tdId of the channel-related advertisement.
serviceId	number	Read only	It indicates the serviceId of the channel-related advertisement.
associateType	number	Read only	It indicates the associateType of the channel-related advertisement.

Annex W

JavaScript-Multi-screen Interactive Unit

(This annex forms an integral part of this Recommendation.)

W.1 Overview

This annex defines the JavaScript interface related to the multi-screen interactive unit.

W.1.1 Scene description

1. There are currently two playback devices of TVOS system, namely A and B, and both A and B systems are connected to the same LAN.

2. The application program of A system uses the startMultiScreenServer method to start the multi-terminal linkage service through the MultiScreen multi-terminal linkage communication object provided by the A system, and transmits the multi-terminal linkage component message of the B system to the A system.

3. Then the application uses the findSPs method to search for the B system equipment under the same LAN, and realizes the connection with the B system through the A system through the connect method.

4. After the connection is successful, the A system program receives the connection status information through the MultiScreenEvent interface provided by the B system, and informs the A system of the connection status by the onConnected method. So far, the A and B systems have realized multi-terminal linkage communication. The application can pass the operation data information of the A system or more video data information to the B system by the inputKeyCode provided by the MultiScreen object, and realize the related operation and playback on the B system.

W.2 Multi-screen interactive module

This module supports finding and connection between the client and server in the LAN. The multi-screen interactive component provides a JavaScript interface for the upper application of the Web APP, which can realize the functions of the WEB application in the LAN to find, connect, and control the server equipment.

This module defines the JS object related to multi-screen interaction: MultiScreen.

W.2.1 MultiScreen object

W.2.1.1 Method

W.2.1.1.1 startMultiScreenServer

Prototype: int startMultiScreenServer (String spName, String spDeviceType, String spServiceInfo, String spVersion,String ipaddress, int port, String hostname)

Description:A Remote interface, starting multi-screen interactive component server.

Parameter:

spName – String type, name of multi-screen interactive component server;

spDeviceType – String type, device type of the multi-screen interactive component server;

spServiceInfo – String type, service information of multi-screen interactive component server;

spVersion – String type, version of multi-screen interactive component server;

ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.1.1.2 stopMultiScreenServer

Prototype: int stopMultiScreenServer()

Description: A Remote interface, stoping multi-screen interactive component server.

Parameter: None.

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.1.1.3 startMultiScreenClient

Prototype: int startMultiScreenClient(String clientName)

Description: A Remote interface, starting multi-screen interactive component client.

Parameter: clientName – String type, name of the multi-screen interactive component client.

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.1.1.4 stopMultiScreenClient

Prototype: int stopMultiScreenClient()

Description: A Remote interface, closing the multi-screen interactive component client.

Parameter: None.

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.1.1.5 findSPs

Prototype: int findSPs()

Description: A Remote interface, searching the multi-screen interactive service component device under the LAN.

Parameter: None.

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.1.1.6 connect

Prototype: int connect(String spName, String spDeviceType, String spServiceInfo, String spVersion,String ipaddress, int port, String hostname)

Description: remote interface, which is connected to the server device of the multi-screen interactive service component under the LAN.

Parameter: spName – String type, name of multi-screen interactive component server;

spDeviceType – String type, device type of the multi-screen interactive component server;

spServiceInfo – String type, service information of multi-screen interactive component server;

spVersion – String type, version of multi-screen interactive component server;

ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device.

Return: int type, returning 0 if the remote interface is called successfully,otherwise returning an error code.

W.2.1.1.7 queryInfo

Prototype: int queryInfo(String ipaddress, int port, String hostname,String cmdid,String atrribute,String params)

Description: Remote interface, the multi-screen interactive component client requests to obtain it.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device;

cmdid – String type, instruction id of the request information;

atrribute – String type, instruction name of the request information;

params – String type, instruction parameter of the request information.

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.1.1.8 execCmd

Prototype: int execCmd(String ipaddress, int port, String hostname,String cmd,String param)

Description: Remote interface, the multi-screen interactive component client requests to execute instructions.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device;

action – String type, key instruction;

param – String type, parameter attached to the key instruction.

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.1.1.9 inputKeyCode

Prototype: int inputKeyCode(String ipaddress, int port, String hostname,String action,String param)

Description: Remote interface, the multi-screen interactive component client sends instructions input by virtual keys.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device;

action – String type, key instruction;

param – String type, parameters attached to the key instruction.

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.1.1.10 boardCastAllDevice

Prototype: int boardCastAllDevice(String cmd,String param)

Description: Remote interface, the multi-screen interactive component server sends broadcast to all connected devices.

Parameter: cmd – String type, broadcast instruction;

param – String type, parameters attached to the broadcast instruction;

Return: int type, returning 0 if the remote interface is called successfully, otherwise returning an error code.

W.2.2 Message callback EventHandler

```
interface MultiScreenEvent : Event {
  readonly attribute String spName;
  readonly attribute String spDeviceType;
  readonly attribute String spServiceInfo;
  readonly attribute String spVersion;
  readonly attribute String ipaddress;
  readonly attribute int port;
readonly attribute String hostname;
readonly attribute String id;
readonly attribute String attribute;
readonly attribute String param;
readonly attribute String action;
readonly attribute String cmd;
};
```

W.2.2.1 Method

W.2.2.1.1 onSpFounded

Prototype: [RuntimeEnabled=MultiScreenEvent]EventHandler onSpFounded

Description: Notifying the Web APP multi-screen interactive component that the service search is completed.

Parameter: spName – String type, service name of the searched multi-screen interactive component;

spDeviceType – String type, device type of the searched multi-screen interactive component;

spServiceInfo – String type, server information of the searched multi-screen interactive component;

spVersion – String type, server service version of the searched multi-screen interactive component;

ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device.

W.2.2.1.2 onConnected

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onConnected

Description: Notifying the Web APP multi-screen interactive component that the service connection is completed.

Parameter: spName – String type, service name of the searched multi-screen interactive component;

spDeviceType – String type, device type of the searched multi-screen interactive component;

spServiceInfo – String type, server information of the searched multi-screen interactive component;

spVersion – String type, server service version of the searched multi-screen interactive component;

ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device.

W.2.2.1.3 onConnectRefused

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onConnectRefused

Description: Notifying the Web APP multi-screen interactive component that the service connection is refused.

Parameter: spName – String type, service name of the searched multi-screen interactive component;

spDeviceType – String type, device type of the searched multi-screen interactive component;

spServiceInfo – String type, server information of the searched multi-screen interactive component;

spVersion – String type, service version of the searched multi-screen interactive component server;

ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device.

W.2.2.1.4 onDisconnected

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onConnectRefused

Description: Notifying the Web APP multi-screen interactive component that the service connection is refused.

Parameter: spName – String type, service name of the searched multi-screen interactive component;

spDeviceType – String type, device type of the searched multi-screen interactive component;

spServiceInfo – String type, server information of the searched multi-screen interactive component;

spVersion – String type, service version of the searched multi-screen interactive component server;

ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device.

W.2.2.1.5 onServiceActivited

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onServiceActivited

Description: Notifying the Web APP multi-screen interactive component that the service is activated.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device.

W.2.2.1.6 onServiceDeactived

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onServiceDeactived

Description: Notifying the Web APP multi-screen interactive component that the service is unregistered.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device.

W.2.2.1.7 onQueryInfo

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onQueryInfo

Description: Notifying the Web APP that the data request sent by the client of the multi-screen interactive component has been received.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device;

id – String type, instruction id of the received request;

attribute – String type, instruction property of the received request;

param – String type, parameters attached to the received request.

W.2.2.1.8 onQueryResponse

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onQueryResponse

Description: Notifying the Web APP that the multi-screen interactive component server has responded to the data request.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device;

id – String type, instruction id of the received request;

attribute – String type, instruction property of the received request;

param – String type, parameters attached to the received request.

W.2.2.1.9 onExecute

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onExecute

Description: Notifying the Web APP that the client of the multi-screen interactive component has sent an execution instruction request.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device;

cmd – String type, instructions executed;

param – String type, parameters attached to the instruction executed.

W.2.2.1.10 onInputKeyCode

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onInputKeyCode

Description: Notifying the Web APP that the client of the multi-screen interactive component has sent a request to perform key injection.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device;

cmd – String type, instructions executed;

param – String type, executing the parameters attached to the instruction.

W.2.2.1.11 onNotify

Prototype: [RuntimeEnabled=MultiScreenEvent] EventHandler onNotify

Description: Notifying the Web APP that the multi-screen interactive component has received a notification.

Parameter: ipaddress – String type, ip address of the searched multi-screen interactive component device;

port – int type, port number of the searched multi-screen interactive component device;

hostname – String type, host name of the searched multi-screen interactive component device;

cmd – String type, received instruction;

param – String type, instruction parameter received.

Annex X

JavaScript-DRM management unit

(This annex forms an integral part of this Recommendation.)

X.1 Overview

This specification defines JavaScript interfaces of the DRM management module.

X.2 DRM management module

X.2.1 DrmManager object

Example of using DrmManager object:

```
var drmManager=new DrmManager();//create DrmManager object
var uuid=new String("2d7d041436f2a048a0c4c1cccbb64546");
drmManager.Drm_RegisterApp("unitend",uuid, 0, register_pridata,"unitend",1,0,0);//DRM APP register
var resultdata=drmManager.Drm_SendCommandToTA(1,"unitend");//send commands to TEE
drmManager.Drm_UnRegisterApp();//unregister DRM APP
```

X.2.1.1 Message

The message definition sent by the DRM module to the application layer is shown in Table X.1.

Table X.1 – DRM module message definition

Message name	event. which	event. modifiers	Message description
MSG_DRM_LICENSEREQ	11000	number	DRM license acquisition message, message character string JSON format: {"DrmLicensereqData":param1, "DrmLicenseDataLen":param2}. Among them, param1 is the data related to obtaining the license; param2 is the length of the data related to obtaining the license.
MSG_DRM_DECRYPTREQ	11001	number	DRM decrypt message, message character string JSON format {"DrmDecryptreqData":param1, "DrmDecryptreqDataLen": param2}. Among them, param1 is the relevant data used to decrypt the stream; param2 is the length of the relevant data used to decrypt the stream.
MSG_DRM_MESSAGE	11002	number	DRM notification message, message character string JSON format: {"DrmMessageType":param1, "DrmMessage":param2, "DrmMessageLen": param3}. Among them, param1 is the type of notification message; param2 is the data transmitted by the notification message; param3 is the length of the data transmitted by the notification message.

X.2.1.2 Method

X.2.1.2.1 Drm_RegisterApp

Prototype: number Drm_RegisterApp (DrmSystemID, TAUUID register_commandid,register_pridata,enflag,licensereq_commandId,decrypt_commandId)

Description: DRM APP registration extension, licensereq and decrypt command ID can be privately defined.

Parameter: DrmSystemID – string type, indicating the unique identifier of DRM APP;

TAUUID – string type, indicating the unique identifier of the TApp corresponding to the DRM APP;

register_commandid – number type, indicating the registered commandid for communication with TApp;

register_pridata – string type, indicating the private registration data carried by the registration of communication with TApp, which is used for TApp to verify the legality of DRM APP;

enflag – number type, indicating a decryption call method;

licensereq_commandId – number type, indicating the commandid corresponding to the query license;

decrypt_commandId – number type, indicating the commandid corresponding to the decrypt data.

Return: number type, 0 indicating success, non-0 indicating failure.

X.2.1.2.2 Drm_UnRegisterApp

Prototype: number Drm_UnRegisterApp()

Description: DRM APP unregistration.

Parameter: None.

Return: number type, 0 indicating success, non-0 indicating failure.

X.2.1.2.3 Drm_SendCommandToTA

Prototype: string Drm_SendCommandToTA(commandId, sendData)

Description: Send commands to TEE.

Parameter: commandId – string type, indicating the command ID;

sendData – string type, indicating the data sent.

Return: string type, returning the TEE processing result in JSON format.

X.2.1.2.4 Drm_SendMessageToPlayer

Prototype: number Drm_SendMessageToPlayer(type,message)

Description: Send a message to the player.

Parameter: type – number type, indicating the message type;

Message – string type, indicating message data.

Return: number type, 0 indicating success, non-0 indicating failure.

Annex Y

JavaScript-DCAS management unit

(This annex forms an integral part of this Recommendation.)

Y.1 Overview

This annex defines the functional modules related to DCAS management. DCAS client software and application software can be developed through the DCAS JavaScript application programming interface.

Y.2 EPG DCAS module

This module defines the DCAS module and EPG-related JavaScript objects: EPG_DCAS object.

Y.2.1 EPG_DCAS object

The EPG_DCAS object is a built-in object, which provides the JavaScript method of the EPG_DCAS module.

Y.2.1.1 Method

Y.2.1.1.1 getActivationStatus

Prototype: number getActivationStatus()

Description: Getting the activation status.

Parameter: None.

Return: number type, value: 0-indicating it has been activated. Non-0 indicating that the status is unknown.

Y.2.1.1.2 getBeidouInfo

Prototype: string getBeidouInfo()

Description: Getting current Beidou information.

Parameter: None.

Return: String type, the current Beidou information, the character string JSON format is:

{

"islocked":param1,

"latitude":param2,

"longitude":param3

"sat num":param4

"SNR":param5

}.

Y.2.1.1.3 getCASVersion

Prototype: string getCASVersion()

Description: Getting CAS version information.

Parameter: None.

Return: String type, value: if it is empty, it indicates that the CAS version information is not obtained, and if it is not empty, it indicates that the CAS version information is received.

Y.2.1.1.4 getCASVenderId

Prototype: number getCASVenderId()

Description: Getting the current CAS vendor ID.

Parameter: None.

Return: number type, value: 0 indicating that the current CAS vendor ID is not obtained, and non-0 indicating that the current CAS vendor ID is obtained.

Y.2.1.1.5 getChipID

Prototype: string getChipID()

Description: Getting the chip ID.

Parameter: None.

Return: String type, value: if it is empty, it indicates that the chip ID is not obtained, and if it is not empty, it indicates that the chip ID is obtained.

Y.2.1.1.6 getHSMID

Prototype: string getHSMID()

Description: Getting the HSM chip ID.

Parameter: None.

Return: String type, value: if it is empty, it indicates that the HSM chip ID is not obtained, and if it is not empty, it indicates that the HSM chip ID is obtained.

Y.2.1.1.7 getValidPosition

Prototype: string getValidPosition()

Description: Getting valid location information allowed by the current set-top box.

Parameter: None.

Return: String type, value: if it is empty, it indicates that no valid location information is obtained, and if it is not empty, it indicates that valid location information is obtained. The return value is in JSON format:

```
{
"latitude":param1,
"longitude":param2
}.
```

Y.2.1.1.8 getPersonalBits

Prototype: number getPersonalBits()

Description: Getting regional personal Bits.

Parameter: None.

Return: number type, value: 0 indicating that the regional personal Bits are not obtained, and non-0 indicating that the regional personal Bits are obtained.

Y.2.1.1.9 getZipCode

Prototype: string getZipCode()

Description: Getting the region code.

Parameter: None.

Return: String type, value: if it is empty, it indicates the area code is not obtained, and if it is not empty, it indicates the area code is obtained.

Y.3 DCAS_APP module

See Table Y.1 for the description of DCAS_APP module objects.

Table Y.1 – Object description of DCAS_APP module

Object name	Description
JSDCAS.CASDescriptor	CA description object, this object is used to express the CA descriptor in PMT or CAT.
JSDCAS.CASEcmEvent	ECM event object, this object contains information about ECM events.
JSDCAS.CASEmmEvent	ECM event object, this object contains information about EMM events.
JSDCAS.CASFilter	Filter object, this object expresses the filter conditions required when filtering in-band or out-of-band EMM.
JSDCAS.CASM	The global object of CASM, from which you can access the CAS manager and controller objects.
JSDCAS.CASModule	CAS module object interface, implemented by the JS DCAS application, is an interface provided to the platform.
JSDCAS.CASModuleManager	CASModuleManager object, this object provides the interface required by JSDCAS application.
JSDCAS.CASPacketEvent	CAS data packet event object notifies the JS DCAS application of out-of-band CAS data packet events.
JSDCAS.CASSession	CAS Sesion object, the platform generates a CAS Session object for each descrambling request.
JSDCAS.CASStatus	CAS status object, through this object, JSDCAS transfers the descrambling status to the platform.
JSDCAS.TeeController	TEE control right object, controller for JS DCAS and TEE communication.
JSDCAS.TeeRetVal	TEE return object, the object contains data, errors and other information returned from TEE.

Y.3.1 Application interface call time sequence

The basic time sequence of the DCAS application interface is shown in Figure Y.1.

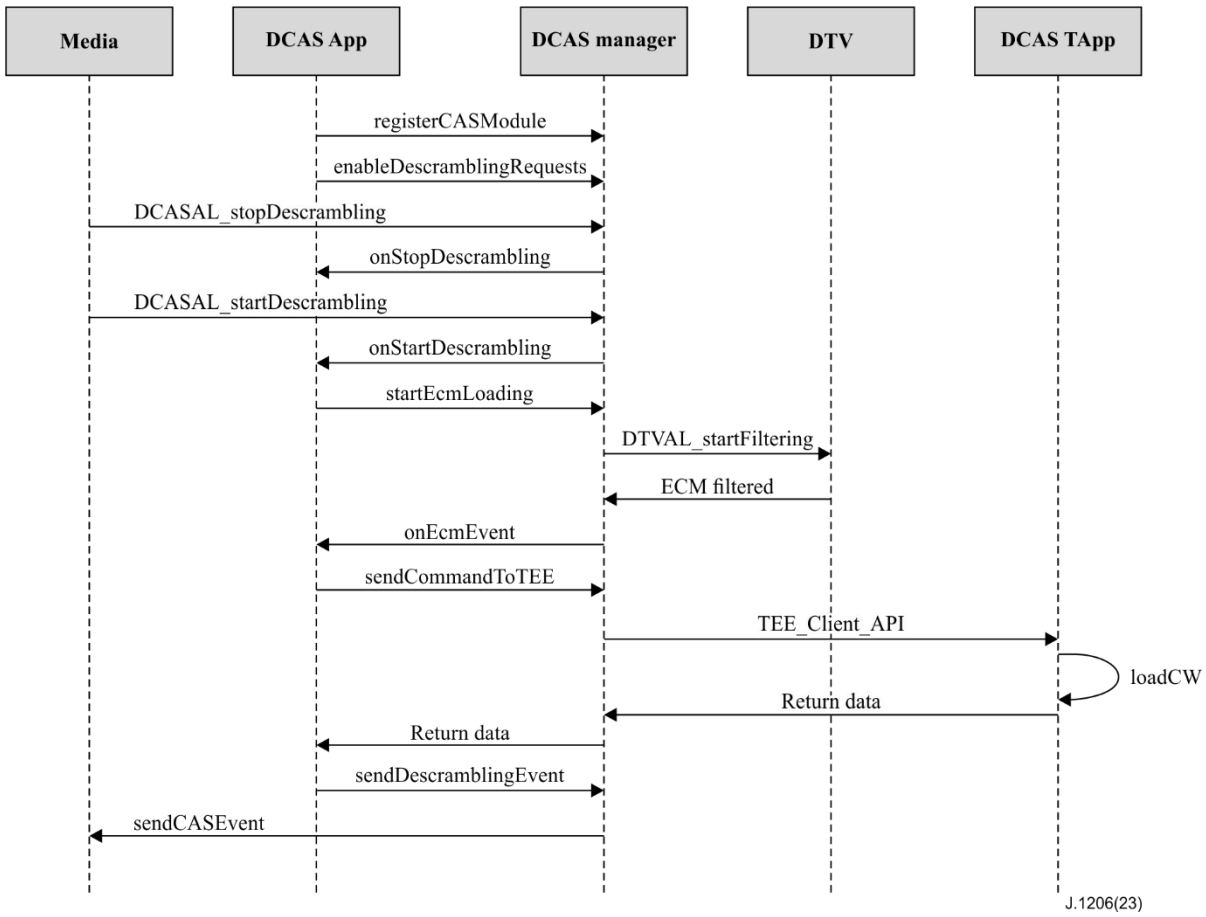


Figure Y.1 – Basic call time sequence of DCAS application interface

Y.3.2 JSDCAS.CASDescriptor object

This object is used to express the CA descriptor in PMT or CAT.

Y.3.2.1 Method

Y.3.2.1.1 getCasId

Prototype: number getCasId()

Description: This method is provided by the terminal software platform and returns the CASID in the CA descriptor.

Parameter: None.

Return: number type, indicating CASID.

Y.3.2.1.2 getPid

Prototype: number getPid()

Description: Returning PID in the CA descriptor. If the descriptor comes from CAT, it indicates EMM PID. If the descriptor comes from PMT, it indicates ECM PID.

Parameter: None.

Return: number type, return ECMPID.

Y.3.2.1.3 getPrivateData

Prototype: Uint8Array getPrivateData()

Description: Returning the private data in the CA descriptor, and the private data is returned in the form of Uint8Array.

Parameter: None.

Return: Uint8Array type, indicating private data.

Y.3.3 JSDCAS.CASEcmEvent object

This object contains information about ECM events, which are passed to the JS DCAS application through CASModule.onEcmEvent or CASModuleManager.onStartDescrambling. It can express that ECM packet is received, or timed out, or an internal error in the filter.

Y.3.3.1 Method

Y.3.3.1.1 getEcmData

Prototype: Uint8Array getEcmData()

Description: Returning complete ECM data. If it is a timeout or an internal error for the event, null is returned.

Parameter: None.

Return: Uint8Array type, indicating ECM data.

Y.3.3.1.2 getError

Prototype: number getError()

Description: Error when returning the internal Section Filter filter. Only used for debugging. This method can only be called when the event does not provide any other information.

Parameter: None.

Return: number type, error message.

Y.3.3.1.3 getTableId

Prototype: number getTableId()

Description: Table ID when returning the ECM packet. This method can only be called when the event provides ECM data.

Parameter: None.

Return: number type, indicating TableID.

Y.3.3.1.4 isTimeout

Prototype: boolean isTimeout()

Description: Whether it is timed out when returning the getting of the first ECM packet. The timeout period can be set in CASModuleManager.enableDescramblingRequests. This method can only be called when the event does not provide ECM data.

Parameter: None.

Return: boolean type, true-timeout; false-no timeout.

Y.3.4 JSDCAS.CASEmmEvent object

This object contains information about EMM events, which are passed to the JS DCAS application through CASModule.onInBandEmmEvent. This event may be received because of the receiving of CAT or any EMM packet, or an internal filter error.

Y.3.4.1 Method

Y.3.4.1.1 getEmmData

Prototype: Uint8Array getEmmData()

Description: Returning complete EMM data. If the event is caused by CAT update, or caused by an internal error, null is returned.

Parameter: None.

Return: Uint8Array type, EMM data.

Y.3.4.1.2 getError

Prototype: number getError()

Description: Error when returning the internal Section Filter filter. Only used for debugging. This method can only be called when the event does not provide any other information.

Parameter: None.

Return: number type, error message.

Y.3.4.1.3 getTableId

Prototype: number getTableId()

Description: Table ID when returning the EMM packet.

Parameter: None.

Return: number type, indicating TableID.

Y.3.4.1.4 isCatUpdateNotification

Prototype: boolean isCatUpdateNotification()

Description: Whether the return event is caused by CAT update, if it is caused by CAT update, the EMM data should be empty.

Parameter: None.

Return: boolean type.

–true – the event is caused by a CAT update;

–false – the event is caused by any EMM packet, or caused by an internal error.

Y.3.5 JSDCAS.CASFilter object

This object is used to express the Section Filter filter conditions required when filtering in-band or out-of-band EMM. The platform should only call the CAS Module when the data packet matches the filter conditions. For those data packets that do not meet the conditions, they should be discarded by the platform, and the JS DCAS application should not be called. CASFilter objects (or object arrays) can be set to the platform through CASModuleManager.startCasPacketLoading and CASModuleManager.startInbandEmmLoading. Filter conditions include:

1. Offset in bytes. The data before the offset will be ignored and will not participate in the comparison.

2. Value used to compare with the data received by the platform, expressed in Bitmap.

3. It is used to express which bits need to participate in the comparison. For the bits set to 0 in the Bitmap, no comparison is required.

Y.3.5.1 Method

Y.3.5.1.1 getBitmapMask

Prototype: Uint8Array getBitmapMask()

Description: Returning the filter mask.

Parameter: None.

Return: Uint8Array type, return the filter mask.

Y.3.5.1.2 getBitmapValue

Prototype: Uint8Array getBitmapValue()

Description: Returning bitmap value used for comparison.

Parameter: None.

Return: Uint8Array type, return the bitmap value.

Y.3.5.1.3 getOffset

Prototype: number getOffset()

Description: Returning offset (in bytes).

Parameter: None.

Return: number type, indicating the offset.

Y.3.6 JSDCAS.CASM object

CASM is a global object, from which all CAS Manager and Controller objects can be accessed.

Y.3.6.1 Method

Y.3.6.1.1 getCASModuleManager

Prototype: JSDCAS.CASModuleManager getCASModuleManager()

Description: Returning CASModuleManager object instance.

Parameter: None.

Return: CASModuleManager object.

Y.3.6.1.2 getTeeController

Prototype: JSDCAS.TeeController getTeeController()

Description: Returning TEEController object instance.

Parameter: None.

Return: TeeController object.

Y.3.7 JSDCAS.CASModule object

CASModule is an interface of CAS module objects, which should be implemented by JS DCAS application and registered in the CAS Module Manager of the platform to receive descrambling requests, ECM, EMM or any metadata that the JS DCAS application cares about. EMM can be obtained in-band or out-of-band. Determined by the end-to-end design and the capabilities of the platform and equipment.

Y.3.7.1 Method

Y.3.7.1.1 getCasId

Prototype: number getCasId()

Description: Returning the unique CASID of the CAS module. This method must be implemented before calling CASModuleManager.registerCASModule. The returned value is the value expected to appear in CA descriptor in CAT or PMT.

Parameter: None.

Return: number type, indicating CASID.

Y.3.7.1.2 onCasPacketEvent

Prototype: onCasPacketEvent(casPacketEvent)

Description: The platform calls this method when it receives out-of-band EMM (or other out-of-band CAS data packets), refer to CASModuleManager.startCasPacketLoading.

Parameter: casPacketEvent – CASPacketEvent object, which contains the CASPacketEveng object instance of the out-of-band EMM or other out-of-band CAS data packets.

Y.3.7.1.3 onEcmEvent

Prototype: onEcmEvent(casSession,ecmEvent)

Description: After the platform filters the new ECM packet, call this method of the JS DCAS application. In fast mode, the platform calls this method after setting CW in ECM to K-LAD.

Parameter: casSession – CASSession object, CAS Session object obtained from CASModule.onStartDescrambling.

CASEcmEvent ecmEvent-ECM event object.

Y.3.7.1.4 onInbandEmmEvent

Prototype: onInbandEmmEvent(casSessionForEMM,emmEvent)

Description: The platform calls this method when it receives a CAT update or in-band EMM. Refer to CASModuleManager.startInbandEmmLoading.

Parameter: casSessionForEMM – A CASSession object, a special CAS Session, this session is related to the TS where the CAT is located, and it also contains the CA descriptor in the CAT (not the CA descriptor in the PMT). The platform should create a CAS Session specifically for this purpose. Note that this CAS Session object may only have some fields valid. Note: If the CA descriptor of the CAS is not specified in the CAT, or the terminal leaves the current frequency point, the CAT update event will still be sent JS DCAS application, but CAS Session is null.

emmEvent – CASEmmEvent object, which contains CAT update or EMM, or other wrong CASEmmEvent object instances.

Y.3.7.1.5 onStartDescrambling

Prototype: onStartDescrambling(casSession,firstEcmEvent)

Description: This method is called by the platform when there is a new descrambling request. It usually happens when the platform starts playing a new scrambled channel. JS DCAS application will receive this descrambling request only upon CASModuleManager.enableDescramblingRequests being called. In the auto-load mode, the platform will automatically start filtering the first ECM, and call this method after receiving the ECM. In this case, the JS DCAS application does not need to call CASModuleManager.startEcmLoading. If the device has multiple Tuners, this method may be called multiple times at the same time, each time corresponding to a playback descrambling request. Each

request will have a different CAS Session. In addition, if the basic stream scrambling method of the service that needs to be descrambled is different, this method may also be called multiple times. Similarly, each call also has a different CAS Session.

Parameter: casSession – A CASSession object, CAS Session object generated by the platform for a specific descrambling request. Each object has a unique session ID, and all information about the service and the basic stream. It also contains the CA descriptor corresponding to this CAS module in the PMT.

firstEcmEvent – A CASEcmEvent object, in auto-load mode, the first ECM packet received by the platform (or timeout, error). If it is not in auto-load mode, it is empty.

Y.3.7.1.6 onStopDescrambling

Prototype: onStopDescrambling(casSession)

Description: The platform calls this method when it stops playing the current channel to notify the JS DCAS application to stop descrambling.

Parameter: casSession – A CASSession object, CAS Session obtained in CASModule.onStartDescrambling.

Y.3.8 JSDCAS.CASModuleManager object

JS DCAS application uses this CAS Module Manager object to receive descrambling requests, ECM, EMM, and report CAS descrambling status. The JS DCAS application should implement the CAS Module object, and then register it to the CAS Module Manager.

Y.3.8.1 Method

Y.3.8.1.1 disableDescramblingRequests

Prototype: number disableDescrambingRequests(casModule)

Description: This method is called by a JSDCAS application to stop receiving descrambling requests. This method rarely has a chance to be called. For example, when the application needs to reconfigure the descrambling request parameters, it needs to call this method to suspend the receiving, and then restart the receiving. Or the application wishes to close itself. Recall CASManager.enableDescramblingRequests to restart receiving.

Parameter: casModule – CASModule object, CAS module instance.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error.

Y.3.8.1.2 enableDescramblingRequests

Prototype: number enableDescramblingRequests(casModule,firstEcmTimeout,

autoLoadFirstEcm,isFastMode,ecmTableIds)

Description: JS DCAS application starts to receive descrambling requests by calling this method. Through different parameters, different working modes between the platform CAS Manager and the CAS module can be configured. This method is usually only called after the CAS module is registered, because usually the JS DCAS application will not change this way of working. If the JS DCAS application wants to change this working mode, it needs to call CASModuleManager.disableDescramblingRequests first, and then call this method again.

Parameter: casModule – A CASModule object, CAS module instance;

firstEcmTimeout – number type, in milliseconds, the longest time the platform waits for the first ECM. If it times out, the CAS module will receive the CASEcmEvent through the onEcmEvent call or the onSTartDescrambling call;

autoLoadFirstEcm – boolean type, specify whether it is auto-load mode or not. Auto-load mode indicates that the platform automatically starts to filter the first ECM after the JS DCAS application calls this method, without waiting for the JS DCAS application to call startEcmLoading.

isFastMode – boolean type, fast mode (temporarily only a placeholder, and has no practical meaning);

ecmTableIds – Array type, if the JS DCAS application wants to specify tableID of ECM.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – the platform does not implement a specific mode;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error.

Y.3.8.1.3 fetchDataFromCasHeadend

Prototype: number fetchDataFromCasHeadend(casModule,inputData,casHeURI)

Description: This method is called by JS DCAS application to obtain data from the headend, via the platform, via GPRS, or other possible means in the future.

Parameter: casModule – A CASModule object, CAS module instance.

inputData – Uint8Array type, data to be sent to the headend.

casHeURI – String type, URI of the headend service.

Return: number type.

success – Data returned by the headend.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported;

CASModuleManager.ACTION_ERROR_NETWORK – Network error.

Y.3.8.1.4 registerCASModule

Prototype: number registerCASModule(vendorId,casModule,networkPriority,applicationContext)

Description: A JS DCAS application, registering itself in the CAS Module Manager of the platform by calling this method.

Parameter: vendorId – number type, Vendor Id of CAS. Each CAS manufacturer has a unique special ID.

casModule – CASModule object, CAS module instance to be registered.

networkPriority – number type, if CASModuleManager allows registration of more than one CAS module, this optional parameter expresses the priority between multiple modules, and the specific

value is determined by the operator. The greater the absolute value, the higher the priority, for example, the priority of 3 is higher than 2. After knowing the priority, when the platform encounters a situation where there are multiple CA descriptors in the PMT, it should send a descrambling request to the CAS module with the highest priority according to the priority. If the operator does not set the priority, each JS DCAS application must use 0 as a parameter. In this case, which CAS module can receive the descrambling request is determined by the implementation of the platform.

applicationContext – Platform-related application parameters. Usually this parameter is told to the application by the platform when it is initialized. The usage of this parameter is project-dependent.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_SECURITY – the caller does not have permission to access CASModuleManager;

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter.

Y.3.8.1.5 removeCASModule

Prototype: number removeCASModule(vendorId,casModule,applicationContext)

Description: This method is called by a JS DCAS application to delete a CAS module from the CAS Module Manager. This method is rarely called, for example, before the application wants to change a CAS ID or before the application closes itself.

Parameter: vendorId – number type, VendorId of CAS. Each CAS manufacturer has a unique special ID.

casModule – CASModule object, CAS module instance to be registered.

applicationContext – Platform-related application parameters. Usually this parameter is told to the application by the platform when it is initialized. The usage of this parameter is project-dependent.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error;

CASModuleManager.ACTION_ERROR_SECURITY – the caller does not have permission.

Y.3.8.1.6 sendCommandToSTB

Prototype: number sendCommandToSTB(casModule,inputData)

Description:A Data channel function. This method is called by a JS DCAS application to send data to DCAS Manager, DCAS Manager forwards commands to the corresponding module for processing, these commands include OSD, upgrade trigger, fingerprint, emergency broadcast, ratings survey, etc., these commands are sent by BOSS, DCAS is only responsible for forwarding and is used as a data channel.

Parameter: casModule – CASModule object, CAS module instance to be registered.

inputData – Uint8Array type, data to be sent to the DCAS manager.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported;

CASModuleManager.ACTION_ERROR_NETWORK – Network error.

Y.3.8.1.7 sendDataToHeadend

Prototype: number sendDataToHeadend(casModule,inputData)

Description: This method is called by JS DCAS application to send data to the headend via the platform, via GPRS, or other possible means in the future.

Parameter: casModule – CASModule object, CAS module instance to be registered.

inputData – Uint8Array type, data to be sent to the headend.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported;

CASModuleManager.ACTION_ERROR_NETWORK – Network error.

Y.3.8.1.8 sendDescramblingEvent

Prototype: number sendDescrambingEvent(casModule,casSession,casStatus)

Description: This method is called by a JS DCAS application to report CAS status. CAS status includes the success or failure of descrambling. Every time the status changes, JS DCAS should report the status.

Parameter: casModule – CASModule object, CAS module instance.

casSession – CASSession object, CASSession obtained from CASModule.onSTartDescrambling.

casStatus – CASStatus object, CASStatus object generated by JSDCAS application.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

Y.3.8.1.9 sendFreeTextOSD

Prototype: number sendFreeTextOSD(casModule,inputData,flags)

Description: JS DCAS application forwards the received text information to the platform by calling this method. The platform may forward this text information to the UI application or process it by itself.

Parameter: casModule – CASModule object, CAS module instance.

inputData – Uint8Array type, text information.

flags – ArrayBuffer type, used to display additional information such as format, project-related.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

Y.3.8.1.10 setCCIBits

Prototype: number setCCIBits(casModule,casSession,cciBits)

Description: Setting CCI (Copy Control Information) data bit.

Parameter: casModule – CASModule object, CAS module instance.

casSession – CASSession object, CAS Session obtained from CASModule.onSTartDescrambling.

cciBits – number type, CCI data bit.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

Y.3.8.1.11 setData

Prototype: number setData(casModule, propertyId, propertyType, propertyValue)

Description: DCAS APP sets property values for the platform, including BouquetID, activation status, CAS information, Beidou information, ChipID, HSMID, CASVenderID, area code, CA version, etc.

Parameter: casModule – CASModule object, CAS module instance;

propertyId – number type, property ID, see JSDCAS.CASModuleManager.PROP_ID_xxx;

propertyType – number type, property type, see JSDCAS.CASModuleManager.PROP_TYPE_xxx;

propertyValue – number type, property value.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter ;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

Y.3.8.1.12 setPinCode

Prototype: number setPinCode(casModule, pinCode)

Description: Notifying the platform to reset PIN.

Parameter: casModule – CASModule object, CAS module instance.

pinCode – number type, PIN code.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

Y.3.8.1.13 setServiceListFilter

Prototype: number setServiceListFilter(casModule,filterData)

Description: Setting the filter conditions of the service list. The definitions of the filter conditions are platform-dependent.

Parameter: casModule – A CASModule object, CAS module instance.

filterData – number type, filter condition.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

Y.3.8.1.14 startCasPacketLoading

Prototype: number startCasPacketLoading(casModule,cableModemFilter,sourceURL,casFilter)

Description: This method is called by a JS DCAS application to start receiving out-of-band CAS data packets. CAS data packets can be EMM or other out-of-band metadata. The receiving method is determined by the device hardware, platform, network, etc. For device with Cable Modem, it can be received under ADSG or BDSG protocol, and it can also be received by adding a multicast address through IP over Cable. For IP television (IPTV) device, it can be received by adding a multicast address via Ethernet or Wifi. It can also be received via a local UDP socket. In either case, JS DCAS applications can perform processing through CASModule.onCasPacketEvent when receiving CAS data packets.

NOTE – The platform may implement receiving of data packets from multiple different sources at the same time. In this case, this method may be called multiple times from different URLs.

Parameter: casModule – CASModule object, CAS module instance.

cableModemFilter – number type, in the case of Cable Modem and DSG tunnelD, a filter must be provided. In the case of non-DSG, this parameter should be null.

sourceURL – String type, if data packets are received from UDP, this parameter needs to be provided. "udp://@127.0.0.1:4444" or "udp://@localhost:4444" is received from local UDP port.

casFilter – CASFilter object, filter condition, it can be a CASFilter array.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

Y.3.8.1.15 startEcmLoading

Prototype: number startEcmLoading(casModule,casSession)

Description: JS DCAS application initiates the receiving of ECM of a specific scrambled program by calling this method and calls it after receiving the scramble request.

NOTE – In auto-load mode, there is no need to call this method.

Parameter: casModule – A CASModule object, CAS module instance.

casSession – CASSession object, CASSession obtained from CASModule.onSTartDescrambling.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error.

Y.3.8.1.16 startInbandEmmLoading

Prototype: number startInbandEmmLoading(casModule,emmTableIds,casFilter, includeCatNotifications)

Description: This method is called by a JS DCAS application to start receiving in-band EMM. If required by JS DCAS application, it can also be used to receive CAT. When there is EMM or CAT update, the JS DCAS application receives data through CASModule.onInbandEmmEvent.

Parameter: casModule – CASModule object, CAS module instance.

emmTableIds – Array type, EMM Table Id array.

casFilter – CASFilter|Array type, CAS filter. Only the data that meets the conditions will be notified to the application by the platform, and it is also possible to pass a Filter array.

includeCatNotifications – boolean type, specify whether CAT update notification is expected to be received.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error;

CASModuleManager.ACTION_ERROR_ACTION_NOT_SUPPORTED – Method not supported.

Y.3.8.1.17 stopCasPacketLoading

Prototype: number stopCasPacketLoading(casModule,cableModemFilter,sourceURL)

Description: This method is called by a JS DCAS application to stop receiving out-of-band CAS data packets.

Parameter: casModule – CASModule object, CAS module instance.

cableModemFilter – number|string type, required by Cable Modem.

sourceURL – String type, required when receiving via UDP.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error.

Y.3.8.1.18 stopEcmLoading

Prototype: number stopEcmLoading(casModule,casSession)

Description: JS DCAS application stops receiving ECM by calling this method. JS DCAS applications rarely have the opportunity to call this method. If ECM receiving is expected to be restarted, CASManager.startEcmLoading needs to be recalled again.

Parameter: casModule – CASModule object, CAS module instance.

casSession – CASSession object, CASSession obtained from CASModule.onSTartDescrambling.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error.

Y.3.8.1.19 stopInbandEmmLoading

Prototype: number stopInbandEmmLoading(casModule)

Description: JS DCAS application stops receiving in-band EMM by calling this method. This method rarely needs to be called. Recall CASManager.startInbandEmmLoading to restart receiving.

Parameter: casModule – CASModule object, CAS module instance.

Return: number type.

success – CASModuleManager.ACTION_OK.

failure – Returning the following error values:

CASModuleManager.ACTION_ERROR_INVALID_PARAMETERS – invalid parameter;

CASModuleManager.ACTION_ERROR_DRIVER – low-level error.

Y.3.9 JSDCAS.CASPacketEvent object

It is used to notify the JS DCAS application of out-of-band CAS data packet events.

Y.3.9.1 Method

Y.3.9.1.1 getCableModemFilter

Prototype: number|string getCableModemFilter()

Description: Returning cableModemFilter used to filter this packet. If Cable Modem DSG is not used, it returns empty.

Parameter: None.

Return: number|string type.

ADSG mode – return CAS Tunner ID, number type.

BDSG mode – Returning virtual MAC address.

Y.3.9.1.2 getPacketData

Prototype: Uint8Array getPacketData()

Description: Returning the data in the data packet.

Parameter: None.

Return: Uint8Array type.

Y.3.9.1.3 getPacketHeader

Prototype: Uint8Array getPacketHeader()

Description: Returning the header of the data packet. The header contains the IP address and UDP header.

Parameter: None.

Return: Uint8Array type.

Y.3.9.1.4 getSourceURL

Prototype: string getSourceURL()

Description: Returning the source address of the CAS packet received via UDP.

Parameter: None.

Return: string type, source address character string.

Y.3.10 JSDCAS.CASSession object

The platform generates a CAS Session object for each descrambling request. It contains the unique session ID, and all the information about the broadcast program, and also contains the CA descriptor related to this descrambling in the PMT. For each descrambling request, the JS DCAS application obtains the CAS Session through CASModule.onStartDescrambling. The CASSession object can also be used in the scenario of receiving CAT update messages. In this scenario, only some fields of the CASSession object are valid.

Y.3.10.1 Method

Y.3.10.1.1 GetCasDescriptor

Prototype: CASDescriptor getCasDescriptor()

Description: Returning CA descriptor, which may be from PMT or CAT.

Parameter: None.

Return: CASDescriptor object, CA descriptor object instance.

Y.3.10.1.2 getChannelNumber

Prototype: number getChannelNumber()

Description: Returning channel number. This method is optional, especially for those platforms that cannot determine the channel number, it can return 0.

Parameter: None.

Return: number type, channel number.

Y.3.10.1.3 getNetworkId

Prototype: number getNetworkId()

Description: Returning original network ID. This method is optional, if the platform is not available, it can return 0.

Parameter: None.

Return: number type, original network ID.

Y.3.10.1.4 getOperationType

Prototype: number GetOperationType()

Description: Returning operation type.

Parameter: None.,

Return: number type, operation type value:

CASSession.OPERATION_TYPE_PRESENTATION ;

CASSession.OPERATION_TYPE_RECORDING ;

CASSession.OPERATION_TYPE_BUFFERING ;

CASSession.OPERATION_TYPE_SECOND_DEVICE.

Y.3.10.1.5 getProgramNumber

Prototype: number getProgramNumber()

Description: Returning program number.

Parameter: None.

Return: number type, program number.

Y.3.10.1.6 getServiceIdentifier

Prototype: number getServiceIdentifier()

Description: Returning identifier of the service being descrambled, which is a value or an object;

Parameter: None.

Return: number type, identifier of service.

Y.3.10.1.7 getSessionId

Prototype: number getSessionId()

Description: Returning SessionID.

Parameter: None.

Return: number type, indicating SessionID.

Y.3.10.1.8 getStreamPath

Prototype: Uint8Array getStreamPath()

Description: Returning StreamPath data.

Parameter: None.

Return: Uint8Array type, indicating StreamPath data.

Y.3.10.1.9 getStreamPIDs

Prototype: Array getStreamPIDs()

Description: Returning Stream PIDs list.

Parameter: None.

Return: Array type, PID list.

Y.3.10.1.10 getStreamTypes

Prototype: Array getStreamTypes()

Description: Returning StreamTypes list.

Parameter: None.

Return: Array type, Streamtypes list.

Y.3.10.1.11 getTransmitterScrambingMode

Prototype: number getTransmitterScrambingMode()

Description: Returning scrambling mode value.

Parameter: None.

Return: number type, scrambling mode.

Y.3.10.1.12 getTransportStreamId

Prototype: number getTransportStreamId()

Description: Returning TSID being descrambled.

Parameter: None.

Return: number type, indicating TSID.

Y.3.10.1.13 getTunerId

Prototype: number getTunerId()

Description: Returning TunerID used by the program being descrambled.

Parameter: None.

Return: number type, indicating TunerID.

Y.3.11 JSDCAS.CASStatus object

Y.3.11.1 Overview of the JSDCAS.CASStatus object

JS DCAS application transmits the descrambling state to the platform through this object. Every time there is a change in the descrambling state, the JS DCAS application should call CASModuleManager.sendDescramblingEvent to notify the platform. After the platform receives this status change, it can handle it by itself or forward it to the UI application. The UI application can simply pop up the OSD to notify the user of the success or failure of the descrambling, and can also parse the additional information in the CASStatus object to show the specific reason for the failure. These additional information formats are project-related. If the JS DCAS application does not have additional information, the UI application obtains the Token in the CASStatus object and can also use this Token to communicate with the JS DCAS application through the means provided by IPC or other platforms to obtain more information.

Y.3.11.2 Method

Y.3.11.2.1 getCasToken

Prototype: number getCasToken()

Description: Returning CAS Token. If the platform forwards the CASStatus information to the UI application, the UI application can use this Token to initiate a request to the JS DCAS application to

obtain more detailed status information. The method of request is determined by the platform, such as IPC.

Parameter: None.

Return: number type, return Token.

Y.3.11.2.2 getMajorContentProblem

Prototype: number getMajorContentProblem()

Description: Returning main error value where the program cannot be watched.

Parameter: None.

Return: number type, error message.

Y.3.11.2.3 getStatusData

Prototype: ArrayBuffer getStatusData()

Description: Returning extended data in descrambling state. With extended data, the UI application can display more detailed information about the descrambling state.

Parameter: None.

Return: ArrayBuffer type, indicating extended data. If there is no extended data available, null should be returned.

Y.3.11.2.4 isSuccess

Prototype: boolean isSuccess()

Description: Returning whether descrambling is successful or not.

Parameter: None.

Return: boolean type, true-success, false-failure.

Y.3.12 JSDCAS.TeeController object

Controller for JS DCAS and TEE communication.

Y.3.12.1 Method

Y.3.12.1.1 sendCommandToTEE

Prototype: TeeRetVal
sendCommandToTEE(teeAppUUID,commandId,inputData,applicationContext)

Description: JS DCAS application uses this method to send commands to TA running in TEE.

Parameter:

teeAppUUID – Uint8Array type, UUID of TA, 16 bytes. Each CA manufacturer has a different ID.

commandId – number type, Command ID in TEE communication. Defined by each CA manufacturer.

inputData – Uint8Array type, data sent to TA.

applicationContext – Application context, platform related. It is usually provided to the application by the platform during initialization.

Return: TeeRetVal object, this object contains information such as data and errors returned from TA.

Y.3.13 JSDCAS.TeeRetVal class

This object is returned by TeeController.sendCommandToTEE. The object contains information such as data and errors returned from the TEE.

Y.3.13.1 Method

Y.3.13.1.1 getOriginCode

Prototype: number getOriginCode()

Description: Returning origin code.

Parameter: None.

Return: number type, indicating origincode.

Y.3.13.1.2 getResponseData

Prototype: Uint8Array getResponseData()

Description: Getting data returned from TA.

Parameter: None.

Return: Uint8Array type, data returned from TA. It can be null for some commands. If there is an error in the call or communication, null is also returned.

Y.3.13.1.3 getReturnCode

Prototype: number getReturnCode()

Description: Returning code.

Parameter: None.

Return: number type, indicating the return code.

Bibliography

[b-ITU-T J.205] Recommendation ITU-T J.205 (2012), *Requirements for an application control framework using integrated broadcast and broadband digital television*.

[b-ITU-T J.1033] Recommendation ITU-T J.1033 (2020), *Downloadable conditional access system for bidirectional networks – The terminal*.

[b-ITU-T J.1203] Recommendation ITU-T J.1203 (2022), *Smart television operating system – Specification*.

[b-ITU-T J.1204] Recommendation ITU-T J.1204 (2022), *Smart television operating system – Security framework.*

[b-ITU-T J.1205] Recommendation ITU-T J.1205 (2022), *Smart television operating system – Hardware abstract layer application programming interface*.

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

Published in Switzerland
Geneva, 2024