International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.764
(06/2012)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

IPTV multimedia services and applications for IPTV –
IPTV multimedia application frameworks

IPTV services enhanced script language

Recommendation ITU-T H.764

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619
Advanced multimedia services and applications	H.620–H.629
Ubiquitous sensor network applications and Internet of Things	H.640–H.649
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	
General aspects	H.700–H.719
IPTV terminal devices	H.720–H.729
IPTV middleware	H.730–H.739
IPTV application event handling	H.740–H.749
IPTV metadata	H.750–H.759
IPTV multimedia application frameworks	**H.760–H.769**
IPTV service discovery up to consumption	H.770–H.779
Digital Signage	H.780–H.789

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

Recommendation ITU-T H.764

IPTV services enhanced script language

Summary

Recommendation ITU-T H.764 describes an object-oriented programming language called "Internet protocol television services enhanced script language (IPTV SESL)", which is based on LIME-Script of Recommendation ITU-T H.762, and includes enhanced functionalities. The language is used in IPTV services for performing computations and manipulating computational objects within an IPTV terminal device environment. There are two primary implementations of this Recommendation, the "Core script profile" and the "Extended script profile". The core script profile is conformant with the LIME-Script of Recommendation ITU-T H.762 and is used for the objects for client-side computation. The extended script profile is necessary for the functional extensions required for IPTV services.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T H.764	2012-06-29	16

Keywords

IPTV SESL, multimedia application framework (MAFR), script language, web-based.

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, 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 TSB patent database at http://www.itu.int/ITU-T/ipr/.

© ITU 2013

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

List of tables

List of figures

Recommendation ITU-T H.764

IPTV services enhanced script language

1 Scope

This Recommendation describes a script language for IPTV services, Internet protocol television services enhanced script language (IPTV SESL). This language is used to provide interoperability and harmonization among IPTV multimedia application frameworks. It is foreseeable that a combination of different standard multimedia application frameworks will be used to provide standard IPTV services globally. This Recommendation describes IPTV SESL, as one of these standard multimedia application frameworks for providing the interoperable use of IPTV services. The core script profile, as well as enhanced functionalities for IPTV services, are given.

2 References

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

[ITU-T H.721]	Recommendation ITU-T H.721 (2009), *IPTV terminal devices: Basic model*.
[ITU-T H.730]	Recommendation ITU-T H.730 (2012), *Web-based terminal middleware for IPTV services*.
[ITU-T H.750]	Recommendation ITU-T H.750 (2008), *High-level specification of metadata for IPTV services.*
[ITU-T H.762]	Recommendation ITU-T H.762 (2009), *Lightweight interactive multimedia environment (LIME) for IPTV services*.
[ITU-T Y.1901]	Recommendation ITU-T Y.1901 (2009), *Requirements for the support of IPTV services*.
[IETF RFC 2326]	IETF RFC 2326 (1998), *Real time Streaming protocol (RTSP).*
[IETF RFC 2616]	IETF RFC 2616 (1999), *Hypertext Transfer Protocol – HTTP/1.1.*
[ISO/IEC 8601]	ISO/IEC 8601:2004, *Data Elements and Interchange Formats – Information Interchange – Representation of dates and times*.
[ISO/IEC 16262]	ISO/IEC 16262:2011, *Information technology – Programming languages, their environments and system software interfaces – ECMAScript language specification*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 ECMAScript [ISO/IEC 16262]: The programming language defined by ISO/IEC 16262:2011.

NOTE – Edition 5.1 of ECMAScript [b-ECMA-262] is fully aligned with [ISO/IEC 16262] (third edition).

3.1.2 electronic programme guide (EPG) [ITU-T Y.1901]: A structured set of data, intended to provide information on available content that may be accessed by end users.

NOTE – In some traditional broadcast services, EPG is defined as an on-screen guide used to display information on scheduled live broadcast television programmes, allowing a viewer to navigate, select and discover programmes by time, title, channel, genre. This traditional definition does not cover "catalogues" for on-demand and download services (sometimes called an electronic content guide (ECG) and broadband content guide (BCG) and bidirectional interactive services (sometimes called an interactive programme guide (IPG)) for end-user interaction with a server or head-end.

3.1.3 end system [ITU-T Y.1901]: A single or set of consumer devices that support IPTV services (e.g., delivery network gateway, display).

3.1.4 end user [ITU-T Y.1910]: The actual user of the products or services.

NOTE – The end user consumes the product or service. An end user can optionally be a subscriber.

3.1.5 Internet protocol television (IPTV) [ITU-T Y.1901]: Multimedia services such as television/video/audio/text/graphics/data delivered over IP-based networks that are managed to support the required level of QoS/QoE, security, interactivity and reliability.

3.1.6 IPTV terminal device [ITU-T Y.1901]: A terminal device which has ITF functionality, e.g., a STB.

3.1.7 IPTV terminal function (ITF) [ITU-T Y.1901]: The end-user function(s) associated with a) receiving and responding to network control channel messages regarding session set-up, maintenance, and teardown, and b) receiving the content of IP transport from the network and rendering.

3.1.8 linear TV [ITU-T Y.1901]: A television service in which a continuous stream flows in real time from the service provider to the terminal device and where the user cannot control the temporal order in which contents are viewed.

3.1.9 metadata [ITU-T Y.1901]: Structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment and management of the described entities.

NOTE – EPG metadata has many applications and may vary in depth from merely identifying the content package title or information to populate an EPG to providing a complete index of different scenes in a movie or providing business rules detailing how the content package may be displayed, copied or sold.

3.1.10 middleware [ITU-T Y.1901]: A layer of software between applications and resources, which consists of a set of service enablers that allow multiple functionalities running on one or more devices in an IPTV system to interact across a network.

3.1.11 network personal video recorder (nPVR) [ITU-T Y.1901]: This is the same as PVR except that the recording device is located at the service provider's premises.

3.1.12 personal video record (PVR) [ITU-T Y.1901]: An end-user controlled device that records, stores and plays back multimedia content. PVR is also known as personal digital recorder (PDR).

3.1.13 SCP [ITU-T Y.1901]: A combination of service protection and content protection.

3.1.14 terminal device (TD) [ITU-T Y.1901]: An end-user device which typically presents and/or processes the content, such as a personal computer, a computer peripheral, a mobile device, a TV set, a monitor, a VoIP terminal or an audiovisual media player.

3.1.15 time shifting [ITU-T Y.1901]: A function which allows playback of content after its initial transmission.

3.1.16 trick mode functionality [ITU-T Y.1901]: The ability to pause, rewind or forward stored content.

3.1.17 TV with trick mode [ITU-T Y.1901]: TV service with trick mode functionality.

3.1.18 user agent [b-W3C WebArch]: One type of Web agent; a piece of software acting on behalf of a person.

3.1.19 video-on-demand (VoD) [ITU-T Y.1901]: A service in which the end user can, on demand, select and view video content and where the end user can control the temporal order in which the video content is viewed (e.g., the ability to start the viewing, pause, fast forward, rewind, etc.).

NOTE – The viewing may occur sometime after the selection of the video content.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 application: A functional implementation realized as software running in one or spread over several interplaying hardware entities.

3.2.2 resource-constrained TD: An IPTV terminal device with a basic set of capabilities, such as those given in [ITU-T H.721].

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

DOM	Document Object Module
DRM	Digital Rights Management
FFW	Fast Forward
FRW	Fast Rewind
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
HTTPS	Hypertext Transport Protocol over Secure socket layer
IGMP	Internet Group Management Protocol
JSON	JavaScript Object Notation
LIME	Lightweight Interactive Multimedia Environment
NPT	Normal Play Time
PAL	Phase Alternating Line
PE	Presentation Engine
RTSP	Real Time Session Protocol
SCP	Service and Content Protection
SP	Service Provider
STB	Set-Top Box

TD	Terminal Device
TV	Television
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VoD	Video on Demand
XML	extensible Markup Language

5 Conventions

The following conventions apply to operational restrictions on attributes and properties:

R/W	The corresponding attribute or property can be read and written. An IPTV terminal device designed for basic services should support R/W of the corresponding attribute or property in the content.
R	The corresponding attribute can be read but cannot be written. An IPTV terminal device designed for basic services should support R/W of the corresponding attribute or property in the content. Write operations to this item may be ignored.
M	M indicates "Mandatory". The corresponding attribute or method must be implemented by an IPTV terminal device supporting this Recommendation.
O	O indicates "Optional". The corresponding attribute or method can either be implemented or ignored by an IPTV terminal device supporting this Recommendation.

6 Overview

The IPTV services enhanced script language (IPTV SESL) is based on LIME-Script [ITU-T H.762] with enhanced functionalities. IPTV SESL is an object-oriented programming language for performing computations and manipulating computational objects within a host environment. It was originally designed to be a web scripting language, providing a mechanism to enliven webpages in browsers and to perform server computation as part of web-based, client-server architecture. Together with the server-side scripting, it is possible to distribute computation between the client and the server, while providing a customised user interface for web-based applications.

NOTE – LIME-Script [ITU-T H.762] is a subset of ECMAScript [ISO/IEC 16262]. Only objects, properties and methods that are suitable for IPTV services are referred to in ITU-T H.762 LIME-Script.

In this Recommendation, two primary parts are defined to implement web-based IPTV services:

– The "core script profile", which is designed for resource-constrained IPTV TDs, such as the basic TD model defined in [ITU-T H.721]. This profile is fully conformant with LIME-Script of [ITU-T H.762] and comprises a subset of ECMAScript.

– The profile hereinafter called "extended script profile" is designed for enhanced IPTV TDs, such as the fully-fledged model (currently under study within ITU-T). This profile defines the functional extensions required for IPTV services. This profile is optional, but, if implemented, there are some mandatory objects, methods and properties.

IPTV SESL can be supported in many applications and can also be included as a component in many presentation engines (PE), e.g., LIME [ITU-T H.762]. Some implementations may have a completely different set of libraries; what makes applications written in one IPTV SESL dialect work does not necessarily work in another dialect.

In the IPTV TD, the user agent (e.g., a web browser) provides an IPTV SESL host environment for client-side computation, which includes objects that represent windows, menus, etc. Further, the user agent provides a means to attach scripting code to events such as page and image loading, unloading, etc. Scripting code appears within the LIME HTML of [ITU-T H.762] and the displayed page is a combination of user interface elements, together with the extended objects for IPTV services to provide web-based multimedia interaction.

7 Core script profile

ECMAScript [ISO/IEC 16262] was originally designed to be a web scripting language, providing a mechanism to enliven webpages in browsers and to perform server computation as part of web-based client-server architecture. It is suitable for being applied in web-based IPTV services to perform computations and provide interoperability with the end users within a host environment, which is the user agent (e.g., a browser) in the IPTV terminal device. It should be noted that not all of the manipulating computational objects defined by ECMAScript are necessary for implementation of the functionality defined for resource-constrained devices. Therefore, an optimized set of objects has been identified to efficiently implement that category of TDs. In this context, the core script profile has been defined.

The following clauses list the core script profile for IPTV services. All the built-in objects listed below are the objects whose semantics are defined in the LIME-Script of [ITU-T H.762]. Built-in objects must be supplied by the implementation at the start of the execution of the user agent in the IPTV terminal device.

7.1 Global

Global properties and functions can be used with built-in objects.

Properties:

- `NaN`

  The not-a-number value

Functions:

- `parseInt ( String, radix )`

  Parses a string and returns an integer

  Radix of `parseInt()` can be 8 for octal, 10 for decimal or 16 for hexadecimal (0 is interpreted as 10)

- `isNaN ( number )`

  Tests whether a value is the not-a-number value (i.e., an illegal number)

7.2 Object

LIME-Script objects provide basic language and facilities. The "Object" is one of the built-in objects.

Properties:

- `prototype`

Functions:

- `Object ( [value] )`

  Creates a new Object object

Constructor:

- `new Object ( [value] )`

  When called with new, a constructor may create an object

7.2.1 Object.prototype

Properties:

- `constructor`

  Returns the function that created the Object prototype

Functions:

- `toString()`

  Returns the string value corresponding to the [Class] internal property of the object

- `valueOf()`

  Returns the primitive value of an Object object

7.3 Function

A function is a callable object that may be invoked as a subroutine.

Properties:

- `prototype`

  Value is used to initialize the prototype internal property

- `length`

  Indicates the number of arguments expected by the function

7.3.1 Function.prototype

Properties:

- `constructor`

  It returns the function that created the prototype (in this case Function)

Functions:

- `toString()`

  It returns a representation of the function

7.4 Array

The Array object is used to store multiple values in a single variable.

Properties:

- `prototype`

  Allows the adding of properties and methods to an object

- `length`

  Sets or returns the number of elements in an array

Functions:

- `Array ( item0, item1, … )`

  Creates a new array

- `new Array ( item0, item1,… )`

  Creates and initializes an array

- `new Array ( [len] )`

  Creates and initializes an array with length set to len

7.4.1 Array.prototype

Properties:

- `constructor`

  Returns the function that creates the Array object prototype

Functions:

- `toString()`

  Converts an array to a String, and returns the result
- `join ( [separator] )`

  Joins all elements of an array into a String
- `reverse()`

  Reverses the order of the elements in an array
- `sort ( [comparefn] )`

  Sorts the elements of an array

7.5 String

A String is used to manipulate a stored piece of text.

Properties:

- `prototype`

  Allows properties and methods to be added to an object
- `length`

  Returns the length of a String

Functions:

- `String ( [value] )`

  When called as a function it performs a type conversion

Constructor:

- `new String ( [value] )`

  Used to create a String object
- `String.fromCharCode ( char0[ ,char1, …] )`

  Returns a String value containing as many characters as arguments

7.5.1 String.prototype

Properties:

- `constructor`

  Returns the function that creates the String object prototype

Functions:

- `toString()`

  Returns this String value
- `valueOf()`

  Returns the primitive value of a String object
- `charAt ( pos )`

  Returns the character at the specified index

- `charCodeAt ( pos )`

  Returns the unicode of the character at the specified index
- `indexOf ( searchString, position )`

  Returns the position of the first found occurrence of a specified value in a String
- `lastIndexOf ( searchString, position )`

  Returns the position of the last found occurrence of a specified value in a String
- `split( separator )`

  Splits a String into an array of substrings
- `substring ( start[,end] )`

  Extracts the characters from a String, between two specified indices
- `toLowerCase()`

  Converts a String to lowercase
- `toUpperCase()`

  Converts a String to uppercase

7.6 Boolean

Boolean is used to convert a non-Boolean value to a Boolean value.

Properties:

- `prototype`

  Allows adding properties and methods to an object

Functions:

- `Boolean ( [value] )`

  Returns a Boolean value

Constructor:

- `new Boolean ( [value] )`

  Used to create a Boolean object with an initial value

7.6.1 Boolean.prototype

Properties:

- `constructor`

  Returns the function that creates the Boolean object prototype

Functions:

- `toString()`

  Converts a Boolean value to a String, and returns the result
- `valueOf()`

  Returns the primitive value of a Boolean object

7.7 Number

Object wrapper for primitive numeric values.

Properties:

- `prototype`

Allows you to add properties and methods to an object

- `MAX_VALUE`

  Returns the largest number possible
- `MIN_VALUE`

  Returns the smallest number possible
- `NaN`

  The not-a-number value

Functions:

- `Number ( [value] )`

  Function used to create a Number object

Constructor:

- `new Number ( [value] )`

  Used to create Number objects

7.7.1 Number.prototype

- `constructor`

  Returns the function that creates the Number object prototype

Functions:

- `toString ( [radix] )`

  Converts a Number object to a String
- `valueOf()`

  Returns the primitive value of a Number object

7.8 Date

Date is used to work with dates and time.

Properties:

- `prototype`

  Allows properties and methods to be added to an object

Functions:

- `Date ( [year, month [, date [, hours [, minutes [, seconds [, ms]]]]]] )`

Constructor:

- `new Date ( [year, month [, date [, hours [, minutes [, seconds [, ms]]]]]] )`

  Used to create Date objects

7.8.1 Date.prototype

Properties:

- `constructor`

  Returns the function that created the Date object prototype

Functions:

- `toString()`

  Converts a Date object to a String

- `getFullYear()`

  Returns the year (four digits)
- `getUTCFullYear()`

  Returns the year, according to universal time (four digits)
- `getMonth()`

  Returns the month (from 0-11)
- `getUTCMonth()`

  Returns the month, according to universal time (from 0-11)
- `getDate()`

  Returns the day of the month (from 1-31)
- `getUTCDate()`

  Returns the day of the month, according to universal time (from 1-31)
- `getDay()`

  Returns the day of the week (from 0-6)
- `getUTCDay()`

  Returns the day of the week, according to universal time (from 0-6)
- `getHours()`

  Returns the hour (from 0-23)
- `getUTCHours()`

  Returns the hour, according to universal time (from 0-23)
- `getMinutes()`

  Returns the minutes (from 0-59)
- `getUTCMinutes()`

  Returns the minutes, according to universal time (from 0-59)
- `getSeconds()`

  Returns the seconds (from 0-59)
- `getUTCSeconds()`

  Returns the seconds, according to universal time (from 0-59)
- `getMilliseconds()`

  Returns the milliseconds (from 0-999)
- `getUTCMilliseconds()`

  Returns the milliseconds, according to universal time (from 0-999)
- `getTimezoneOffset()`

  Returns the time difference between GMT and local time, in minutes
- `setMilliseconds ( ms )`

  Sets the milliseconds (from 0-999)
- `setUTCMilliseconds ( ms )`

  Sets the milliseconds, according to universal time (from 0-999)
- `setSeconds ( sec, [, ms] )`

  Sets the seconds (from 0-59)

- `setUTCSeconds ( sec, [, ms] )`

  Set the seconds, according to universal time (from 0-59)
- `setMinutes ( min [, sec, [, ms]] )`

  Set the minutes (from 0-59)
- `setUTCMinutes ( min [, sec, [, ms]] )`

  Set the minutes, according to universal time (from 0-59)
- `setHours ( hour [, min [, sec, [, ms]]] )`

  Sets the hour (from 0-23)
- `setUTCHours ( hour [, min [, sec, [, ms]]] )`

  Sets the hour, according to universal time (from 0-23)
- `setDate(date)`

  Sets the day of the month (from 1-31)
- `setMonth ( mon [, date] )`

  Sets the month (from 0-11)
- `setUTCMonth ( mon [, date] )`

  Sets the month, according to universal time (from 0-11)
- `setFullYear ( year [, mon [, date]] )`

  Sets the year (four digits)
- `setUTCFullYear ( year [, mon [, date]] )`

  Sets the year, according to universal time (four digits)
- `setYear ( year )`

  Use the setFullYear() method instead
- `toLocaleString()`

  Converts a Date object to a String object using locale conventions

  Produces the same output format as `Date.prototype.toString()`
- `toUTCString()`

  Converts a Date object to a String, according to universal time

  Produces the same output format as `Date.prototype.toString()`

8 Extended script profile

The core script profile for IPTV provides the basic native built-in objects implemented by resource-restrained IPTV TDs. Since web-based IPTV also requires video controlling and user interactivity, an extension is needed to the core script profile for IPTV. The extended script profile defined in this Recommendation provides for these additional manipulating computational objects for video and interactivity-related performing computations, which may be optionally implemented by IPTV TDs.

The procedure on how the core script profile and extended script profile coordinate with the web-based terminal middleware to implement interactivity amongst the end user, terminal device and IPTV service platform is given in [ITU-T H.730].

8.1 MediaController object

The MediaController object encapsulates the capabilities of the terminal device to play live channel, nPVR, VoD, music and other media types. The MediaController object can provide necessary methods and properties to control a media.

The essential behaviour of playing for live channel, nPVR, VoD, music and other media types are the same as for those which can be abstracted to the methods and properties of MediaController, including the status of play, trick mode, stop, etc.

The programmed MediaController object in the web document not only carries the necessary information of the current media with it, but also provides methods for user interactivity such as trick mode control and web document refreshment.

8.1.1 Media control

8.1.1.1 Properties

Since the MediaController object provides the ability to control and display the media for IPTV, the identifier, status and other properties of the player are listed in the following clauses.

8.1.1.1.1 allowTrickmode

The property controls trick mode (fast forward/fast rewind/pause) operation during the life cycle of the player. The player property is logically "AND" with the media "`playmode`" property.

- Optionality: Mandatory
- DataType: `unsignedInt[0:1]`
- Reference value:
  - `0`: TRICK mode disabled
  - `1 (default)`: TRICK mode enabled
- Read-write: R/W

8.1.1.1.2 currentPlayTime

The current playing time of the media. For VoD, the time represents the relative time (NPT) since the start point of the media. For nPVR, it represents the current absolute time (ClockTime). For channel, it is meaningless and an empty string will be returned.

- Optionality: Mandatory
- DataType: `String(128)`
- Reference value:
  - Empty string
- Read-write: R

NOTE – NormalPlayTime(NPT) and AbsoluteTime(ClockTime) refer to [IETF RFC 2326].

8.1.1.1.3 channelNum

The number of the current channel.

- Optionality: Mandatory
- DataType: `int[-1:4294967295]`
- Reference value:
  - If the current media is not a channel, the property is set to "–1"
- Read-write: R

8.1.1.1.4 cycleFlag

The property controls media loop play.

- Optionality: Mandatory
- DataType: `unsignedInt[0:1]`

- Reference value:
  - `0`: loop mode
  - `1 (default)`: Single play mode
- Read-write: R/W

8.1.1.1.5 height

The height of the video output (in pixels). Each time the property is changed, the video output remains unchanged until the "`refreshVideoDisplay`" method is called.

- Optionality: Mandatory
- DataType: `int[4]`
- Reference value:
  - The default value is set to the height of the browser window (e.g., for PAL the height is 576 pixels)
- Read-write: R/W

8.1.1.1.6 instanceId

The unique ID of the local media player instance created.

- Optionality: Mandatory
- DataType: `unsignedInt[0:255]`
- Reference value:
  - The default value is `0` indication no MediaController object instance has been initialized
- Read-write: R/W

8.1.1.1.7 left

The relative excursion from the left-top corner of the browser window (in pixels). Each time the property is changed, the video output remains unchanged until the "`refreshVideoDisplay`" method is called.

- Optionality: Mandatory
- DataType: `int[4]`
- Reference value:
  - default value is ZERO
- Read-write: R/W

8.1.1.1.8 playMode

The play mode of the MediaController. The property has to be defined since the initiation of the player. It cannot be changed during play until the MediaController is stopped.

- Optionality: Mandatory
- DataType: `unsignedInt[0:1]`
- Reference value:
  - `0 (default)`: single media mode
  - `1`: playlist mode
- Read-write: R/W

8.1.1.1.9 width

The width of the video output (in pixels). Each time the property is changed, the video output remains unchanged until the `refreshVideoDisplay` (see clause 8.1.1.2.15) method is called.

- Optionality: Mandatory
- DataType: `int[4]`
- Reference value:
  - The default value is set to the width of the browser window (e.g., for PAL, the width is 720 pixels)
- Read-write: R/W

8.1.1.1.10 mute

Set the mute/sound status of the MediaController. It works as soon as the property is set.

- Optionality: Mandatory
- DataType: `unsignedInt[0:1]`
- Reference value:
  - `0 (default)`: sound
  - `1`: mute
- Read-write: R/W

8.1.1.1.11 mediaCode

The unique identifier of the current media played by the MediaController instance. If the "`playMode`" property of the MediaController is set to playlist mode, the property represents the "`mediaCode`" of the current played media.

- Optionality: Mandatory
- DataType: `String`
- Reference value:
  - None
- Read-write: R

8.1.1.1.12 mediaDuration

The duration (in seconds) of the current media.

- Optionality: Mandatory
- DataType: `unsignedInt`
- Reference value:
  - `0`
- Read-write: R

8.1.1.1.13 playbackMode

The player functional block should notify the web document by Event object, whenever the status of the player changes. This property is encapsulated into a formatted string like JSON or XML, which contains the following 2 parameters.

- Optionality: Mandatory
- Current Mode: NormalPlay, Pause, Trickmode.
- Speed: `0x`, `1x`, `2x/-2x`, `4x/-4x`, `8x/-8x`, `16x/-16x`, `32x/-32x`, etc.

- DataType: String(128)
- Reference value:
  - `0`
- Read-write: R

8.1.1.1.14 top

The relative excursion from the left-top corner of the browser window (in pixels). Each time the property is changed, the video output remains unchanged until the `refreshVideoDisplay` method is called.

- Optionality: Mandatory
- DataType: `int[4]`
- Reference value:
  - `0`
- Read-write: R/W

8.1.1.1.15 videoAlpha

It represents the transparency of the video.

- Optionality: Optional
- DataType: `unsignedInt[0:100]`
- Reference value:
  - `0 (default)`: Opaque
  - `1`: Transparent
- Read-write: R/W

8.1.1.1.16 videoDisplayMode

The video display mode of the MediaController. Each time the `videoDisplayMode` is changed, the video output remains unchanged until the `refreshVideoDisplay` method is called.

- Optionality: Mandatory
- DataType: `unsignedInt[0:3,255]`
- Reference value:
  - `0`: The video should be displayed according to the "`left`", "`top`", "`width`", "`left`" properties.
  - `1 (default)`: Full screen mode.
  - `2`: The video should be displayed according to the height of the screen without changing the height video aspect ratio.
  - `3`: The video should be displayed according to the width of the screen without changing the width video aspect ratio.
  - `255`: The video output should be hidden while the audio is still playing.
- Read-write: R/W

8.1.1.2 Methods

8.1.1.2.1 Constructor

The constructor of the MediaController object with a default value.

- Optionality: Mandatory

- Syntax:

```
MediaController()
```

- Parameters:
  - None
- Return value:
  - None

8.1.1.2.2 bindNativePlayerInstance

The method is called to create and maintain a native player instance, which identifies a certain media stream or player in the STB. Once a new player is created by the constructor and bound with an existing `instanceId`, the player will take full charge of the player instance in the STB.

- Optionality: Mandatory
- Syntax:

```
number bindNativePlayerInstance ( input number instanceID )
```

- Parameters:
  - `instanceID`: Refer to "`instanceId`".
- Return value:
  - `0`: success
  - `–1`: failure
  - Others: undefined

NOTE – If the newly created player cannot match a local "`instanceId`", the operation will be reported with failure by returning –1.

8.1.1.2.3 fastForward

Media fast forwarding with the rate indicated by "`speed`".

- Optionality: Mandatory
- Syntax:

```
fastForward ( input number speed )
```

- Parameters:
  - `speed ( int )`: valid values are positive integers, e.g., 64, 32, 16, 8, 4, 2, etc.
- Return value:
  - None

8.1.1.2.4 fastRewind

Media fast rewinding with the rate indicated by "speed".

- Optionality: Mandatory
- Syntax:

```
fastRewind ( input number speed )
```

- Parameters:
  - `speed ( int )`: valid values are negative integers, e.g., –64, –32, –16, –8, –4, –2, etc.
- Return value:
  - None

8.1.1.2.5 getVolume

Get the current volume of the STB.

- Optionality: Mandatory
- Syntax:
  ```
  number getVolume()
  ```
- Parameters:
  - None
- Return value:
  - Current volume of the STB (`unsignedInt[0:100]`)

8.1.1.2.6 gotoEnd

Jump to the end of the media.

- Optionality: Mandatory
- Syntax:
  ```
  gotoEnd()
  ```
- Parameters:
  - None
- Return value:
  - None

8.1.1.2.7 gotoStart

Jump to the beginning of the media.

- Optionality: Mandatory
- Syntax:
  ```
  gotoStart()
  ```
- Parameters:
  - None
- Return value:
  - None

8.1.1.2.8 initMediaController

Initialize the property of MediaController. This method would not be called until the MediaController is created and binds an "`instanceId`", which is generated by another MediaController object in the previous web documents.

- Optionality: Mandatory
- Syntax:
  ```
  number initMediaController ( input number instanceID,
     input number playlistFlag,
     input number videoDisplayMode,
     input number height,
     input number width,
     input number left,
     input number top,
     input number muteFlag,
     input number useNativeUIFlag,
     input number subtitleFlag,
  ```

```
   input number videoAlpha,
   input number cycleFlag)
```

- Parameters:
  - Refer to related MediaController properties
- Return value:
  - `0`: success
  - `-1`: failure
  - Others: undefined

8.1.1.2.9 joinChannel

Join in a channel and return the result immediately. The channel could be:

– a multicast channel using IGMP and providing RTSP based nPVR

– a unicast channel using RTSP for linear TV with trick mode and nPVR

– a webChannel, which provides EPG browsing with embedded video.

- Optionality: Mandatory
- Syntax:

```
number joinChannel ( input number userchannelid )
```

- Parameters:
  - `userchannelid ( unsignedInt[0: 4294967295] )`
- Return value:
  - `0`: success
  - `-1`: failure
  - Others: undefined

NOTE – Before joining a channel, the "leaveChannel" method should be called first to quit the previously joined channel.

8.1.1.2.10 launchIPTVContent

To launch an IPTV content identified by "`content_uri`" from the temporal position of the start of the viewing identified by "`start_npt`" with the licence identifier "`license_id`", which is optional. When the video is finished, the document to be displayed should be identified by "`ret_uri`".

- Optionality: Mandatory
- Syntax:

```
number launchIPTVContent ( input String content_uri,
   input String ret_uri,
   input number start_npt[,
   input number license_id])
```

- Parameters:
  - `content_uri ( String )`: URI of the content to be played.
  - `ret_uri ( String )`: the URI of the document to be displayed when the video is finished.
  - `start_npt ( unsignedInt )`: the temporal position of the start of the viewing (the temporal position from the beginning of the content).
  - `license_id ( String )`: the licence ID used for the viewing of the content.
- Return value:
  - `1`: success

– `NaN`: failure

8.1.1.2.11 leaveChannel

Quit the current channel.

- Optionality: Mandatory
- Syntax:

```
number leaveChannel()
```

- Parameters:
  – None.
- Return value:
  – `0`: success
  – `-1`: failure

NOTE – This method can only be called to quit the channels joined by the "`joinChannel`" method.

8.1.1.2.12 pause

Pause the media which is playing.

- Optionality: Mandatory
- Syntax:

```
pause()
```

- Parameters:
  – None
- Return value:
  – None

8.1.1.2.13 playFromStart

Play from the beginning of the media. This method works only when the `playMode` property is set to 0.

- Optionality: Mandatory
- Syntax:

```
number playFromStart()
```

- Parameters:
  – None
- Return value:
  – `0`: success
  – `-1`: failure

8.1.1.2.14 playByTime

Play from a certain time of the current media (the selected media for the playlist). This method is invalid for linear TV with trick mode but valid for linear TV with trick mode in trick mode.

- Optionality: Mandatory
- Syntax:

```
number playByTime ( input number type,
   input String timestamp[,
   input number speed])
```

- Parameters:
  - `type (unsignedInt[1:2])`: 1: NPT, 2: Clock Time
  - `timestamp (String)`: Refer to `NormalPlayTime(NPT)` and `AbsoluteTime (ClockTime)` in [IETF RFC 2326].
  - `speed`: play rate
- Return value:
  - `0`: success
  - `-1`: failure

NOTE – For VoD, the time represents the relative time (NPT) since the start point of media. For nPVR, it represent the current absolute time (ClockTime).

8.1.1.2.15 refreshVideoDisplay

Adjust the display of the video according to the property of the "`videoDisplayMode`", "`left`", "`top`", "`width `", "`left`" properties.

- Optionality: Mandatory
- Syntax:

```
refreshVideoDisplay()
```

- Parameters:
  - None
- Return value:
  - None

8.1.1.2.16 releaseMediaController

Release the resources occupied by the player which is identified by the "`instanceId`".

- Optionality: Mandatory
- Syntax:

```
number releaseMediaController (input number instanceID)
```

- Parameters:
  - `instanceID`: Refer to "`instanceId `".
- Return value:
  - `0`: success
  - `-1`: failure
  - Others: undefined

8.1.1.2.17 resume

Resume from the pause/forward/rewind mode of the current media.

- Optionality: Mandatory
- Syntax:

```
resume()
```

- Parameters:
  - None
- Return value:
  - None

8.1.1.2.18 setSingleMedia

Locate the resource of a single media.

- Optionality: Mandatory
- Syntax:
  `setSingleMedia ( input anySimpleType mediaStr )`
- Parameters:
  - `mediaStr`: formatted string containing content metadata. Refer to Annex B on Content metadata
- Return value:
  - None

8.1.1.2.19 setVolume

Set the volume of the STB.

- Optionality: Mandatory
- Syntax:
  `setVolume ( input number volume )`
- Parameters:
  - `volume ( unsignedInt[0:100] )`: 0 means mute and 100 means maximum.
- Return value:
  - None

8.1.1.2.20 stop

Stop the current media.

- Optionality: Mandatory
- Syntax:
  `stop()`
- Parameters:
  - None
- Return value:
  - None

8.1.2 Graphic user interface

8.1.2.1 Properties

8.1.2.1.1 audioVolumeUI

Enable or disable the display of the player's local volume reminder. Also, the property should comply with the "`nativeUIFlag`" property. It works for as long as the property is set.

- Optionality: Optional
- DataType: `unsignedInt[0:1]`
- Reference value:
  - `0`: Disable the player's local UI of the volume reminder.
  - `1 (default)`: Enable the player's local UI of the volume reminder.
- Read-write: R/W

8.1.2.1.2 audioTrackUI

Enable or disable the display of the player's local audio track selection reminder. Also, the property should comply with the "`nativeUIFlag`" property. It works for as long as the property is set.

- Optionality: Optional
- DataType: `unsignedInt[0:1]`
- Reference value:
  - `0`: Disable the player's local UI of the audio track selection reminder.
  - `1 (default)`: Enable the player's local UI of the audio track selection reminder.
- Read-write: R/W

8.1.2.1.3 channelNoUI

Enable or disable the display of the player's local channel number reminder. Also, the property should comply with the "`nativeUIFlag`" property. It works for as long as the property is set.

- Optionality: Optional
- DataType: `unsignedInt[0:1]`
  - `0`: Disable the player's local UI of the channel number reminder.
  - `1 (default)`: Enable the player's local UI of the channel number reminder.
- Read-write: R/W

8.1.2.1.4 muteUI

Enable or disable the display of the player's local mute reminder. Also, the property should comply with the "`nativeUIFlag`" property. It works for as long as the property is set.

- Optionality: Optional
- DataType: `unsignedInt[0:1]`
- Reference value:
  - `0`: Disable the player's local UI of the mute reminder.
  - `1 (default)`: Enable the player's local UI of the mute reminder.
- Read-write: R/W

8.1.2.1.5 nativeUIFlag

Enable or disable the display of the local UI, e.g., the process bar, volume, channel number, etc. It works for as long as the property is set.

- Optionality: Optional
- DataType: `unsignedInt[0:1]`
- Reference value:
  - `0`: Disable the local UI display of the player.
  - `1 (default)`: Enable the local UI display of the player.
- Read-write: R/W

8.1.2.1.6 progressBarUI

Enable or disable the display of the player's local process bar. Also, the property should comply with the "`nativeUIFlag`" property. It works for as long as the property is set.

- Optionality: Optional
- DataType: `unsignedInt[0:1]`

- Reference value:
  - `0`: Disable the player's local UI of the process bar.
  - `1 (default)`: Enable the player's local UI of the process.
- Read-write: R/W

8.1.2.2 Methods

No methods are provided for the graphic user interface.

8.1.3 Audio and subtitles

8.1.3.1 Properties

8.1.3.1.1 audioTrack

The audio track of the current media.

- Optionality: Optional
- DataType: `String`
- Reference value:
  - Possible values are "`English`", "`Spanish`", "`Chinese`", "`Japanese`", "`Korean`", etc.
- Read-write: R

8.1.3.1.2 currentAudioChannel

The audio channel of the current media.

- Optionality: Optional
- DataType: `String`
- Reference value:
  - Possible values are "`LEFT`", "`RIGHT`","`STEREO`","`JOINTSTEREO`", etc.
- Read-write: R

8.1.3.1.3 subtitle

Disable/enable the display of subtitles. It works as soon as the property is set.

- Optionality: Optional
- DataType: `unsignedInt[0:1]`
- Reference value:
  - `0`: Subtitles disabled
  - `1 (default)`: Subtitles enabled
- Read-write: R/W

8.1.3.1.4 subtitleType

The subtitle type of the current media.

- Optionality: Optional
- DataType: `String`
- Reference value:
  - Possible values are "`English`", "`Spanish`", "`Chinese`", "`Japanese`", "`Korean`", etc.
- Read-write: R

8.1.3.2 Methods

8.1.3.2.1 switchAudioChannel

Switch the audio channel in rotation.

- Optionality: Optional
- Syntax:
  `switchAudioChannel()`
- Parameters:
  – None
- Return value:
  – None

8.1.3.2.2 switchAudioTrack

Switch the audio track in rotation.

- Optionality: Optional
- Syntax:
  `switchAudioTrack()`
- Parameters:
  – None
- Return value:
  – None

8.1.3.2.3 switchSubtitle

Switch the subtitles in rotation.

- Optionality: Optional
- Syntax:
  `switchSubtitle()`
- Parameters:
  – None
- Return value:
  – None

8.1.4 Playlist mode

8.1.4.1 Properties

8.4.1.1.1 playlist

The playlist of the media player instance, encapsulated in a formatted string like JSON or XML. The property is only available when the "`playMode`" is set to 1.

- Optionality: Optional
- DataType: `String`
- Reference value:
  – None
- Read-write: R

NOTE – Refer to content metadata in Annex B.

8.1.4.1.2 playlistCount

The amount of media in the playlist. The property is only available when the "`playMode`" is set to 1.

- Optionality: Optional
- DataType: `unsignedInt`
- Reference value:
  - If the "`playMode`" is set to 0, this property is always "1".
- Read-write: R

8.1.4.1.3 playlistEntryID

The unique ID of the media in the playlist, for fast access. The property is only available when the "`playMode`" is set to 1.

- Optionality: Optional
- DataType: `String`
- Reference value:
  - None
- Read-write: R

8.1.4.1.4 playlistIndex

The index of the current media in the playlist. The index starts from 0, which indicates the top of the playlist. The property is only available when the "`playMode`" is set to 1.

- Optionality: Optional
- DataType: `unsignedInt`
- Reference value:
  - None
- Read-write: R

8.1.4.2 Methods

8.1.4.2.1 addMedia

Add one or several media to the current playlist.

- Optionality: Optional
- Syntax:

```
addMedia ( input number index,
   input String mediaStr)
```

- Parameters:
  - `Index`: the position that the media is added to in the current playlist starting from 0.
  - `mediaStr`: the description of the media. Refer to Annex B on Content metadata.
- Return value:
  - None

8.1.4.2.2 clearAllMedia

Clear all the media in the current playlist.

- Optionality: Optional
- Syntax:

```
clearAllMedia()
```

- Parameters:
  – None
- Return value:
  – None

8.1.4.2.3 getCurrentMediaIndex

Get the index of the current playing media in the playlist.

- Optionality: Optional
- Syntax:

  `number getCurrentMediaIndex()`
- Parameters:
  – None
- Return value:
  – Index of the current playing media in the playlist; 0 indicates the first media.

8.1.4.2.4 getCurrentPlaylistEntryID

Get the "`PlaylistEntryID`" of the media currently playing in the playlist.

- Optionality: Optional
- Syntax:

  `String getCurrentPlaylistEntryID()`
- Parameters:
  – None
- Return value:
  – the "`PlaylistEntryID`" of the current playing media in the playlist.

8.1.4.2.5 getMediaCount

Get the number of media in the current playlist

- Optionality: Optional
- Syntax:

  `number getMediaCount()`
- Parameters:
  – None
- Return value:
  – The number of media in the current playlist.

8.1.4.2.6 getPlaylist

Get current playlist.

- Optionality: Optional
- Syntax:

  `String getPlaylist()`
- Parameters:
  – None
- Return value:
  – The current playlist.

8.1.4.2.7 moveMediaByIndex

Move one media identified by the "`playlistEntryID`" to a specific position in the current playlist.

- Optionality: Optional
- Syntax:

```
moveMediaByIndex ( input String playlistEntryID,
   input number toIndex )
```

- Parameters:
  - `playlistEntryID`: Refer to the "`playlistEntryID`" property.
  - `toIndex`: The destination position in the playlist starting from 0.
- Return value:
  - None

8.1.4.2.8 moveMediaByOffset

Move one media identified by the "`playlistEntryID`" by `offset` in the current playlist.

- Optionality: Optional
- Syntax:

```
moveMediaByOffset ( input String playlistEntryID,
   input number offset)
```

- Parameters:
  - `playlistEntryID`: Refer to the "`playlistEntryID`" property.
  - `offset`: The offset count from the position of the current media. A negative number indicates moving to the front of the media and a positive number indicates moving to the back of the media.
- Return value:
  - None

8.1.4.2.9 moveMediaByIndex1

Move the media from one position to another in the current playlist.

- Optionality: Optional
- Syntax:

```
moveMediaByIndex1 ( input number index,
   input number toIndex)
```

- Parameters:
  - `index`: The position of the media in the current playlist.
  - `toIndex`: The destination position in the playlist.
- Return value:
  - None

8.1.4.2.10 moveMediaByOffset1

Move the media in a specific position by `offset` in the current playlist.

- Optionality: Optional
- Syntax:

```
moveMediaByOffset1 ( input number index,
   input number offset )
```

- Parameters:

  - `index`: The position of the media in the current playlist.
  - `offset`: The offset count from the position of the current media. A negative number indicates moving to the front of the media and a positive number indicates moving to the back of the media.
- Return value:
  - None.

8.1.4.2.11 moveMediaToNext

Move one media identified by the "`playlistEntryID`" to the next position in the playlist.

- Optionality: Optional
- Syntax:

```
moveMediaToNext ( input String playlistEntryID )
```

- Parameters:
  - `playlistEntryID`: Refer to the "`playlistEntryID`" property.
- Return value:
  - None.

8.1.4.2.12 moveMediaToPrevious

Move one media identified by the "`playlistEntryID`" to the previous position in the playlist.

- Optionality: Optional
- Syntax:

```
moveMediaToPrevious ( input String playlistEntryID )
```

- Parameters:
  - `playlistEntryID`: Refer to the "`playlistEntryID`" property.
- Return value:
  - None

8.1.4.2.13 moveMediaToTop

Move one media identified by the "`playlistEntryID`" to the top in the playlist.

- Optionality: Optional
- Syntax:

```
moveMediaToTop ( input String playlistEntryID )
```

- Parameters:
  - `playlistEntryID`: Refer to the "`playlistEntryID`" property.
- Return value:
  - None

8.1.4.2.14 moveMediaToBottom

Move one media identified by the "`playlistEntryID`" to the bottom of the playlist.

- Optionality: Optional
- Syntax:

```
moveMediaToBottom ( input String playlistEntryID )
```

- Parameters:
  - `playlistEntryID`: Refer to the `"playlistEntryID"` property.

- Return value:
  - None

8.1.4.2.15 moveMediaToNext1

Move the media from the current position to the next.

- Optionality: Optional
- Syntax:

```
moveMediaToNext1 ( input number index )
```

- Parameters:
  - `index`: The position of the media in the current playlist.
- Return value:
  - None

8.1.4.2.16 moveMediaToPrevious1

Move the media from the current position to the previous one.

- Optionality: Optional
- Syntax:

```
moveMediaToPrevious1 ( input number index )
```

- Parameters:
  - `index`: The position of the media in the current playlist.
- Return value:
  - None.

8.1.4.2.17 moveMediaToTop1

Move the media from the current position to the top of the playlist.

- Optionality: Optional
- Syntax:

```
moveMediaToTop1 ( input number index )
```

- Parameters:
  - `index`: The position of the media in the current playlist.
- Return value:
  - None

8.1.4.2.18 moveMediaToBottom1

To move the media from the current position to the bottom of the playlist.

- Optionality: Optional
- Syntax:

```
moveMediaToBottom1 ( input number index )
```

- Parameters:
  - `index`: The position of the media in the current playlist.
- Return value:
  - None

8.1.4.2.19 removeCurrent

Remove the currently selected media in the playlist. Once this method is called, the current media that is being played will stop and the media following the removed media becomes the current selection, but the media will not be played until the media control-related methods are called.

- Optionality: Optional
- Syntax:

```
removeCurrent()
```

- Parameters:
  - None
- Return value:
  - `0:` success
  - `-1:` failure
  - Others: undefined

8.1.4.2.20 removeMediaByIndex

Remove one or more media by its or their index in the playlist.

- Optionality: Optional
- Syntax:

```
removeMediaByIndex ( input number Array index )
```

- Parameters:
  - `index`: The array of the position of the media in the current playlist.
- Return value:
  - `0:` success
  - `-1:` failure
  - Others: undefined

8.1.4.2.21 removeMediaByPlaylistEntryID

Remove one or more media identified by "`playlistEntryID`" in the playlist.

- Optionality: Optional
- Syntax:

```
removeMediaByPlaylistEntryID ( input String Array
   playlistEntryID )
```

- Parameters:
  - `playlistEntryID`: The string array of the media identified by the `playlistEntryID` property in the current playlist.
- Return value:
  - `0:` success
  - `-1:` failure
  - Others: undefined

8.1.4.2.22 removeNext

Remove the next media of the current selected/playing media in the playlist.

- Optionality: Optional
- Syntax:

```
removeNext()
```

- Parameters:
  - None.
- Return value:
  - `0:` success
  - `-1:` failure
  - Others: undefined

8.1.4.2.23 removePrevious

Remove the previous media of the current selected/playing media in the playlist.

- Optionality: Optional
- Syntax:

```
removePrevious()
```

- Parameters:
  - None
- Return value:
  - `0:` success
  - `-1:` failure
  - Others: undefined

8.1.4.2.24 selectMediaByIndex

Select one media by its index in the playlist. The media will not be played until the media control-related methods are called.

- Optionality: Optional
- Syntax:

```
selectMediaByIndex ( input number index )
```

- Parameters:
  - `index`: The position of the media in the current playlist.
- Return value:
  - None

8.1.4.2.25 selectMediaByOffset

Select one media according to its given offset from the current media that has been selected or playing in the playlist. The media will not be played until the media control-related methods are called.

- Optionality: Optional
- Syntax:

```
selectMediaByOffset ( input number offset )
```

- Parameters:
  - `offset`: The offset count from the position of the current media. A negative number indicates moving to the front of the media and a positive number indicates moving to the back of the media.
- Return value:
  - None

8.1.4.2.26 selectNext

Select the next media of the current selected/playing media in the playlist. The media will not be played until the media control-related methods are called.

- Optionality: Optional
- Syntax:

```
selectNext()
```

- Parameters:
  - None
- Return value:
  - None

8.1.4.2.27 selectPrevious

Select the previous media of the current selected/playing media in the playlist. The media will not be played until the media control-related methods are called.

- Optionality: Optional
- Syntax:

```
selectPrevious()
```

- Parameters:
  - None
- Return value:
  - None

8.1.4.2.28 selectTop

Select the first media in the playlist. The media will not be played until the methods related to media control are called.

- Optionality: Optional
- Syntax:

```
selectTop()
```

- Parameters:
  - None
- Return value:
  - None

8.1.4.2.29 selectBottom

Select the last media in the playlist. The media will not be played until the media control-related methods are called.

- Optionality: Optional
- Syntax:

```
selectBottom()
```

- Parameters:
  - None
- Return value:
  - None

8.1.4.2.30 selectMediaByPlaylistEntryID

Select the media identified by "`playlistEntryID`" in the playlist. The media will not be played until the media control-related methods are called.

- Optionality: Optional
- Syntax:

```
SelectMediaByPlaylistEntryID ( input String playlistEntryID )
```

- Parameters:
  - `playlistEntryID`: Refer to the `playlistEntryID` property.
- Return value:
  - None.

8.2 Event object

The Event object is an object designed for event notification from the service component layer of the IPTV TD to the web-based user agent. Events are caused by user interaction or remote server notification. Once an event happens, an event notification is generated by the service component layer and handed to the user agent in the presentation engine layer. Virtual key operation events are encapsulated in an Event object; they will be generated by the user agent and executed according to the service logic preprogrammed in the web document.

NOTE – See [ITU-T H.730] for the definition of the service component layer and presentation engine layer.

The types of events include channel change, play mode change, system error and so on. With the combination of different types of Event objects, the service provider who generated the web document, will easily handle the behaviour in the IPTV TD caused by the service logic preprogrammed in the web documents to provide the corresponding service experience.

8.2.1 General data structure

An event object is a data structure which contains the basic information of the event generated by the service component layer and provides a formatted structure with "anySimpleType" for the user agent's script execution.

In this Recommendation, the Event object mainly concerns the media-related event. Table 8-1 shows the data structure of the Event object.

Table 8-1 – Data structure

Properties	DataType	Instructions
`type`	String(64)	The types of event being referred to in Table 8-2
`instance_id`	unsignedInt[0:255]	The native MediaController instance ID of the STB
Others, for future use	anySimpleType	Extended properties according to different statuses

8.2.2 Event types

According to the different values of the "type" property in the Event object of Table 8-1, different data structures are defined. Table 8-2 gives the definitions of the Event types, with further details being in the following clauses.

Table 8-2 – Event types

EVENT_TYPE	Instructions
`EVENT_GO_CHANNEL`	Event triggered when the CH+, CH- or the number keys are pressed, the STB open the new channel.
`EVENT_MEDIA_END`	Event triggered when the media in the MediaController is played to the end.
`EVENT_MEDIA_BEGINNING`	Event triggered when the media in the MediaController is played from the start.
`EVENT_MEDIA_ERROR`	Event triggered when an error happens in the MediaController.
`EVENT_PLAYMODE_CHANGE`	Event triggered when the playback mode of the MediaController changes.
`EVENT_REMINDER`	Event triggered when the STB sends a reminder.

8.2.2.1 EVENT_GO_CHANNEL

Table 8-3 gives the data structure of the Event object when the IPTV TD successfully joins in a live channel.

Table 8-3 – Definition of EVENT_GO_CHANNEL

Properties	DataType	Optionality	Instructions
`type`	String(64)	M	The value is `EVENT_GO_CHANNEL`
`instance_id`	unsignedInt [0:255]	M	The native MediaController instance ID of the STB
`channel_code`	unsignedInt [0:4294967295]	M	The unified code of the channel
`channel_num`	unsignedInt [0:4294967295]	O	The channel number

8.2.2.2 EVENT_MEDIA_END

Table 8-4 gives the data structure of the Event object when the IPTV TD quits a VoD or nPVR.

Table 8-4 – Definition of EVENT_MEDIA_END

Properties	DataType	Optionality	Instructions
`type`	String(64)	M	The value is `EVENT_MEDIA_END`
`instance_id`	unsignedInt[0:255]	M	The native MediaController instance ID of the STB
`media_code`	String	O	The unified code of the media
`entry_id`	String(1024)	O	The unified code of the media in the media playlist

8.2.2.3 EVENT_MEDIA_BEGINNING

Table 8-5 gives the data structure of the Event object when the IPTV TD starts a VoD or nPVR.

Table 8-5 – Definition of EVENT_MEDIA_BEGINNING

Properties	DataType	Optionality	Instructions
`type`	String(64)	M	The value is `EVENT_MEDIA_BEGINNING`
`instance_id`	unsignedInt[0:255]	M	The native MediaController instance ID of the STB
`media_code`	String	O	The unified code of the media
`entry_id`	String(1024)	O	The unified code of the media in the media playlist

8.2.2.4 EVENT_MEDIA_ERROR

Table 8-6 gives the data structure of the Event object when any error occurs during the VoD or nPVR.

Table 8-6 – Definition of EVENT_MEDIA_ERROR

Properties	DataType	Optionality	Instructions
`type`	String(64)	M	The value is `EVENT_MEDIA_ERROR`
`instance_id`	unsignedInt[0:255]	M	The native MediaController instance ID of the STB
`error_code`	int	M	The code of the error
`error_message`	String	O	The detail of the error
`media_code`	String	O	The unified code of the media

8.2.2.5 EVENT_PLAYMODE_CHANGE

Table 8-7 gives the data structure of the Event object when the event of the pause, fast rewind and fast forward actions is triggered by the end user and successfully handled by the IPTV TD during the VoD or nPVR.

Table 8-7 – Definition of EVENT_PLAYMODE_CHANGE

Properties	DataType	Optionality	Instructions
`type`	String(64)	M	The value is `EVENT_PLAYMODE_CHANGE`
`instance_id`	unsignedInt[0:255]	M	The native MediaController instance ID of the STB
`new_play_mode`	unsignedInt[0:255]	M	It is the play mode after the play mode changes. For possible values refer to Table 8-8.
`new_play_rate`	int	O	When the play mode is trick mode, the information of the play rate has to be provided. For possible values refer to Table 8-9.
`old_play_mode`	unsignedInt[0:255]	M	This is the play mode before the play mode changes. For possible values refer to Table 8-8.
`old_play_rate`	int	O	When the play mode is trick mode, the information of the play rate has to be provided. For possible values refer to Table 8-9.

8.2.2.5.1 PLAY_MODE

Table 8-8 gives the possible values of the PLAY_MODE.

Table 8-8 – Definition of PLAY_MODE

Value	PLAY MODE	Instructions
0	STOP	Stop
1	PAUSE	Pause
2	NORMAL_PLAY	Normal play
3	TRICK_MODE	Fast forward, fast rewind, slow forward and slow rewind
4	MULTICAST_CHANNEL_PLAY	The broadcast state of the multicast channel.
5	UNICAST_CHANEL_PLAY	The broadcast state of the unicast channel.
Others		Unspecified

8.2.2.5.2 PLAY_RATE

The support of trick mode in VoD service is recommended in [ITU-T Y.1901]. The applicable PLAY_RATE values are listed in Table 8-9, with their respective descriptions. In the table, positive numbers stand for normal and forward play, and the negative values stand for rewind play. It should be noted that the values need not be restricted to those listed in Table 8-9.

Table 8-9 – Definition of PLAY_RATE

Value	Instructions
1	Normal speed
2	Double speed fast forward
4	Quad speed fast forward
0.5	Half speed slow forward
0	Pause
-1	Reverse play at normal speed
-2	Double speed fast rewind
-4	Quad-speed fast rewind
-0.5	Half-speed slow rewind

8.2.2.6 EVENT_REMINDER

Table 8-10 gives the data structure of the Event object when some messages need to be informed by the service component layer to the presentation engine layer.

Table 8-10 – Definition of EVENT_REMINDER

Properties	DataType	Optionality	Instructions
type	String(64)	M	The value is EVENT_REMINDER
message	String(2048)	M	The content of the reminder

8.3 Service object

Once the web-based IPTV terminal device connects to a service discovery server and an IPTV service provider from the service list is chosen by the end user, the service information of the chosen IPTV service provider (SP) should be specified. For web-based terminal devices that follow this Recommendation, a top-level, built-in object which is called a service object must be implemented.

The service object can be accessed without using a constructor and can be called by the SPs to specify the service information used by the end users during the service procedure. By calling the object methods in the web documents, SPs can specify the necessary service information, such as the service URL for different colour keys on the remote, the service entry for linear TV with trick mode and VoD, etc.

8.3.1 Content-related functionalities

8.3.1.1 Properties

No property is defined for content-related functionalities.

8.3.1.2 Methods

8.3.1.2.1 getContentPackageEndDate

To obtain the end date of the purchased content package.

- Optionality: Optional
- Syntax:
```
Date getContentPackageEndDate ( input String id )
```
- Parameters:
  - `id (String)`: the identifier for the content package.
- Return value:
  - Valid end date of the content package is returned

8.3.1.2.2 getContentPackageStartDate

To obtain the start date of the purchased content package.

- Optionality: Optional
- Syntax:
```
date getContentPackageStartDate ( input String id )
```
- Parameters:
  - `id ( String )`: the identifier for the content package.
- Return value:
  - Valid start date of the content package is returned

8.3.1.2.3 setContentPackageInfo

To set the information on the purchased content package.

- Optionality: Optional
- Syntax:
```
Number setContentPackageInfo ( input String id
   ,input Date valid_start_date
   [,input Date valid_end_date] )
```
- Parameters:
  - `id (String)`: the identifier for the content package.
  - `valid_start_date (Date)`: valid start date of the content package.
  - `valid_end_date (Date):` valid end date of the content package.

- Return value:
  - `1`: success
  - `NaN`: failure

8.3.2 Parental control

Parental control-related properties and methods allow the IPTV SPs to provide parental control services on web documents.

8.3.2.1 Properties

8.3.2.1.1 parentalCtrlEnabled

Enable parental control.

- Optionality: Mandatory
- DataType: Boolean
- Reference value:
  - `true`: parental control enabled
  - `false`: parental control disabled
- Read-write: R/W

8.3.2.1.2 parentalCtrlPassword

The parental control password.

- Optionality: Mandatory
- DataType: `String`
- Reference value:
  - None.
- Read-write: R/W

8.3.2.2 Methods

8.3.2.2.1 enableParentalCtrl

Enable parental control functionality.

- Optionality: Optional
- Syntax:

```
Number enableParentalCtrl ( input Boolean parentalCtrlEnabled )
```

- Parameters:
  - `parentalCtrlEnabled ( Boolean )`: enable or disable parental control.
- Return value:
  - `0`: success
  - `-1`: failure

8.3.2.2.2 checkParentalCtrlPassword

To check the parental control password: confirms the password for ensuring parental control.

- Optionality: Optional
- Syntax:

```
Number checkParentalCtrlPassword ( input String parentalCtrlPassword )
```

- Parameters:
  - `parentalCtrlPassword (String)`: the password of parental control.
- Return value:
  - `1`: Password verified successfully, or parental control is unlocked
  - `0`: incorrect password
  - `NaN`: Password has not been set

8.3.2.2.3 setParentalCtrlPassword

To set the parental control password.

- Optionality: Optional
- Syntax:

  `Number setParentalCtrlPassword ( input String parentalCtrlPassword )`
- Parameters:
  - `parentalCtrlPassword (String)`: the password of parental control.
- Return value:
  - `0`: success
  - `-1`: failure

8.3.3 Service registration

8.3.3.1 Properties

8.3.3.1.1 serviceEntry

This property identifies unique service information kept by the terminal itself, which can be defined by the SPs to approve the user identity and designate the available services for the end user.

- Optionality: Optional
- DataType: `String`
- Reference value:
  - The service information that can be provided by the SPs is listed in Table 8-11.
- Read-write: R

Table 8-11 – Fields defined for serviceEntry

Field name	Instructions
`PortalURL`	URL of the portal for a specific SP service platform
`BTVEPGURL`	URL defined to access the webpage presenting the channel list
`VoDEPGURL`	URL defined to access the webpage presenting VoD programmes
`SelfServiceEPGURL`	URL defined to access the webpage presenting customer self service
`UserSpaceURL`	URL defined to access the webpage presenting IPTV STB local services
`InfoEPGURL`	URL defined to access the webpage presenting IPTV information services
`GameEPGURL`	URL defined to access the webpage presenting IPTV games services
`EmailEPGURL`	URL defined to access the webpage presenting IPTV mail services
`UMEPGURL`	URL defined to access the webpage presenting IPTV on-line message services

8.3.3.2 Methods

8.3.3.2.1 checkIPTVServiceRegistrationInfo

To find out the information related to the basic IPTV service, i.e., linear TV and VoD.

- Optionality: Optional
- Syntax:

```
Array checkIPTVServiceRegistrationInfo ( input String id )
```

- Parameters:
  - `id (String)`: Service provider ID
- Return value:

  Array that stores data
  - Array containing data: Success
  - `Array[0] (String)`: key
  - `Array[1] (Date)`: `expire_date`
  - `Array[2] (String)`: `license_uri`
  - `null`: failure

8.3.3.2.2 getServiceEntryURL

The method provides the ability to get service information from the STB, which is specified by the *setServiceEntry* method. This method returns the registered service entry URL.

- Optionality: Optional
- Syntax:

```
String getServiceEntryURL (input String serviceEntry)
```

- Parameters:
  - `serviceEntry (String)`: An identifier for different service entries.
- Return value:
  - The URL of the registered service entry is returned to describe the service entry.

8.3.3.2.3 setIPTVServiceRegistrationInfo

To set the information related to a basic IPTV service, i.e., linear TV and VoD.

- Optionality: Optional
- Syntax:

```
Number setIPTVServiceRegistrationInfo ( input String id
   ,input String key
   ,input Date expire_date
   [,input String license_uri
    ,input String signature
    ,input String certificate_uri])
```

- Parameters:
  - `id (String)`: service provider ID
  - `key (String)`: key information for authentication
  - `expire_date (Date)`: basic registration expiration date/time
  - `license_uri (String)`: URI of the SCP server for linear TV/VoD of the licenser
  - `signature (String)`: signature of `license_uri`

  - `certificate_uri (String)`: URI of the public key certificate (chain) used for signature verification
- Return value:
  - `1`: success
  - `-1`: failed to verify the signature of the SCP server URL
  - `NaN`: other failure

8.3.3.2.4 setServiceEntry

To register service information in the STB. When this method is called, the former service information registered for the same service entry will be substituted.

- Optionality: Optional
- Syntax:

```
Number setServiceEntry ( input String serviceEntry
    , input String serviceEntryURL
    [, input String serviceEntryHotKey,
       input String serviceEntryDesc] )
```

- Parameters:
  - `serviceEntry (String)`: An identifier for different service entries.
  - `serviceEntryURL (String)`: A URL for a specified service entry.
  - `serviceEntryHotKey (String)`: Hot keys for a specified service entry, such as the blue, red, green, yellow colour keys.
  - `serviceEntryDesc (String)`: A brief introduction about the service entry.
- Return value:
  - `0`: success
  - `-1`: failure

8.3.4 Licences and DRM

8.3.4.1 Properties

8.3.4.1.1 drmSystem

The location identifier of the DRM system.

- Optionality: Mandatory
- DataType: `String`
- Reference value:
  - None
- Read-write: R/W

8.3.4.2 Methods

8.3.4.2.1 getDRMID

To obtain the identifier of the CAS/DRM client supporting the specified CAS/DRM.

- Optionality: Optional
- Syntax:

```
String getDRMID ( input String drm_system
    , input String id
    , input Array license_id )
```

- Parameters:
  - `drm_system (String)`: the string that identifies the DRM and/or conditional access system
  - `id (String)`: the identifier for the service provider
  - `license_id (Array)`: the array of the identifiers for the licences to be used
- Return value:
  - `DRMID`: success

8.3.4.2.2 getIPTVLicense

Obtain the licence for the specified content.

- Optionality: Optional
- Syntax:

```
Number getIPTVLicense ( input String drm_system
    , input String id
    , input Array license_id )
```

- Parameters:
  - `drm_system (String)`: the string that identifies the DRM and/or conditional access system
  - `id (String)`: the identifier for the service provider
  - `license_id (Array)`: the array of the identifiers for the licences to be used
- Return value:
  - `1`: incorrect SCP method
  - `2`: unknown SCP server URL
  - `3`: licence acquisition rejected
  - `4` client authentication failure on the server
  - `5` server authentication failure
  - `6` communication error with SCP server
  - `NaN`: other failure

8.3.4.2.3 getIPTVLicenseInfo

To obtain information concerning a specified licence.

- Optionality: Optional
- Syntax:

```
Array getIPTVLicenseInfo ( input String license_id
    ,input Number search_type )
```

- Parameters:
  - `license_id (String)`: The ID of the specified licence.
  - `search_type (Number)`: type of searching for the specified licence.
- Return value:
  - Arrays with information: success
  - `Array[1] (Date)`: viewing start time
  - `Array[2] (Date)`: viewing end time
  - `Array[3] (String)`: tier bit array

  - `Array[4] (Boolean)`: update-false: no, update-true: yes
  - `Array[5] (Date)`: update start date and time
  - `-1`: specified licence does not exist
  - `NaN`: other failure
  - `null`: failure

8.3.4.2.4 updatePackageLicenseInfo

To update all the package information

- Optionality: Optional
- Syntax:

  `Number updatePackageLicenseInfo ( input String id )`
- Parameters:
  - `id(String)`: the identifier for the package
- Return value:
  - `1`: success
  - `NaN`: failure

8.4 XMLHttpRequest object

The XMLHttpRequest object implements an interface exposed by a scripting engine that allows scripts to perform an HTTP client functionality (for example, submission of form data or loading data from a server). By using the XMLHttpRequest object, the user agent may retrieve and submit XML data directly in the background and conveniently connect an HTML presentation directly to XML data for interim updates without reloading the page. The user agent converts the retrieved XML data into renderable HTML content by reading the XML document node tree and composing HTML elements.

The XMLHttpRequest object is an HTTP application programming interface used by scripts to programmatically send HTTP or HTTPS requests directly to a web server and load the server response data directly back into the script. The data might be received from the server either as XML text or as plain text. Data from the response can be used directly to alter the document object module (DOM) of the currently active document in the user agent without loading a new webpage document. The response data can also be evaluated by client-side scripting.

8.4.1 Properties

The properties provided by the XMLHttpRequest object indicate related status changes; once a request has been sent by the user agent, the data are retrieved from the server.

8.4.1.1 onreadystatechange

An event handler for an event that is triggered at every state change.

- Optionality: Mandatory
- DataType: event handler for call back function.
- Reference value:
  - None
- Read-write: R

8.4.1.2 readyState

The *readyState* should be used inside the event handler function that processes request object state change events. While the object may undergo interim state changes during its creation and processing, the value that signals the completion of the transaction is `4` (complete).

- Optionality: Mandatory
- DataType: `unsignedInt[0:4]`.
- Reference value:
  - `0`: uninitialized
  - `1`: loading
  - `2`: loaded
  - `3`: interactive
  - `4`: complete
- Read-write: R

8.4.1.3 responseText and responseXML

The `responseText` property is the `String` version of the data returned from the server process. The `responseXML` property is a DOM-compatible document object of data returned from the server process. Note that the `responseXML` property is a fully-fledged *document* node object, which contains XML rather than HTML, and can be examined and parsed using DOM node tree methods and properties.

- Optionality: Mandatory
- DataType: `String`
- Reference value:
  - None
- Read-write: R

8.4.1.4 status and statusText

The status property is a numeric code returned by the server, such as 404 for "Not Found" or 200 for "OK". The statusText property is the `String` message accompanying the status code. Accessing the `status` property provides more information on whether the transaction has been completed successfully; however, further checks are advisable before operating on the results.

- Optionality: Mandatory
- DataType:
  - `status: integer`
  - `statusText: String`
- Reference value:
  - None
- Read-write: R

8.4.2 Methods

8.4.2.1 Constructor

The constructor of the XMLHttpRequest object with a default value.

- Optionality: Mandatory
- Syntax:

```
XMLHttpRequest()
```

- Parameters:
  – None
- Return value:
  – An instantiated XMLHttpRequest object.

8.4.2.2 abort

Stops the current request.

- Optionality: Mandatory
- Syntax:

```
abort()
```

- Parameters:
  – None
- Return value:
  – None

8.4.2.3 getAllResponseHeaders

Gets a complete set of headers (labels and values).

- Optionality: Mandatory
- Syntax:

```
String getAllResponseHeaders()
```

- Parameters:
  – None
- Return value:
  – Complete set of headers as a `String`.

8.4.2.4 getResponseHeader

Returns the `String` value of a single header label.

- Optionality: Mandatory
- Syntax:

```
String getResponseHeader ( input String headerLabel )
```

- Parameters:
  – `headerlabel`: available label defined in the HTTP header field such as "Content-type", "Content-length", etc.
- Return value:
  – Formatted `String` of the HTTP header specified by `headerlabel`.

8.4.2.5 open

Assigns the destination URL, method, and other optional attributes of a pending request.

- Optionality: Mandatory
- Syntax:

```
open ( input String method,
    input String URL
    [, input Boolean asyncFlag
    [, input String userName
    [,input String password]]])
```

- Parameters:
  - `method`: the HTTP method intended for the request. For the method parameter, use "*GET*" on operations that are primarily data retrieval requests; use "*POST*" on operations that send data to the server, especially if the length of the outgoing data is potentially greater than 512 bytes.
  - `URL`: the URL for the connection. It may be either a complete or a relative URL.
  - `asyncFlag`: Controls whether the upcoming transaction should be handled asynchronously. The default value "`true`" is to act asynchronously, which means that script processing carries on immediately after the `send()` method is invoked, without waiting for a response. The value "`false`" means that the script waits for the request to be sent and for a response to arrive from the server.
  - `userName`: the username to access the server.
  - `password`: the password to access the server.
- Return value:
  - None

8.4.2.6 send

Transmits the request to the server.

- Optionality: Mandatory
- Syntax:

```
send ( input String content )
```

- Parameters:
  - `content`: the data transmitted to the server, which is optionally with a postable `String` or DOM object data.
- Return value:
  - None

8.4.2.7 setRequestHeader

Assigns a label/value pair to the header to be sent with a request.

- Optionality: Mandatory
- Syntax:

```
setRequestHeader ( input String headerlabel
    , input String value )
```

- Parameters:
  - `headerlabel`: available label defined in the HTTP header field such as "Content-type", "Content-length", etc.
  - `value`: formatted String of the specified HTTP header label.
- Return value:
  - None

Annex A

Data types

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

The following table lists the set of extra data types that are not defined in the core script profile and which is utilized in this Recommendation, along with the notation used to represent these types.

Table A.1 – Definition of data types

Type	**Description**
String	For strings listed in this specification, a maximum allowed length can be listed using the form String(N), where N is the maximum string length in characters. If a string does not have an explicitly indicated maximum length or is not an enumeration, the default maximum is 32 characters.
int	int, which in the range –2147483648 to +2147483647 (inclusive) is a pronoun for integer Number object in the core script profile. For some int types listed, a value range is given using the form int[Min:Max], where the Min and Max values are inclusive. If either Min or Max are missing, this indicates no limit.
unsignedInt	unsignedInt, which in the range 0 to 4294967295, inclusive, is a pronoun for unsigned integer Number object in the core script profile. For some unsignedInt types listed, a value range is given using the form unsignedInt[Min:Max], where the Min and Max are inclusive values. If either Min or Max are missing, this indicates no limit.
dateTime/Date	The subset of the [ISO/IEC 8601] date-time format. The dateTime equals the return value of toUTCString() of Date type defined in the core script profile. If the time is unknown or not applicable, the following value representing "Unknown Time" MUST be used: 00010101T000000Z.
anySimpleType	The value of an element defined to be of type "anySimpleType" MAY be of any simple data types, including (but not limited to) any of the other types listed in this table. The element of this type MUST be well formatted with a name-value pair such as XML, JSON, etc.

Annex B

Content Metadata used by MediaController object

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

According to [ITU-T H.750], content metadata is produced by content providers and service providers. Content providers typically supply the title, synopsis, genre and other descriptive metadata about the content. The formatted content metadata is set into the MediaController methods, like `setSingleMedia()`, `addMedia()`, etc.

It should be noted that not all the information provided by content providers is necessary for the MediaController to provide media control during the lifecycle of the media. The following metadata elements are the ones required in service and content metadata.

- Identifier
  - [b-ETSI TS 102 822-3-1] 6.3.6 programId, 6.3.7 groupId (Group), 6.4.2 InstanceMetadataId, 6.4.3 serviceId, 6.6.5 segmentId, 6.6.6 groupId (SegmentGroup)
  - [b-ETSI TS 102 822-3-3] 8.4 item_id, 8.5 component_id, 8.6 crid (Package)
  - [b-ETSI TS 102 471] 5.4.1 serviceID, 5.5.2 serviceBundleID, 5.6.1 contentID, 5.7.1 scheduleId
  - [b-UPnP CDS2] B.8.3 upnp:programID, B.8.4 upnp:seriesID, B.8.5 upnp:channelID, B.9.1.1 upnp:channelGroupName@id, B.10.2 upnp:radioStationID
  - [b-IETF RFC 4287] 4.2.6 id
- Encrypted or not
  - [b-ETSI TS 102 471] 5.4.1 clearToAir (Service), 5.7.1 clearToAir (ScheduleEvent)
- Codec or format
  - [b-ETSI TS 102 822-3-1] 6.3.5 Coding
  - [b-ETSI TS 102 822-3-3] 6.1.1.1 Coding (Audio), 6.1.1.2 Coding (Video)
  - [b-ETSI TS 102 471] 5.10.7.1 Codec, ProfileLevelIndication
- Aspect ratios, resolution, bit rate, frame rate for video
  - [b-ETSI TS 102 822-3-1] 6.3.5 AspectRatio, HorizontalSize, VerticalSize, Bitrate, FrameRate
  - [b-ETSI TS 102 822-3-3] 6.1.1.2 BitRate, AspectRatio, HorizontalSize, VerticalSize, FrameRate
  - [b-ETSI TS 102 471] 5.10.7.2 Bandwidth, FrameRate
  - [b-UPnP CDS2] B.2.1.10 res@resolution, B.2.1.6 res@bitrate
- Mono, stereo, multi-channel indication for audio
  - [b-ETSI TS 102 822-3-1] 6.3.5 MixType, NumOfChannels
  - [b-ETSI TS 102 822-3-3] 6.1.1.1 MixType, NumOfChannels
  - [b-ETSI TS 102 471] 5.10.7.2 Mode
  - [b-UPnP CDS2] B.2.1.9 res@nrAudioChannels

– Duration
  - [b-ETSI TS 102 822-3-1] 6.3.4 Duration (Program, Group), 6.6.5 SegmentLocator (Segment), 6.6.6 duration (SegmentGroup)
  - [b-ETSI TS 102 471] 5.6.2 Duration (Content)
  - [b-UPnP CDS2] B.2.1.4 res@duration

Appendix I

Remote control key code

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

The user agent of a web-based terminal device will dispatch the key press event to the web document with IPTV SESL. The user agent will handle the key code and execute the corresponding IPTV SESL functions predefined in the web document, as identified in Table I.1.

Table I.1 – Key code list

Remote control key	Key code	Remarks
Backspace	BACKSPACE	"Backspace" can be combined with "Stop" in a media playing scenario.
OK	OK	
Cancel	CANCEL	"Cancel" can be combined with "Stop" in a media playing scenario.
PageUp	PAGE_UP	
PageDown	PAGE_DOWN	
Left arrow	LEFT	
Up arrow	UP	
Right arrow	RIGHT	
Down arrow	DOWN	
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
#	POUND	
*	STAR	
Channel up	CH_UP	
Channel down	CH_DOWN	
Volume up	VOL_UP	
Volume down	VOL_DOWN	
Mute	MUTE	
Play	PLAY	"Play" and "Pause" are to always be used serially.
Pause	PAUSE	

Table I.1 – Key code list

Remote control key	**Key code**	**Remarks**
Fast rewind	FFR	"Fast rewind" can be combined with "Left arrow" in a media playing scenario.
Fast forward	FFW	"Fast Forward" can be combined with "Right arrow" in a media playing scenario.
Stop	STOP	
Go to end	GO_TO_END	Go directly to the end of the media.
Go to beginning	GO_TO_BEGIN	Go directly to the beginning of the media.
Portal key	MENU	Hot key for portal.
Red key	HKEY_RED	Hot key for specified URL defined by IPTV SP.
Green key	HKEY_GREEN	
Yellow key	HKEY_YELLOW	
Blue key	HKEY_BLUE	
Audio channel switching	AUDIO_CH_SWITCH	
Others	undefined	Other key codes on the remote control can be defined by the SPs at will.

Appendix II

Informative text on extended script profile

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

II.1 MediaController object

The following code sample explains how to create an instance of MediaController to control and display media content.

```
// A web page including the following ECMAscript sample can control and display
the media content:
// create an instance of MediaController
var mp = new MediaController ();
//  If this is the first time an instance of MediaController has been created,
then get the unique instanceId of the local media player created in the STB and
save it for probable further use in other webpages
var nativePlayerInstanceId = mp.instanceId;
// Or
// else the new player created by MediaController () just binds an existed
nativePlayerInstanceID. Note: In order to support services such as PIP, the STB
shall support more than one native controller simultaneous. In this case, the
instanceId property of different instances indicate different local media
players.
mp.bindNativePlayerInstance(nativePlayerInstanceId);
// set the URL of the media
  mp.setSingleMedia(mediaUrl);
// Start to play the media content
mp.play();
//Trick mode control such as FFW/RW/PAUSE.
mp.fastforward();
//Stop to play the media content and release the native media controller
mp.stop();
mp.releaseMediaController (nativePlayerInstanceId);
```

II.2 Event object

Figure II.1 illustrates the use of the Event object to provide the control of the media playing for users. The relationship with MediaController is explained in [ITU-T H.730].

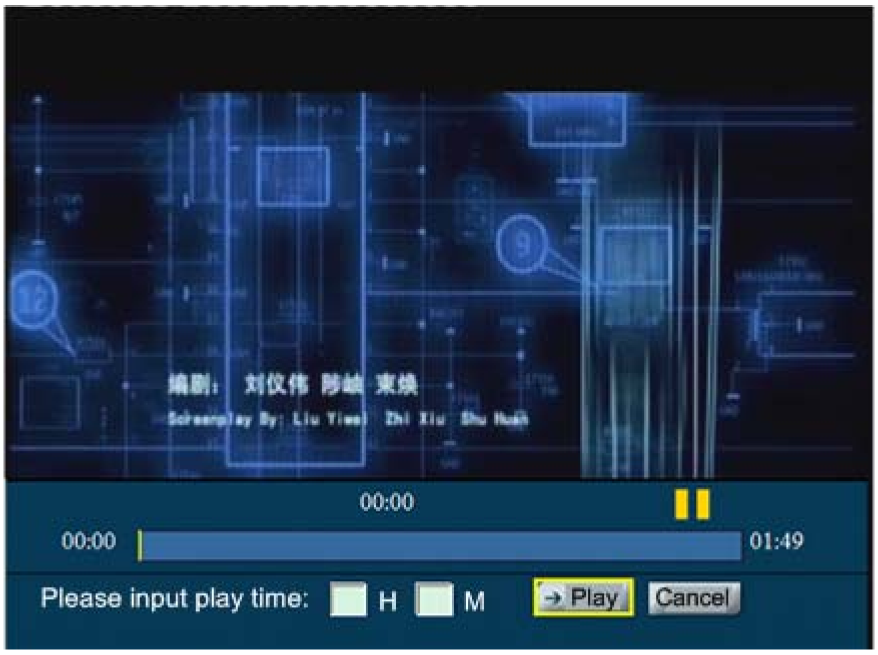

Figure II.1 – Example of the Event object controlling the media status

II.2.1 State transition for Event object in a VoD scenario

The Event object is used to inform webpages with IPTV SESL about the state transition of the MediaController object, which is caused by user interaction during the VoD scenario. The webpages provide the service information and control the service logic of the IPTV service.

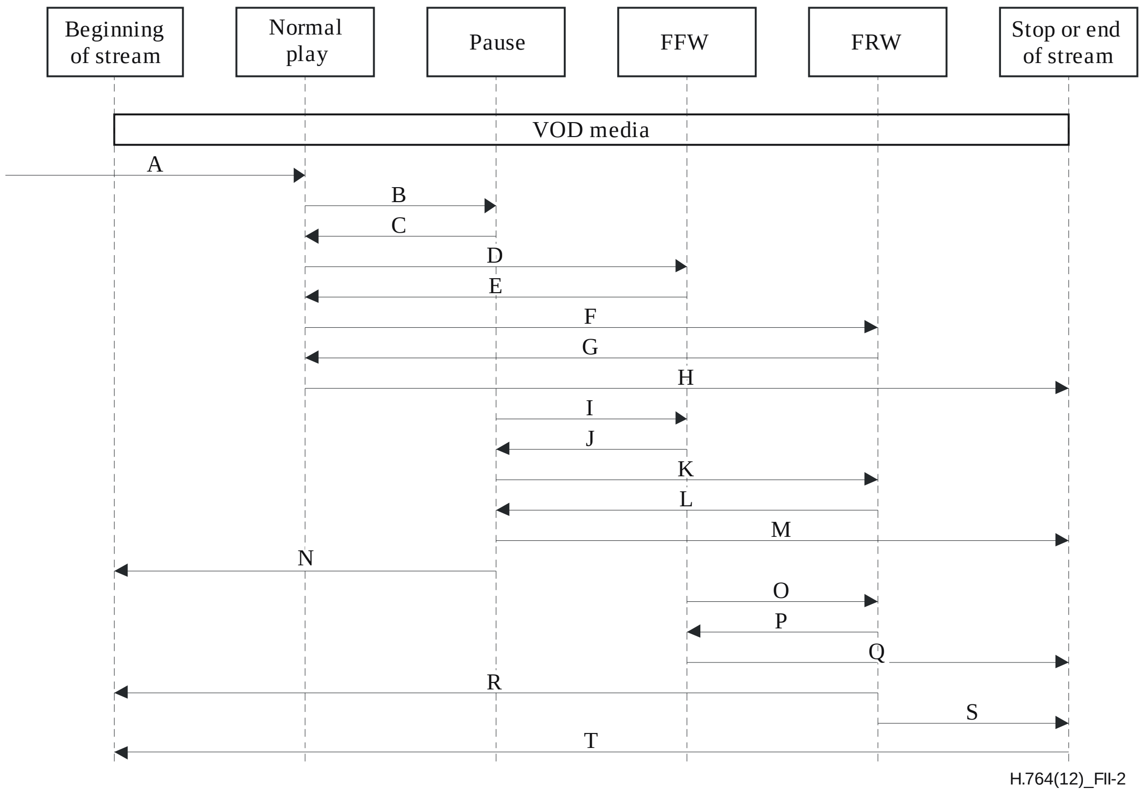

Figure II.2 – State transition for Event object in a VoD scenario

Assumptions

– The ID of the MediaController object instance in this scenario, which is described by "instance_id", is set to 1.

– The code of the VoD media, which is described by "media_code", is set to "vod/2102". The "entry_id" of the media is set to "12".

– The sampled play rate is limited to the listed values in Table 8-9 "Definition of PLAY_RATE".

– The information of the Event object can be encapsulated with "anySimpleType", which is defined in Annex A, and the XML formatted expression is used here.

Instructions

The state transition information caused by user interaction e.g., from "normal play" to "pause" is generated by the media client in the IPTV terminal device and is sent to the presentation engine agent. The event dispatching component in the presentation engine will generate the Event object structure and send it to the agent so that the agent can inform the webpages with IPTV SESL to handle the service logic by e.g., changing the display of webpages, etc. The IPTV SESL execution functional block in the presentation engine will execute the extended script profile according to the information contained in the Event object.

State transition A: starting to play the VoD from the beginning

```
<EVENT>
  <type> EVENT_MEDIA_BEGINNING</type>
  <instance_id>1</instance_id>
  <media_code>vod/2102</media_code>
  <entry_id>12</entry_id>
</EVENT>
```

State transition B: from play to pause

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>1</new_play_mode>
  <new_play_rate>0</new_play_rate>
  <old_play_mode>2</old_play_mode>
  <old_play_rate>1</old_play_rate>
</EVENT>
```

State transition C: from pause to play

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>2</new_play_mode>
  <new_play_rate>1</new_play_rate>
  <old_play_mode>1</old_play_mode>
  <old_play_rate>0</old_play_rate>
</EVENT>
```

State transition D: from play to forward

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>3</new_play_mode>
  <new_play_rate>2</new_play_rate>
  <old_play_mode>2</old_play_mode>
  <old_play_rate>1</old_play_rate>
</EVENT>
```

State transition E: from forward to play

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>2</new_play_mode>
  <new_play_rate>1</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>2</old_play_rate>
</EVENT>
```

State transition F: from normal play to rewind play

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>3</new_play_mode>
  <new_play_rate>-2</new_play_rate>
  <old_play_mode>2</old_play_mode>
  <old_play_rate>1</old_play_rate>
</EVENT>
```

State transition G: from rewind to play

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>2</new_play_mode>
  <new_play_rate>1</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>-2</old_play_rate>
</EVENT>
```

State transition H: from play to stop or end of the VoD

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
  <media_code>vod/2102</media_code>
  <entry_id>12</entry_id>
</EVENT>
```

State transition I: from pause to forward

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>3</new_play_mode>
  <new_play_rate>2</new_play_rate>
  <old_play_mode>1</old_play_mode>
  <old_play_rate>0</old_play_rate>
</EVENT>
```

State transition J: from forward to pause

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>1</new_play_mode>
  <new_play_rate>0</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>2</old_play_rate>
</EVENT>
```

State transition K: from pause to rewind

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>3</new_play_mode>
  <new_play_rate>-2</new_play_rate>
  <old_play_mode>1</old_play_mode>
  <old_play_rate>0</old_play_rate>
</EVENT>
```

State transition L: from rewind play to pause

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>1</new_play_mode>
  <new_play_rate>0</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>-2</old_play_rate>
</EVENT>
```

State transition M: from pause to stop

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
  <media_code>vod/2102</media_code>
  <entry_id>12</entry_id>
</EVENT>
```

State transition N: from pause to beginning of the VoD

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>2</new_play_mode>
  <new_play_rate>1</new_play_rate>
  <old_play_mode>1</old_play_mode>
  <old_play_rate>0</old_play_rate>
</EVENT>

<EVENT>
  <type>EVENT_MEDIA_BEGINNING</type>
  <instance_id>1</instance_id>
  <media_code>vod/2102</media_code>
  <entry_id>12</entry_id>
</EVENT>
```

State transition O: from forward to rewind

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>3</new_play_mode>
  <new_play_rate>-2</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>2</old_play_rate>
</EVENT>
```

State transition P: from rewind to forward

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>3</new_play_mode>
  <new_play_rate>2</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>-2</old_play_rate>
</EVENT>
```

State transition Q: from forward play to stop or end of the VoD

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
  <media_code>vod/2102</media_code>
  <entry_id>12</entry_id>
</EVENT>
```

State transition R: from rewind until reaching the beginning of the VoD

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>2</new_play_mode>
  <new_play_rate>1</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>-2</old_play_rate>
</EVENT>

<EVENT>
  <type>EVENT_MEDIA_BEGINNING</type>
  <instance_id>1</instance_id>
  <media_code>vod/2102</media_code>
  <entry_id>12</entry_id>
</EVENT>
```

State transition S: from rewind play to stop or the end of the VoD

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
  <media_code>vod/2102</media_code>
  <entry_id>12</entry_id>
</EVENT>
```

State transition T: rewind the media until reaching the the beginning of the VoD

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>2</new_play_mode>
  <new_play_rate>1</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>-2</old_play_rate>
</EVENT>
//If the media reaches the beginning, the following event will occur.
<EVENT>
  <type>EVENT_MEDIA_BEGINNING</type>
  <instance_id>1</instance_id>
  <media_code>vod/2102</media_code>
  <entry_id>12</entry_id>
</EVENT>
```

II.2.2 State transition for Event object in a linear TV with trick mode scenario

The Event object is used to inform the webpages with IPTV SESL about a state transition in the MediaController object (caused by user interaction during the linear TV with trick mode scenario). The Event object provides the service information and controls the service logic of the IPTV service.

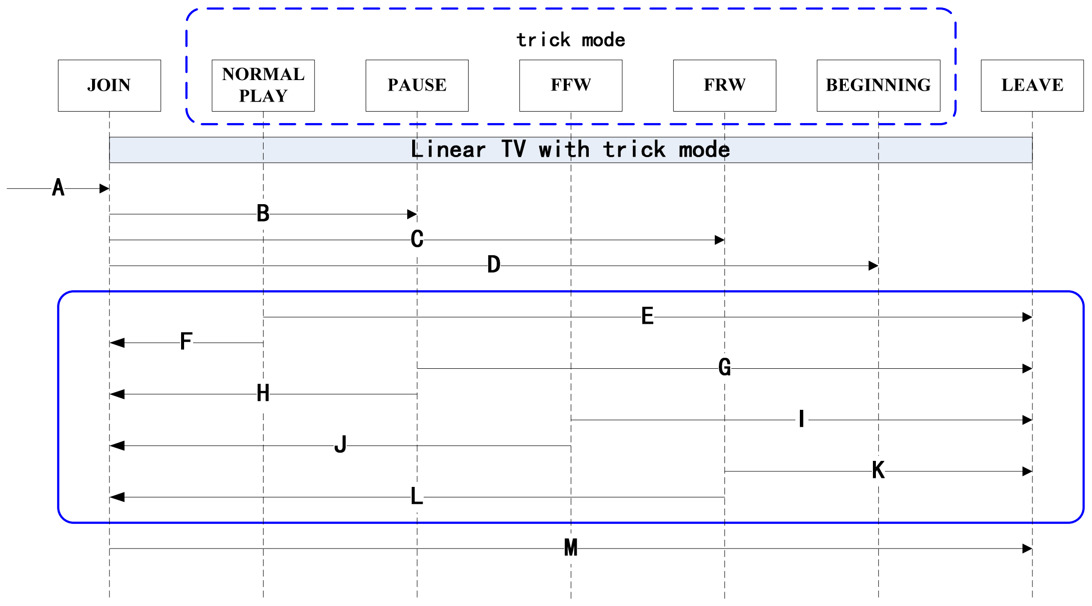

Figure II.3 – State transition for Event object in a linear TV with trick mode scenario

Assumptions

– The ID of the MediaController object instance in this scenario, which is described by "instance_id", is set to 1.

– The unique ID of a linear TV channel with trick mode, which is described by "channel_code", is set to "433". The channel number displayed on screen, which is described by "channel_num", is set to "1".

– The sampled play rate is limited to the listed values in Table 8-9.

– The information of the Event object can be encapsulated with "anySimpleType" (see definition in Annex A) and the XML formatted expression is used.

Instructions

The state transition information caused by user interaction e.g., from "normal play" to "pause" is generated by the media client in the IPTV terminal device and is sent to the presentation engine agent, which could be a browser, by internal interfaces defined in [ITU-T H.730]. In this scenario, the WBTM and IPTV SESL work in the same way as in the VoD scenario.

State transition A: joining linear TV with trick mode

```
<EVENT>
  <type>EVENT_GO_CHANNEL</type>
  <instance_id>1</instance_id>
  < channel_code >433</ channel_code >
  < channel_num >1</ channel_num >
</EVENT>
```

State transition B: from linear TV with trick mode to pause (trick mode)

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>1</new_play_mode>
  <new_play_rate>0</new_play_rate>
  <old_play_mode>4</old_play_mode>
  <old_play_rate>1</old_play_rate>
</EVENT>
```

State transition C: from linear TV to rewind (trick mode)

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>3</new_play_mode>
  <new_play_rate>-2</new_play_rate>
  <old_play_mode>4</old_play_mode>
  <old_play_rate>1</old_play_rate>
</EVENT>
```

State transition D: from linear TV to the left border of the record

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>2</new_play_mode>
  <new_play_rate>1</new_play_rate>
  <old_play_mode>4</old_play_mode>
  <old_play_rate>1</old_play_rate>
</EVENT>
```

State transition E: from play (trick mode) to leaving linear TV

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
</EVENT>
```

State transition F: from trick mode to re-joining linear TV

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>4</new_play_mode>
  <new_play_rate>-1</new_play_rate>
  <old_play_mode>2</old_play_mode>
  <old_play_rate>1</old_play_rate>
</EVENT>
```

State transition G: from pause (trick mode) to leaving linear TV

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
</EVENT>
```

State transition H: from pause state (trick mode) to re-join the live TV channel

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>4</new_play_mode>
  <new_play_rate>-1</new_play_rate>
  <old_play_mode>1</old_play_mode>
  <old_play_rate>0</old_play_rate>
</EVENT>
```

State transition I: from forward (trick mode) to leaving linear TV

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
</EVENT>
```

State transition J: from forward (trick mode) to re-joining linear TV

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>4</new_play_mode>
  <new_play_rate>-1</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>2</old_play_rate>
</EVENT>
```

State transition K: from rewind play state (trick mode) to leaving the live TV channel

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
</EVENT>
```

State transition L: from rewind (trick mode) to re-joining linear TV

```
<EVENT>
  <type>EVENT_PLAYMODE_CHANGE</type>
  <instance_id>1</instance_id>
  <new_play_mode>4</new_play_mode>
  <new_play_rate>1</new_play_rate>
  <old_play_mode>3</old_play_mode>
  <old_play_rate>-2</old_play_rate>
</EVENT>
```

State transition M: from play (trick mode) to leaving linear TV

```
<EVENT>
  <type>EVENT_MEDIA_END</type>
  <instance_id>1</instance_id>
</EVENT>
```

II.3 Service object

When the web-based IPTV terminal boots up, the browser is initialized to progress service discovery, which is a specific URI (URL) for the browser. When it visits the URL, the service server will respond with a webpage containing LIME HTML [ITU-T H.762] and IPTV SESL. A service object could be embedded in the webpage to gain information about the end user after user authentication; additionally, SPs can provide individual services for the user.

Since the identity of the STB and the end user can be specified after the user authentication procedure, the SP service server will redirect the browser to a webpage that provides a typical service of the specified SP and an individual service for the end user. The service object here should provide a method to keep this service information in the terminal for subsequent usage during the whole service session.

Since the service object provides the ability for SPs to approve user identity and designate the available services for the end user, unique service information could be kept by the terminal itself and be called by the EPG (which is provided by the same SP) to keep in step with the service attribute of the end user.

Bibliography

[b-ECMA-262]	Standard ECMA-262 Edition 5.1 (2011), *ECMAScript Language Specification*.
[b-ETSI TS 102 471]	ETSI TS 102 471 V1.2.1 (2006), *Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Electronic Service Guide (ESG)*.
[b-ETSI TS 102 822-3-1]	ETSI TS 102 822-3-1 V1.4.1 (2007), *Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 1: Phase 1 – Metadata schemas*.
[b-ETSI TS 102 822-3-3]	ETSI TS 102 822-3-3 V1.2.1 (2007), *Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 3: Phase 2 – Extended Metadata Schema*.
[b-IETF RFC 4287]	IETF RFC 4287 (2005), *The Atom Syndication Format*.
[b-UPnP CDS2]	UPnP Forum (2006), *ContentDirectory:2 Service Template Version 1.01*.
[b-W3C WebArch]	W3C Recommendation, *Architecture of the World Wide Web, Volume One (2004)*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
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	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	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
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 and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2013