International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.762

(05/2011)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

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

Lightweight interactive multimedia environment (LIME) for IPTV services

Recommendation ITU-T H.762

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
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

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

Recommendation ITU-T H.762

Lightweight interactive multimedia environment (LIME) for IPTV services

Summary

Recommendation ITU-T H.762 describes the high-level functionalities of the lightweight interactive multimedia environment (LIME) for IPTV. LIME supports functionalities in IPTV terminal devices to provide interactivity and a variety of content such as audio, video, graphics and text. Expected services include additional data such as text to enrich television programmes, and two-way portal pages.

This Recommendation describes the profile called "LIME-HTML" of W3C Recommendation XHTML 1.0, the profile called "LIME-CSS" of cascading style sheets 1 (CSS1), and a part of CSS2, the profile of document object model (DOM) called "LIME-DOM", and a script language called "LIME-Script" that is a subset of ECMAScript but has functional extensions required for IPTV services. It describes the use of IP-based protocols for transport of LIME and IPTV-related services.

This revision includes the addition of explanatory text and definitions to enhance readability and improve the understanding of the extended functions. Specifically, the change of the string "BML" to "LIME" and the addition of explanations of the browser pseudo-object methods were considered. These changes do not modify any accepted functions nor introduce any new functionality.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T H.762	2009-12-14	16
2.0	ITU-T H.762	2011-05-14	16

Keywords

Application multimedia framework, character encoding, CSS, delivery protocol, DOM, ECMA script, HTML, metadata, monomedia, multimedia coding, XML.

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/.

Table of Contents

Page

Recommendation ITU-T H.762

Lightweight interactive multimedia environment (LIME) for IPTV services

1 Scope

This Recommendation describes the high-level functionalities of the lightweight interactive multimedia framework (LIME) for IPTV. LIME supports functionalities in IPTV terminal devices to provide interactivity and a variety of content such as audio, video, graphics and text. Expected services include additional data such as text to enrich TV programmes, and two-way portal pages.

This Recommendation describes the profile (called "LIME-HTML") of XHTML1.0 [b-W3C XHTML], the profile (called "LIME-CSS") of cascading style sheets 1(CSS1), and a part of CSS2, the profile of document object model (DOM) (called "LIME-DOM"), and a script language (called "LIME-Script") that is a subset of ECMAScript but has functional extensions required for IPTV services. It describes the use of IP-based protocols for transport of LIME and IPTV-related services.

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.262] Recommendation ITU-T H.262 (2000) | ISO/IEC 13818-2:2000, *Information technology – Generic coding of moving pictures and associated audio information: Video*.

[ITU-T H.264] Recommendation ITU-T H.264 (2011), *Advanced video coding for generic audiovisual services*.

[ITU-T H.720] Recommendation ITU-T H.720 (2008), *Overview of IPTV terminal devices and end systems.*

[ITU-T H.721] Recommendation ITU-T H.721 (2009), *IPTV terminal devices: Basic model*.

[ITU-T H.760] Recommendation ITU-T H.760 (2009), *Overview of multimedia application frameworks for IPTV services*.

[ITU-T H.761] Recommendation ITU-T H.761 (2011), *Nested context language (NCL) and Ginga-NCL*.

[ITU-T T.81] Recommendation ITU-T T.81 (1992) | ISO/IEC 10918-1:1994, *Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines*.

[IETF RFC 2616] IETF RFC 2616 (1999), *Hypertext Transfer Protocol – HTTP/1.1*.

[IETF RFC 2965] IETF RFC 2965 (2000), *HTTP State Management Mechanism*.

[ISO/IEC 9899] ISO/IEC 9899:1999, *Programming languages-C*.

[ISO/IEC 15948] ISO/IEC 15948:2004, *Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification*.

[ISO/IEC 11172-3] ISO/IEC 11172-3:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio*.

[ISO/IEC 13818-7] ISO/IEC 13818-7:1997, *Information Technology – Generic coding of moving pictures and associated audio information – Part 7: Advanced Audio Coding (AAC)*.

[ARIB STD-B24] ARIB STD-B24 V.5.2 (2008), *Data coding and Transmission Specification for Digital Broadcasting*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 application [b-ITU-T Y.101]: A functional implementation realized as software running in one or spread over several interplaying hardware entities.

3.1.2 broadcast markup language (BML) [ARIB STD-B24]: The XML application language specified in [ARIB STD-B24] is to be solely responsible for tags and attributes for multimedia representation.

3.1.3 ECMAScript [b-ISO/IEC 16262]: The programming language defined by [b-ISO/IEC 16262].

3.1.4 electronic programme guide (EPG) [b-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 electronic content guide (ECG) and broadband content guide (BCG) and bidirectional interactive service (sometimes called interactive programme guide (IPG)) for end-user interaction with a server or head-end.

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

3.1.6 IPTV terminal device [b-ITU-T Y.1901]: A terminal device which has IPTV terminal function (ITF) functionality, e.g., a set-top box (STB).

3.1.7 IPTV terminal function (ITF) [b-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 tear-down, and b) receiving the content of an IP transport from the network and rendering.

3.1.8 video-on-demand (VoD) [b-ITU-T Y.1901]: A service in which the end-user can, on demand, select and view a 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 some time after the selection of the video content.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 LIME-CSS: The profile of cascading style sheets (CSS) defined in this Recommendation that composes lightweight interactive multimedia environment (LIME).

3.2.2 LIME-DOM: The profile of the document object model (DOM) defined in this Recommendation that composes lightweight interactive multimedia environment (LIME).

3.2.3 LIME-HTML: The profile of extensible hypertext markup language (XHTML) defined in this Recommendation that composes lightweight interactive multimedia environment (LIME).

3.2.4 LIME-Script: The subset of ECMAScript defined in this Recommendation that composes lightweight interactive multimedia environment (LIME).

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AAC-LC	Advanced Audio Coding – Low Complexity
API	Application Programming Interface
BCG	Broadband Content Guide
BML	Broadcast Markup Language
CAS	Conditional Access System
CSS	Cascading Style Sheets
DII	Download Info Indication
DOM	Document Object Model
DRM	Digital Rights Management
DTD	Data Type Definition
ECG	Electronic Content Guide
ECMA	European Computer Manufacturers Association
EIT	Event Information Table
EPG	Electronic Programme Guide
ES	Elementary Stream
EUC-JP	Extended UNIX Code – Japanese
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
IPG	Interactive Programme Guide
IPTV	Internet Protocol TeleVision
ISDB-T	Integrated Services Digital Broadcasting – Terrestrial
ISP	Internet Service Provider

ITF	Internet protocol television Terminal Function
LIME	Lightweight Interactive Multimedia Environment for IPTV
MAFR	Multimedia Application Framework
MHEG	Multimedia and Hypermedia information coding Experts Group
MNG	Multiple-image Network Graphics
NPT	Normal Play Time
OSD	On-Screen Display
PES	Packetized Elementary Stream
PNG	Portable Network Graphics
SSL	Secure Socket Layer
STB	Set-Top Box
TCP	Transport Control Protocol
TLS	Transport Layer Security
TS	Transport Stream
URI	Uniform Resource Identifier
URL	Uniform resource locator
VoD	Video on demand
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language

5 Conventions

The following conventions are used to describe operational guidelines:

R1 Basic service required item. An IPTV terminal device designed for basic service should appropriately interpret the attribute if it is present in the content.

R2 Basic service required item. It is assumed that the value for this attribute is not present in the content. An IPTV terminal device designed for basic service assumes the default value for this attribute.

– Item not required for basic service. It is assumed the value for this attribute is not present in the content. An IPTV terminal device designed for basic service does not need to handle the attribute, even if it is present in the content.

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

RW Read/write for basic services. The corresponding attribute or property can be read and written. An IPTV terminal device designed for basic services should support the ability to read and write the corresponding attribute or property in the content.

R Read for basic services. The corresponding attribute can be read but cannot be written. An IPTV terminal device designed for basic services should support the ability to read and write the corresponding attribute or property in the content. Write operations to this item may be ignored.

6 Overview

This Recommendation describes the lightweight interactive multimedia environment (LIME) for IPTV services, the services such as those described in [ITU-T H.720]. It is expected to provide interactivity using multimedia to embedded IPTV terminal devices, such as those described in [ITU-T H.721]. LIME has been specified especially for the operation of portal service, as is described in [ITU-T H.721]. LIME has evolved from [b-ARIB TR-B14] volume 3, Section 2 (profile A), as well as [ARIB STD-B24], the so-called BML, but due to the differences in requirements and the environment between the data broadcasting service targeted by the original BML specifications and the IPTV service, some modifications were needed, resulting in LIME.

The main part of LIME consists of the following components:

– The profile hereinafter called "LIME-HTML" of XHTML 1.0. This profile is compliant with the "HTML for IPTV services" Recommendation of the multimedia application framework (MAFR) series currently under development.

– The profile hereinafter called "LIME-CSS" of CSS1 and a part of CSS2. This profile is compliant with the "CSS for IPTV services" Recommendation of the MAFR series currently under development.

– The profile of DOM, hereinafter called "LIME-DOM". This profile is compliant with the "DOM for IPTV services" Recommendation of the MAFR series currently under development.

– The script language hereinafter called "LIME-Script", which is a subset of ECMAScript but has functional extensions required for IPTV services. LIME-Script is compliant with the "ECMAScript for IPTV services" Recommendation of the MAFR series currently under development.

7 LIME-HTML

7.1 LIME-HTML document

A LIME-HTML document is an XHTML document, which includes CSS and script, i.e., ECMAScript. This clause describes requirements for a LIME-HTML document as an XHTML document.

7.1.1 Character encoding scheme

Refer to Annex C for the character encoding scheme used. Only one scheme must be used in any single LIME-HTML document and any external data, including LIME-Script files and LIME-CSS files, referenced by the document.

7.1.2 Declarations in a LIME-HTML document

7.1.2.1 XML declaration

A LIME-HTML document is required to begin with an XML declaration which specifies the version of XML being used. The XML version in the XML declaration is required to be 1.0.

In an encoding declaration, if the encoding is in the various encodings and transformations of Unicode/ISO/IEC 10646, the values "UTF-8", "UTF-16", "ISO-10646-UCS-2", and "ISO-10646-UCS-4" are required to be used. If the encoding is in JIS X-0208-1997, the value "euc-jp" is required to be used. It is recommended that character encodings registered (as charsets) with the Internet Assigned Numbers Authority (IANA), other than those just listed above, be referred to using their registered names.

7.1.3 XHTML elements of LIME-HTML

This clause describes elements that can be used in a LIME-HTML document.

7.1.3.1 Core modules

7.1.3.1.1 Structure module

This module includes elements: "body", "head" and "title".

7.1.3.1.2 Text module

This module includes elements: "br", "div", "p" and "span".

7.1.3.1.3 Hypertext module

This module defines an element for specifying hypertext links to other LIME-HTML documents. The module consists of the element: "a".

7.1.3.1.4 List module

This module defines elements for providing list-style presentations. This module does not include elements used in a LIME-HTML document.

7.1.3.2 Text extension modules

These modules are used to add textual presentations. A LIME-HTML document does not use elements from these modules.

7.1.3.3 Forms modules

7.1.3.3.1 Basic forms module and forms module

These modules define elements for controlling interactive data input operations. These modules for LIME-HTML include the element "input".

7.1.3.4 Table module

7.1.3.4.1 Basic table module and tables module

These modules define elements for providing table-style presentations. The elements from these modules are not used in LIME-HTML.

7.1.3.5 Image module

This module defines an element for embedding images in a LIME-HTML document. The module consists of the "img" element.

7.1.3.6 Client-side map module

This module defines elements for ensuring image mapping that is responsible for an IPTV terminal device or client. The elements from this module are not used in a LIME-HTML document.

7.1.3.7 Server-side map module

This module defines elements for ensuring image mapping that is responsible for a server. The elements from this module are not used in a LIME-HTML document.

7.1.3.8 Object module

This module defines elements for generic objects that represent images, video and audio. The module consists of the "object" element.

7.1.3.9 Frames module

This module defines elements for frame-style presentations. The elements from this module are not used in a LIME-HTML document.

7.1.3.10 Target module

This module defines attributes for describing target-related information. The elements from this module are not used in a LIME-HTML document.

7.1.3.11 Iframe module

This module defines elements for inserting frames into text. The module consists of the iframe element. The element from this module is not used in a LIME-HTML document.

7.1.3.12 Intrinsic events module

This module defines attributes that correspond to events generated by user operation. The attributes include the "onclick" attribute. The elements from this module are not used in a LIME-HTML document.

7.1.3.13 Meta-information module

This module defines elements for presenting meta-information of a document. The module consists of the "meta" element.

7.1.3.14 Scripting module

This module defines elements for scripts that describe behaviours and elements for controlling scripts. The module consists of the "script" element.

7.1.3.15 Style sheet module

This module defines elements for describing style sheets. The module consists of the "style" element.

7.1.3.16 Style attribute module

This module defines the style attribute.

7.1.3.17 Link module

This module defines an element for providing document-related information for a browser. The module consists of the "link" element.

7.1.3.18 Basic module

This module defines an element for defining a base uniform resource indicator (URI). The element from this module is not used in a LIME-HTML document.

7.1.3.19 Extension modules (LIME/basic LIME modules)

LIME-HTML has the following extension modules to define the following elements and attributes. The basic LIME module is limited to the basic features.

7.1.3.19.1 Basic LIME module

The elements from this module are not used in a LIME-HTML document.

7.1.3.19.2 LIME module

The LIME module supports the necessary features. This module includes elements: "bml", "bevent", "beitem", "body&", "div&", "p&", "span&", "object&".

7.1.4 Attributes

The following HTML attributes are used in a LIME-HTML document.

Table 7-1 – HTML attributes used in a LIME-HTML document

Elements	Attributes	Operation	Restrictions for operation
Common attributes Core attributes			
	id	R1	Character string with a maximum of 128 bytes
	class	R1	
	title	–	
I18N attributes			
	xml:lang	R2	(Note 4)
Events attributes			
	onclick	R1	
	ondbclick	–	
	onmousedown	–	
	onmouseup	–	
	onmouseover	–	
	onmousemove	–	
	onmouseout	–	
	onkeypress	–	
	onkeydown	R1	
	onkeyup	R1	
Style attributes			
	style	R1	
Core modules Structure module			
body	%Common.attrib		
	%Core.attrib	R1	
	%I18n.attrib	R2	
	%Events.attrib	–	
head	%I18n.attrib	R2	
	profile	–	
title	%I18n.attrib	R2	
Text module			
br	%Core.attrib	R1	
div	%Common.attrib	R1	
p	%Common.attrib	R1	
span	%Common.attrib	R1	
Hypertext module			
a	%Common.attrib	R1	
	accesskey	R1	

Table 7-1 – HTML attributes used in a LIME-HTML document

Elements	Attributes	Operation	Restrictions for operation
	charset	R2	(Note 5)
	href	R1	
	hreflang	–	
	rel	–	
	rev	–	
	tabindex	–	
	type	–	
Forms module			
input	%Common.attrib	–	
	%Core.attrib	R1	
	%I18n.attrib	R2	
	%Events.attrib	R1	Cannot be specified when "inputmode attribute" is "direct" or "indirect"
	accesskey	R1	
	checked	–	
	disabled	R1	
	readonly	R1	
	maxlength	R1	From 1 to 40 (Note 1)
	alt		
	name	–	
	size	–	
	src	–	
	tabindex	–	
	accept	–	
	type	R1	Either "text" or "password"
	value	R1	
	inputmode	R1	
	charctertype	R1	
Client-side image map			
a&	cords	–	
	shape	–	
input&	usemap	–	
object&	usemap	–	
Server-side image map			
input&	ismap	–	
Object module			
object	%Common.attrib	R1	
	archive	–	

Table 7-1 – HTML attributes used in a LIME-HTML document

Elements	Attributes	Operation	Restrictions for operation
	classid	–	
	codebase	–	
	codetype	–	
	data	R1	
	declare	–	
	height	–	
	name	–	
	standby	–	
	tabindex	–	
	type	R1	
	width	–	
Target module			
a&	target	–	
Intrinsic events module			
a&	onblur	R1	
	onfocus	R1	
body&	onload	R1	
	onunload	R1	(Note 3)
input&	onfocus	R1	
	onblur	R1	
	onselect	–	
	onchange	R1	(Note 2)
Metainformation module			
meta	%I18n.attrib	R2	
	http-equiv	–	
	name	R1	
	content	R1	
	scheme	–	
Scripting module			
script	charset	R2	(Note 5)
	type	R2	(Note 6)
	src	R1	
	defer	–	
	xml.space	–	
Style sheet module			
style	%I18n.attrib	R2	
	type	R2	Fixed to "text/css"
	media	R2	Fixed to "tv"

Table 7-1 – HTML attributes used in a LIME-HTML document

Elements	Attributes	Operation	Restrictions for operation
	title	–	
	xml:space	–	
Link module			
link	%Common.attrib	–	
	charset	R2	(Note 5)
	href	R1	
	hreflang	–	
	media	R2	Fixed to "tv"
	rel	R2	Fixed to "stylesheet"
	rev	–	
	type	R2	Fixed to "text/css"
LIME module			
bml	%I18n.attrib	R2	
	version	–	
	xmlns	–	
bevent	id	R1	
beitem	id	R1	
	type	R1	One of the following is taken: "TimerFired", "CCStatusChanged", "MediaStopped", "DataButtonPressed"
	onoccur	R1	
	es_ref	R1	
	message_group_id	R1	It is "0" or "1". When omitted, specification of "0" is assumed.
	message_id	R1	
	message_version	R1	
	module_ref	R1	
	language_tag	R1	
	register_id	–	
	service_id	–	
	event_id	–	
	peripheral_ref	–	
	time_mode	R1	The following is taken: "absolute"
	time_value		
	object_id	R1	Only the object element ID that indicates data transmitted by carousel and type attribute is either "audio/X-arib-mpeg2-aac"
	subscribe	R1	

Table 7-1 – HTML attributes used in a LIME-HTML document

Elements	Attributes	Operation	Restrictions for operation
iframe&	align	–	
body&	invisible	R1	
div&	accesskey	R1	
	onfocus	R1	
	onblur	R1	
p&	accesskey	R1	
	onfocus	R1	
	onblur	R1	
span&	accesskey	R1	
	onfocus	R1	
	onblur	R1	
a&	effect	–	
bdo&	orientation	–	
object&	streamposition	R1	The frame number is specified (type="image/X-arib-mng") when the monomedia that refers to the relevant object element is MNG. In case of other media, it is "0".
	streamlooping	R2	Fixed to "1"
	streampositionnumerator	–	
	streampositiondenominator	–	
	streamstatus	R1	An initial value must be specified depending on the monomedia referenced by the relevant object element (Note 7)
	streamlevel	–	
	remain	R1	Applicability depends on the monomedia referenced by the object element (Note 8)
	accesskey	R1	
	onfocus	R1	
	onblur	R1	

Table 7-1 – HTML attributes used in LIME-HTML document

NOTE 1 – When the input exceeds the maximum length, then it is rounded down. If it goes beyond the frame, the exceeded part will not be displayed.

NOTE 2 – Generated timing of the change event is when the focus is shifted to a different element.

NOTE 3 – The only extended functions for broadcasting that can be used in the "onunload" event handler are "writePersistentArray()" and "unlockModuleOnMemory()". Processing contents should be limited to processes that end in a short time, such as set-up to Ureg, where quick document transition is possible and processes for simple status checking, etc.

NOTE 4 – This is currently fixed to "ja" in Japan.

NOTE 5 – This is currently fixed to "EUC-JP" in Japan. Other codings are for further study.

NOTE 6 – This is currently fixed to "text/X-arib-ecmascript"; charset="euc-jp"". Other codings are for further study.

NOTE 7 – The values of streamstatus for an object referencing media of type attribute are summarized as follows (other type attributes are for further study):

type attribute	streamstatus
video/X-arib-mpeg2	play (initial value: play)
video/X-arib-mpeg1	play (initial value: play)
audio/X-arib-mpeg2-aac	play, stop (initial value: stop)
image/jpeg	play (initial value: play)
image/X-arib-png	play (initial value: play)
image/X-arib-mng	play, stop, pause (initial value: stop)
text/X-arib-jis8text	play (initial value: play)

NOTE 8 – Applicability of the remaining attribute depends on the type attribute according to the following list (other type attributes are for further study):

type attribute	applicability
video/X-arib-mpeg2	yes
video/X-arib-mpeg1	yes
audio/X-arib-mpeg2-aac	only when the scheme is "arib: (PES)"
image/jpeg	yes (lockModuleOnMemory is also used)
image/X-arib-png	no
image/X-arib-mng	no
text/X-arib-jis8text	no

7.1.5 used-key-list

This Recommendation specifies the use of used-key-list features as indicated in Tables 7-2 through 7-4.

Table 7-2 – used-key-list

Items	Features
Value of <key-group> (Note 1)	"special-1" is used for video on demand (VoD) playback control key group (Note 3)
Key code (Note 2)	To be eventually added in the vendor-dependent keys (key code 150-)
Access key characters (Note 2)	Not specified (Note 2)

Table 7-2 – used-key-list

Items	Features
Behaviour	When an LIME document contains a description of playback control procedure, it is desirable to mask "special-1", the VOD playback control <key-group>, to avoid confusion of the user. When masked, events from the VoD playback control keys are received by the LIME browser. Since key codes depend on implementation, it is not recommended that such key information is included in LIME content.
NOTE 1 – Refer to Table 7-3. NOTE 2 – Refer to Table 7-4. NOTE 3 – This refers to keys for, e.g., playback, stop, rewind, fast forward, chapter-jump. These are software keys implemented in the on-screen display (OSD), and whether this is implemented as physical keys or not depends on each implementation.	

Table 7-3 – Values Applicable to <key-group>

Value of <key-group>	Semantics	
Basic	Up, down, right and left arrow keys, enter key and back key	
data-button	Keys for data broadcasting operations (e.g., red, green, blue and yellow colour keys)	(Note 1)
numeric-tuning	Channel keypad (0 to 9, or 0 to 12)	(Note 2)
Other-tuning	Other channel keys (e.g., up/down and direct selection)	(Note 2)
special-1	Special key 1	(Note 3)
special-2	Special key 2	(Note 3)
special-3	Special key 3	(Note 3)
special-4	Special key 4	(Note 3)
Misc	Keys except the above keys and power key (e.g., volume control keys)	(Note 4)
NOTE 1 – Additional keys for data broadcasting services, as needed, are specified in an operational standard regulation. NOTE 2 – Actual usage of channel keys is specified in an operational standard regulation or optionally implemented by the vendor. NOTE 3 – The broadcaster specifies this key for each medium. NOTE 4 – Any receiver must provide a power key. Masking of the power key by the LIME content is not allowed.		

Table 7-4 – Relationship among remote control keys, key codes and access keys

Remote control key	Key code	Access key character
Up arrow	1	N/A
Down arrow	2	N/A
Left arrow	3	N/A
Right arrow	4	N/A

Table 7-4 – Relationship among remote control keys, key codes and access keys

Remote control key	Key code	Access key character
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	5-17	N/A
"Enter"	18	N/A
"Back"	19	"X"
"Data" (Note 1)	20	N/A
Colour key (blue)	21	"B"
Colour key (red)	22	"R"
Colour key (green)	23	"G"
Colour key (yellow)	24	"Y"
Data button 1	25	"E"
Data button 2	26	"F"
Data button 3	27	N/A
Data button 4	28	N/A
Reserved for ARIB data broadcast standard	29-99	Reserved
"Bookmark" key (Note 2)	100	N/A
Reserved for future extended features	101-149	Reserved
Vendor-dependent	150-	Not defined
NOTE 1 – An event DataButtonPressed occurs and no keydown nor keyup event occurs. NOTE 2 – Implementing a "bookmark" key is optional.		

7.1.6 Media types used in LIME

Media types are required to comply with Table 7-5 and the following points.

CSS data (media type "text/css") may appear in LIME documents in some cases and may be transmitted as independent resources in other cases. CSS data transmitted as monomedia is required to be a complete description of the style sheet, as defined by CSS, on its own.

LIME-Script data may appear in LIME HTML documents in some cases and may be transmitted as independent resources in other cases. LIME-Script data transmitted as monomedia is required to be a complete description of the scripting, as defined by LIME-Script, on its own.

Table 7-5 – List of media types and monomedia schemes

Scheme	Media type	Used	Operation (referenced by object/ element)	Remarks
http:, https:	multipart/mixed	Yes	—	
	text/css	Yes	—	
	text/X-arib-bml; charset=" "	Yes	—	(Note 1)
	text/X-arib-ecmascript; charset=" "	Yes	—	(Note 1)
	image/jpeg	Yes	Yes	
	image/X-arib-png	Yes	Yes	

Table 7-5 – List of media types and monomedia schemes

Scheme	Media type	Used	Operation (referenced by object/ element)	Remarks
	image/X-arib-mng	Yes	Yes	
	audio/X-arib-mpeg2-aac	Yes	Yes	
	application/X-arib-bmlclut	Yes	—	
	application/X-arib-btable	Yes	—	
	application/X-arib-resourceList	Yes	—	
	application/X-arib-contentPlayControl	Yes	Yes	(Note 2)
Arib:	application/X-arib-mpeg2-tts	Yes	Yes	
romsound:	audio/X-arib-romsound	Yes	—	
NOTE 1 – The "charset" specification cannot be omitted. NOTE 2 – "application/X-arib-contentPlayControl" is the media type assigned to the metafile used for VoD streaming playback control. The server specifies this media type in the Content-Type of the HTTP message header so that the IPTV terminal device can identify the metafile.				

7.2 Display control of LIME

7.2.1 Display control of linear IPTV streaming

Table 7-6 shows the guideline for the use of "type", "streamposition", "streamstatus" and "streamlooping" attributes when presenting the audio and video of linear IPTV streaming content as part of an "object" element.

Table 7-6 – Use of attribute for displaying stream

type attribute	streamposition	streamstatus	streamlooping
application/X-arib-mpeg2-tts (Note 1)	Not used	play/stop (Notes 2, 3)	1 (fixed)
NOTE 1 – The linear IPTV service is expected to be specified in the "data" attribute using the namespace. NOTE 2 – The dynamic change of "type" attribute and of schema following the change of "data" attribute is not expected. NOTE 3 – In this Recommendation, the initial value of streamstatus for this type attribute is "play".			

7.2.2 Display control of VoD streaming

Table 7-7 shows the guideline for the use of "type", "streamposition", "streamstatus", and "streamlooping" attributes when presenting the audio and video of VoD streaming content as part of an "object" element.

Table 7-7 – Use of attribute for displaying stream

type attribute	streamposition	streamstatus	streamlooping
application/X-arib-contentPlayControl (Note 1)	Read only (Note 4)	play/stop/pause (Notes 2, 3, 5, 6)	1 (fixed)
NOTE 1 – Metadata file for stream playback control is required to be specified in the "data" attribute. NOTE 2 – The dynamic change of the "type" attribute and change of schema due to a change of the "data" attribute are not allowed. NOTE 3 – When the playback ends, the streamstatus changes to "stop" automatically. NOTE 4 – As an exception, the initial value of "streamposition" attribute of the "object" element in a LIME document can optionally be specified. NOTE 5 – When the state of the media player changes, the value of streamstatus must automatically change. The timing for this automatic change is implementation dependent. NOTE 6 – The initial value of streamstatus is "play".			

8 Use of LIME-CSS in LIME

Table 8-1 shows the LIME-CSS profile which includes CSS 1 and CSS 2 properties.

Table 8-1 – Profile of CSS properties in LIME

Property	Operation
Selector	
*	R1
E	R1
EF	–
E:focus	R1
E:active	R1
E:myclass	R1
#myid	R1
Value assignment/inheritance	
@import	–
!important	–
Media type	
@media	R1
Box model	
margin-top	–
margin-right	–
margin-bottom	–
margin-left	–
Margin	R1
padding-top	R1
padding-right	R1
padding-bottom	R1
Padding-left	R1

Property	Operation
Visual formatting model	
position	R1
left	R1
top	R1
width	R1
height	R1
z-index	R1
line-height	R1
vertical-align	–
display	R1
bottom	–
right	–
float	–
clear	–
direction	–
unicode-bidi	–
min-width	–
max-width	–
min-height	–
max-height	–
Other visual effects	
visibility	R1
overflow	R1

Table 8-1 – Profile of CSS properties in LIME

Property	Operation
Padding	–
border-top-width	–
border-right-width	–
border-bottom-width	–
border-left-width	–
border-width	R1
border-top-color	–
border-right-color	–
border-bottom-color	–
border-left-color	–
border-color	–
border-top-style	–
border-right-style	–
border-bottom-style	–
border-left-style	–
border-style	R1
border-top	–
border-right	–
border-bottom	–
border-left	–
Border	-
Background	
background	–
background-color	–
background-image	R1
background-repeat	R1
background-position	–
background-attachment	–
Font	
color	–
font-family	R1
font-style	–
font-size	R1
font-variant	–
font-weight	R1
font	–
font-stretch	–
font-size-adjust	–

Property	Operation
clip	–
Generated content/auto numbering list	
content	–
quotes	–
counter-reset	–
counter-increment	–
marker-offset	–
list-style-type	–
list-style-image	–
list-style-position	–
list-style	
Page media	
"@page"	–
size	–
marks	–
page-break-before	–
page-break-after	–
page-break-inside	–
page	–
orphans	–
widows	–
User interface	
outline-color	–
outline-width	–
outline-style	–
outline	–
cursor	–
Voice style sheet	
volume	–
speak	–
pause-before	–
pause-after	–
Pause	–
cue-before	–
cue-after	–
cue	–
play-during	–
azimuth	–

Table 8-1 – Profile of CSS properties in LIME

Property	Operation
Text	
text-indent	–
text-align	R1
text-decoration	–
text-shadow	–
letter-spacing	R1
word-spacing	–
text-transform	–
white-space	R1
Pseudo class/pseudo element	
:link	–
:visited	–
:active	R1
:hover	–
:focus	R1
:lang	–
:first-child	–
:first-line	–
:first-letter	–
:before	–
:after	–
Table	
caption-side	–
border-collapse	–
border-spacing	–
table-layout	–
empty-cells	–
speak-header	–

Property	Operation
elevation	–
speech-range	–
voice-family	–
pitch	–
pitch-range	–
stress	–
richness	–
speak-punctuation	–
peak-numeral	–
Extended property	
clut	R1
color-index	R1
background-color-index	R1
border-color-index	–
border-top-color-index	R1
border-right-color-index	R1
border-bottom-color-index	R1
border-left-color-index	R1
outline-color-index	–
resolution	R1
display-aspect-ratio	R1
grayscale-color-index	R1
nav-index	R1
nav-up	R1
nav-down	R1
nav-left	R1
nav-right	R1
used-key-list	R1

A value defined as a fixed value should be specified as the most important rule (`!important`) in the default style sheet. This most important rule always takes priority over the normal rules. This most import rule is not operated in the LIME-CSS and the normal rules are operated. Therefore, the value specified as a fixed value is always obtained.

9 Use of LIME-Script

This clause describes the use of LIME-Script in LIME.

9.1 Profile of built-in objects

Table 9-1 is the profile of built-in objects for LIME-Script.

Table 9-1 – Profile of the LIME-Script built-in objects

Built-in object	Method/property	Operation	Remarks
(global)			
	NaN	R1	
	Infinity	–	(Note 1)
	eval(x)	–	
	parseInt(string, radix)	R1	(Note 7)
	parseFloat(string)	–	(Note 1)
	escape(string)	–	(Note 2)
	unescape(string)	–	(Note 2)
	isNaN(number)	R1	
	isFinite(number)	–	(Note 1)
Object			
	prototype	R1	
	Object([value])	R1	
	new Object([value])	R1	
Object.prototype			
	constructor	R1	
	toString()	R1	
	valueOf()	R1	
Function			
	prototype	R1	
	Length	R1	
	Function(p1,p2,…pn,body)	–	
	new Function(p1,p2,…pn,body)	–	
Function.prototype			
	constructor	R1	
	toString()	R1	(Note 3)
Array			
	prototype	R1	
	Length	R1	
	Array(item0, item1, …)	R1	
	new Array(item0, item1,…)	R1	
	new Array([len])	R1	
Array.prototype			
	constructor	R1	

Table 9-1 – Profile of the LIME-Script built-in objects

Built-in object	Method/property	Operation	Remarks
	toString()	R1	
	join([separator])	R1	
	reverse()	R1	
	sort([comparefn])	R1	
String			
	prototype	R1	
	Length	R1	
	String([value])	R1	
	new String([value])	R1	
	String.fromCharCode(char0[,char1, …])	R1	
String.prototype			
	constructor	R1	
	toString()	R1	
	valueOf()	R1	
	charAt(pos)	R1	
	charCodeAt(pos)	R1	
	indexOf(searchString, position)	R1	
	lastIndexOf(searchString, position)	R1	
	split(separator)	R1	
	substring(start[,end])	R1	
	toLowerCase()	R1	
	toUpperCase()	R1	
Boolean			
	prototype	R1	
	Boolean([value])	R1	
	new Boolean([value])	R1	
Boolean.prototype			
	constructor	R1	
	toString()	R1	
	valueOf()	R1	
Number			
	prototype	R1	
	MAX_VALUE	R1	
	MIN_VALUE	R1	
	NaN	R1	
	NEGATIVE_INFINITY	–	(Note 1)
	POSITIVE_INFINITY	–	(Note 1)
	Number([value])	R1	
	new Number([value])	R1	

Table 9-1 – Profile of the LIME-Script built-in objects

Built-in object	Method/property	Operation	Remarks
Number.prototype			
	constructor	R1	
	toString([radix])	R1	
	valueOf()	R1	
Math			
	E	–	
	LN10	–	
	LN2	–	
	LOG 2E	–	
	LOG 10E	–	
	PI	–	
	SQRT1 2	–	
	SQRT2	–	
	abs(x)	–	
	acos(x)	–	
	asin(x)	–	
	atan(x)	–	
	atan2(y, x)	–	
	cos(x)	–	
	exp(x)	–	
	floor(x)	–	
	log(x)	–	
	max(x, y)	–	
	min(x, y)	–	
	pow(x, y)	–	
	random()	–	
	round(x)	–	
	sin(x)	–	
	sqrt(x)	–	
	tan(x)	–	
Date			
	prototype	R1	
	Date([year, month [, date [, hours [, minutes [, seconds [, ms]]]]]])	R1	
	new Date([year, month [, date [, hours [, minutes [, seconds [, ms]]]]]])	R1	
	Date(value)	–	(Note 4)
	new Date(value)	–	(Note 4)
	Date.parse(string)	–	(Note 4)

Table 9-1 – Profile of the LIME-Script built-in objects

Built-in object	Method/property	Operation	Remarks
	Date.UTC([year, month [, date [, hours [, minutes [, seconds [, ms]]]]]])	–	(Note 4)
Date.prototype			
	constructor	R1	
	toString()	R1	(Note 3)
	valueOf()	–	(Note 4)
	getTime()	–	(Note 4)
	getYear()	–	(Note 5)
	getFullYear()	R1	
	getUTCFullYear()	R1	
	getMonth()	R1	
	getUTCMonth()	R1	
	getDate()	R1	
	getUTCDate()	R1	
	getDay()	R1	
	getUTCDay()	R1	
	getHours()	R1	
	getUTCHours()	R1	
	getMinutes()	R1	
	getUTCMinutes()	R1	
	getSeconds()	R1	
	getUTCSeconds()	R1	
	getMilliseconds()	R1	
	getUTCMilliseconds()	R1	
	getTimezoneOffset()	R1	
	setTime(time)	–	(Note 4)
	setMilliseconds(ms)	R1	(Note 6)
	setUTCMilliseconds(ms)	R1	(Note 6)
	setSeconds(sec, [, ms])	R1	(Note 6)
	setUTCSeconds(sec, [, ms])	R1	(Note 6)
	setMinutes(min [, sec, [, ms]])	R1	(Note 6)
	setUTCMinutes(min [, sec, [, ms]])	R1	(Note 6)
	setHours(hour [, min [, sec, [, ms]]])	R1	(Note 6)
	setUTCHours(hour [, min [, sec, [, ms]]])	R1	(Note 6)
	setDate(date)	R1	(Note 6)
	setMonth(mon [, date])	R1	(Note 6)
	setUTCMonth(mon [, date])	R1	(Note 6)
	setFullYear(year [, mon [, date]])	R1	(Note 6)
	setUTCFullYear(year [, mon [, date]])	R1	(Note 6)

Table 9-1 – Profile of the LIME-Script built-in objects

Built-in object	Method/property	Operation	Remarks
	setYear(year)	–	(Note 5)
	toLocaleString()	R1	(Note 3)
	toUTCString()	R1	(Note 3)
	toGMTString()	–	(Note 5)
NOTE 1 – Not operated because it is related to Float. NOTE 2 – Not operated because it is related to Unicode. NOTE 3 – Result of Function.prototype.toString() ([b-ISO/IEC 16262], page 69) "function FUNCTIONNAME() {}"(FUNCTIONNAME is the name of a specified function). Results of Date.prototype.toLocaleString() and Date.prototype.toUTCString() must be of the same output format as Date.prototype.toString() NOTE 4 – Not operated because it is related to Number. NOTE 5 – Not operated because it is specified to maintain compatibility with old source codes. NOTE 6 – Operated with restricted specification because it is related to Number. NOTE 7 – The radix of parseInt() is 8, 10 and 16 (0 is interpreted as 10).			

9.2 Extensions to ECMAScript

Table 9-2 describes the browser pseudo-object profile of LIME-Script. This object provides the interfaces for the following functions:

1) EPG: The function to tune from EPG.
2) Non-volatile memory: The function to read and to write in the persistent array.
3) Bidirectional function over TCP/IP: The function to transmit text-data over IP and to set cache resources over IP.

Table 9-2 – Browser pseudo-object of LIME-Script

	Function	Operation	Remarks
EPG functions			
	epgGetEventStartTime()	R1	
	epgGetEventDuration()	R1	
	epgTune()	R1	
	epgTuneToDocument()	R1	
	epgIsReserved()	R1	
	epgReserve()	R1	
	epgCancelReservation()	R1	
	epgRecIsReserved()	R1	
	epgRecReserve()	R1	
	epgRecCancelReservation()	R1	
Interaction channel communication – TCP/IP			
	setISPParams()	R1	
	getISPParams()	R1	
	connectPPP()	R1	

Table 9-2 – Browser pseudo-object of LIME-Script

	Function	Operation	Remarks
	connectPPPWithISPParams()	R1	
	disconnectPPP()	R1	
	getConnectionType()	R1	
	isIPConnected()	R1	
	sendTextMail()	Optional	
	sendMIMEMail()	Optional	
	transmitTextDataOverIP()	R1	
	setCacheResourceOverIP()	Optional	
Operational control functions			
	reloadActiveDocument	R1	
	getNPT()	R1	
	getProgramRelativeTime()	R1	
	isBeingBroadcast()	R1	
	lockModuleOnMemory()	R1	
	unlockModuleOnMemory()	R1	
	setCachePriority()	R1	
	getIRDID()	R1	
	getBrowserVersion()	R1	
	getProgramID()	R1	
	getActiveDocument()	R1	
	lockScreen()	R1	
	unlockScreen()	R1	
	getBrowserSupport()	R1	
	launchDocument()	R1	
	launchDocumentRestricted()	R1	
	quitDocument()	R1	
	launchExApp()	Optional	(Note)
	getFreeContentsMemory()	R1	
	isSupportedMedia()	R1	
	detectComponent()	R1	
	lockModuleOnMemoryEx()	R1	
	unlockModuleOnMemoryEx()	R1	
	unlockAllModulesOnMemory()	R1	
	getLockedModuleInfo()	R1	
	getBrowerStatus()	R1	
	getResidentAppVersion()	R1	
	isRootCertificateExisting()	R1	
	getRootCertifiacteInfo()	R1	
	startResidentApp()	Optional	

Table 9-2 – Browser pseudo-object of LIME-Script

	Function	Operation	Remarks
Receiver audio control			
	playRomSound()	R1	
Timer functions			
	sleep()	R1	
	setInterval()	R1	
	clearTimer()	R1	
	pauseTimer()	R1	
	resumeTimer()	R1	
	setCurrentDateMode()	R1	
External character functions			
	loadDRCS()	R1	
Other functions			
	random()	R1	
	subDate()	R1	
	addDate()	R1	
	formatNumber()	R1	
Closed caption display control functions			
	setCCDisplayStatus()	R1	
	getCCDisplayStatus()	R1	
	getCCLanguageStatus()	R1	
NOTE – Even when using independent services.			

9.2.1 Explanations of the methods

`epgGetEventStartTime()`

Obtains the start time of a program described in EIT.

Syntax

```
Date epgGetEventStartTime(input String event_ref)
```

Argument

event_ref: Specifies an event

Return values

Start time of a program: Success

null: Could not obtain the event information specified by event_ref.

Description

The description of event_ref conforms to the conventions defined in clause 9.2.1.1.

`epgGetEventDuration()`

Obtains the duration time of a program described in EIT.

Syntax

```
Number epgGetEventDuration(input String event_ref)
```

Argument

event_ref: Specifies an event.

Return values

Duration time of a program (in seconds): Success

NaN: Obtained no event information, as specified by event_ref.

Description

The description of event_ref conforms to the conventions defined in clause 9.2.1.1.

`epgTune()`

Quits displaying the presented LIME document and selects a specified service.

Syntax

```
Number epgTune(input String service_ref)
```

Arguments

service_ref: Specifies a service.

Return values

1: Success

NaN: Failure

Description

The description of event_ref conforms to the conventions defined in clause 9.2.1.1.

– The scripts after the epgTune() are not executed.
– If epgTune() is executed in a global code, neither the "load" event nor the "unload" event occurs.
– If epgTrue() fails, it is not ensured that the following scripts are executed.

`epgTuneToDocument()`

Quits displaying the presented LIME document and presents the specified LIME document.

Syntax

```
Number epgTuneToDocument(input String documentName)
```

Argument

DocumentName: String specifying the LIME document to be presented.

Return values

1: Success

NaN: Failure

Description

The description of documentName conforms to the conventions defined in clause 9.2.1.2. This object selects the service transmitting the LIME document specified in documentName and presents the specified LIME document.

– The scripts following epgTuneToDocument() are not executed.

– If epgTuneToDocument() is executed in the global code, neither the "load" event nor the "unload" event occurs.

– If epgTuneToDocument() fails, it is not ensured that the following scripts are executed.

`epgIsReserved()`

Verifies whether or not the specified event is reserved for watching.

Syntax

```
Number epgIsReserved(input String event_ref
[,input Date startTime]
)
```

Arguments

event_ref: Specifies an event.

startTime: Start time of an event.

Return values

1: Reserved for watching.

0: Not reserved.

NaN: Failure.

Description

This object verifies whether the event designated by the event_ref which is scheduled to start at the time designated by startTime is reserved for watching or not. The investigation result is returned by the value.

The description of event_ref conforms to the conventions defined in clause 9.2.1.1. If startTime is omitted, this function acts on the event specified by event_ref.

`epgReserve()`

Reserves a specified event for watching.

Syntax

```
Number epgReserve(input String event_ref [,input Date startTime])
```

Argument

event_ref: Specifies an event.

startTime: Start time of an event.

Return values

1: Success.

NaN: Failure.

Description

This object reserves the event designated by the event_ref for watching which is scheduled to start at the time designated by startTime. Success or failure is returned by the value. The description of event_ref conforms to the conventions defined in clause 9.2.1.1. If startTime is omitted, this function acts on the event specified byevent_ref.

`epgCancelReservation()`

Cancels the reservation for watching of a specified event.

Syntax

```
Number epgCancelReservation(input String event_ref)
```

Argument

event_ref: Specifies an event.

Return values

1: Success.

NaN: Failure.

Description

This object cancels the watching reservation of the event designated at event_ref. Success or failure is returned by the value. The description of event_ref conforms to the conventions defined in clause 9.2.1.1. If startTime is omitted, this function acts on the event specified by event_ref.

`epgRecIsReserved()`

Verifies whether or not a specified event has been reserved for recording.

Syntax

```
Number epgRecIsReserved(input String event_ref [,input Date startTime])
```

Arguments

event_ref: Specifies an event.

startTime: Start time of an event.

Return values

1: Reserved for recording.

0: Not reserved.

NaN: Failure.

Description

This object whether or not the event designated by the event_ref which is scheduled to start at the time designated by startTime is reserved for recording. The result is returned by the return value. The description of event_ref conforms to the conventions defined in clause 9.2.1.1. If startTime is omitted, this function acts on the event specified by event_ref.

`epgRecReserve()`

Reserves a specified event for recording.

Syntax

```
Number epgRecReserve(input String event_ref [,input Date startTime])
```

Arguments

event_ref: Specifies an event.

startTime: Start time of an event.

Return values

1: Success.

NaN: Failure.

Description

This object reserves the event designated by the event_ref for recording which is scheduled to start at the time designated by startTime. Success or failure is returned by the value. The description of event_ref conforms to the conventions defined in clause 9.2.1.1. If startTime is omitted, this function acts on the event specified by event_ref.

`epgRecCancelReservation()`

Cancels the reservation for recording of a specified event.

Syntax

```
Number epgRecCancelReservation(input String event_ref)
```

Argument

event_ref: Specifies an event.

Return values

1: Success.

NaN: Failure.

Description

This object cancels the reservation for recording of an event specified by event_ref and returns the result of cancellation. The description of event_ref conforms to the conventions defined in clause 9.2.1.1.

`random()`

Generates random numbers.

Syntax

```
Number random(input Number num)
```

Arguments

`num`: Upper limit of random numbers.

Return values

Random number.

Description

This function returns integer random numbers in a range from 1 to num. Pseudo random numbers are acceptable, but they must generate uniform random numbers. The argument of `random()` is a natural number.

`subDate()`

Calculates the time difference between two dates in a specified unit.

Syntax

```
Number subDate(
input Date target,
input Date base,
input Number unit
)
```

Arguments

`target`: Subtracted Date object

`base`: Subtracting Date object

`unit`: Unit of calculation 0: milliseconds, 1: seconds, 2: minutes, 3: hours, 4: days, 5: weeks

Return values

Time difference in the specified unit: success

NaN: failure

Description

This function subtracts base from target and returns the result in a unit of time specified in unit. The fraction is truncated. The result is guaranteed to be handled as a signed 32-bit integer. If the result is in the range from –2147483648 to 2147483647 (maximum range of a signed 32 bit integer), it is returned as it is. If the result is out of this range, NaN is returned. (Note: If unit is '0' (milliseconds), the effective range is from –24 to 24 days.) If unit is an invalid value, it is treated as '0' (zero).

`addDate()`

Add time in a specified unit to a specified Date object.

Syntax

```
Date addDate(
input Date base,
input Number time,
input Number unit
)
```

Arguments

base: Base Date object.

time: Time to be added.

unit: Unit of time (0: milliseconds, 1: seconds, 2: minutes, 3: hours, 4: days, 5: weeks)

Return values

A Date object that indicates the result of addition: success.

NaN: failure.

Description

This function adds time in the unit specified by unit to a base and returns the result. The base does not change.

If time is NaN, base itself is returned.

If unit is an invalid value, it is treated as '0' (zero).

`formatNumber()`

Formats a numeric value by inserting "," every three digits and returns the result as a character string.

Syntax

```
String formatNumber( input Number value )
```

Argument

value: Numeric value to be formatted and converted into a character string.

Return values

Formatted character string: Success.

null: Failure

Description

This function formats a numeric value by inserting "," every three digits and returns the result as a character string. For example, it is used to format monetary values. If value is an invalid value, it is treated as '0' (zero).

`reloadActiveDocument()`

Reloads a LIME document that is currently displayed.

Syntax

```
Number reloadActiveDocument()
```

Arguments

None.

Return values

NaN

Description

This function reloads a document that is currently displayed. The `reloadActiveDocument()` acts as the same as `launchDocument()` to itself. If `reloadActiveDocument()` fails, it is not ensured that the following scripts are executed.

`getNPT()`

Obtains an NPT.

Syntax

```
Number getNPT()
```

Argument

None.

Return values

Time specified by NPT: Success

NaN: Failure

Description

This function obtains an NPT value for a stream calculated from the NPT reference descriptor. The return value is an integer in milliseconds.

`getProgramRelativeTime()`

Obtains a relative time from the beginning of the event.

Syntax

```
Number getProgramRelativeTime()
```

Argument

None.

Return values

Non-negative integer: Relative time from the beginning of the event.

NaN: Failure

Description

This function returns the relative time (in seconds) from the beginning of the event that is being watched.

`isBeingBroadcast()`

Verifies whether or not a specified event (broadcast program) is currently broadcast.

Syntax

```
Boolean isBeingBroadcast(input String event_ref)
```

Arguments

event_ref: Specifies an event.

Return values

false: Currently not broadcast.

true: Currently broadcast.

Description

The description of event_ref conforms to the namespace conventions defined in clause 9.2.1.1 It is not ensured that the function verifies whether or not a stored program is currently played.

`lockModuleOnMemory()`

Receives a module into cache memory and locks the module.

Syntax

```
Number lockModuleOnMemory(input String module)
```

Argument

module: Module name.

Return values

NaN: Failure because of other causes.

1: Success.

–1: Specified module does not exist.

–2: Cannot receive because of insufficient cash.

NOTE – When a return value is 1, –1, or –2, the state can be confirmed using DII.

Description

This function receives any module which was transmitted in a same component of the module specified with module (data other than contents data is allowed) from the carousel and lock it in the content memory. The contents module is locked in cache memory until unlockModuleOnMemory() or unlockAllModuleOnMemory() is called, or the Multimedia Service ends. The description of module conforms to the conventions on namespace defined in clause 9.2.1.2. This function exits without waiting for the module to be actually obtained. When the module is actually obtained, ModuleLocked specified with event occurs. If unlockModuleOnMemory() or unlockAllModulesOnMemory() is invoked while this function tries to lock a module that has not been locked in the content memory, the request to lock the module is cancelled. If unlockModuleOnMemory() or unlockAllModulesOnMemory() is invoked to unlock a module which has not been locked in the content memory and on which lockModuleOnMemory() is not working, an error is returned. The function returns the result of processing as a returned value

`unlockModuleOnMemory()`

Unlocks a locked module.

Syntax

```
Number unlockModuleOnMemory(input String module)
```

Argument

module: Module name.

Return values

1: Success.

NaN: Failure.

Description

This function unlocks a module specified with module (data other than contents data is allowed) to release it from the content memory. If the module has not been locked in the content memory by lockModuleOnMemory(), the execution of this function fails. The description of module conforms to the conventions on namespace defined in clause 9.2.1.2.

`setCachePriority()`

Sets a cache priority of a module.

Syntax

```
Number setCachePriority(
input String module,
input Number priority
)
```

Arguments

`module`: Module name.

`priority`: Cache priority.

Return values

1: Success.

NaN: Failure.

Description

This function assigns a cache priority specified with priority to a module specified with module (data other than contents data is allowed). The larger the value of priority, the higher the cache priority. The description of module conforms to the conventions on namespace defined in clause 9.2.1.2.

`getIRDID()`

Obtains a receiver ID(identifier).

Syntax

```
String getIRDID (input Number type)
```

Arguments

`type`: Type of ID to obtain.

Return values

Identifier specific to receiver: Success

null: Failure.

Description

This function returns ID that is specific to the receiver specified in type. If the function failed to obtain the ID, it returns null. The following is applicable to type:

1) CardID of CA

CardID is used to support a multiple transport receiver. A separate type argument is specified for each CA system. The CA_system_id identification is used as the value of the type argument. In this case, a returned value is a hexadecimal string consisting of six hexadecimal numbers and twelve characters for zero-padding. Each hexadecimal number is obtained by converting each byte of the 6-byte CardID into a hexadecimal representation.

2) Receiver ID

Receiver ID is used to recognize a receiver as hardware. Receiver ID must not be the same as CA_system_id. Detailed usage of Receiver ID is defined in an operational standard regulation.

3) MakerID and ModelID

MakerID and ModelID are used for downloading. MakerID and ModelID must not be the same as CA_system_id. Detailed usage of these IDs is defined in an operational standard regulation.

`getBrowserVersion()`

Obtains information to identify a LIME browser.

Syntax

```
Array getBrowserVersion()
```

Argument

None.

Return values

Array[0]: String representing MakerID.

Array[1]: String representing the name of LIME browser.

Array[2]: String representing the major version number.

Array[3]: String representing the minor version number.

Description

This function obtains the information to identify the LIME browser that controls presentation of the currently displayed LIME document. Array[0] contains the string representing MakerID used for downloading software for the receiver. Any string contained in Array[0] is a two-digit hexadecimal representation. Note that this string does not have to be explicitly marked as a hexadecimal representation. That is, this string does not have to be preceded with "0x" nor be followed by "h". Instead, this string requires "0" for padding to form a two-digit representation.

Array[1] contains a string that is not more than 20-character long. This string a combination of the "0"-" 9" and " A"-" Z" alphanumeric to identify a manufacturer. Array[2] and Array[3] contain a string that is a three-digit decimal representation consisting of a version number, as specified by a manufacturer and "0"s as required for padding. Note: Updating major/minor version numbers is responsible for vendors of receivers. However, it is recommended that any modification or change in a LIME browser causes a minor version number to be updated. It is also recommended that when different types of receivers use a same version of LIME browser, the same major/minor version number is returned.

`getProgramID()`

Obtains the ID of a broadcast program being received.

Syntax

```
String getProgramID(input Number type)
```

Argument

type Type of ID to be obtained

Return values

null: Failure

Character string indicating the ID of a broadcast program being received (dependent of type specification): Success

Description

Depending on type, this function returns a value that is recognized based on the broadcasting standard. The available values to type and obtained strings are for further study.

`getActiveDocument()`

Returns the URI of a currently presented LIME document.

Syntax

```
String getActiveDocument()
```

Argument

None.

Return values

null: failure

Character string that conforms to the conventions on the namespace: success

Description

This function returns the URI of a currently presented LIME document.

`lockScreen()`

Locks the screen display.

Syntax

```
Number lockScreen()
```

Argument

None.

Return values

1: Success

NaN: Failure

Description

This function disables updating the screen.

`unlockScreen()`

Unlocks screen display.

Syntax

```
Number unlockScreen()
```

Argument

None.

Return values

1: Success.

NaN: Failure.

Description

This function enables updating the screen.

`getBrowserSupport( )`

Returns specified function is implemented or not by the browser.

Syntax

```
Number getBrowserSupport(
input String sProvider,
input String functionname
[,input String additionalinfo]+
)
```

Arguments

sProvider: Character string indicating the operators who defined this function.

functionname: Character string representing name of the function.

additionalinfo: Character string representing additional information of the function.

Return values

1: Specified function is implemented.

0: Specified function is not implemented.

Description

This function returns whether or not an extended function specified by a set of sProvider, functionname, and additionalinfo is implemented. Character strings assigned to sProvider, functionname, and additionalinfo are operationally defined. If a character string specified with one of these arguments is unknown to the implementation, the function returns 0 (specified function is not implemented). The character strings used for sProvider and functionname are case sensitive. The four-character string "ARIB" is reserved as an identifier of the functions specified in this standard, that is available to sProvide. More detailed usage of sProvider and functionname is operationally defined.

`launchDocument()`

Presents a LIME document.

Syntax

```
Number launchDocument(
input String documentName,
input String trasitionStyle
```

```
)
```

Arguments

`documentName`: Character string to specify a LIME document.

`transitionStyle`: Transition style.

Return values

1: Success.

NaN: Failure.

Description

This function opens a LIME document specified with documentName and presents it on the screen with a specified transition style.

– The scripts following the launchDocument() are not executed.

– If launchDocument() is executed in a global code, neither the "load" event nor the "unload" event occurs.

– If launchDocument() fails, it is not ensured that the following scripts are executed.

`launchDocumentRestricted()`

Presents a LIME document under a restricted condition.

Syntax

```
Number launchDocumentRestricted(
input String documentName,
input String trasitionStyle
)
```

Arguments

`documentName`: Character string to specify a LIME document.

`transitionStyle`: Transition style.

Return values

1: Success.

NaN: Failure.

Description

This function opens a LIME document specified with documentName and presents it on the screen with a specified transition style. Note that this function is applicable to a transition from a content received in real time or retained in a storage device to a LIME document over an interaction channel. Any LIME document to which the documentName LIME document transits based on this function or any further LIME document to which the destination document for launchDocumentRestricted ( ) transits based on the a element, the launchDocument() function , or others is not allowed to reference a resource broadcast in real time or a resource stored via a broadcasting service and to share information using Greg and NVRAM.

– The scripts following the launchDocumentRestricted ( ) are not executed.

– If launchDocumentRestricted ( ) is executed in a global code, neither the "load" event nor the "unload" event occurs.

– If launchDocumentRestricted ( ) fails, it is not ensured that the following scripts are executed.

`quitDocument()`

Quits presenting a LIME document.

Syntax

`Number quitDocument()`

Argument

None.

Return values

NaN

Description

This function quits presenting the specified LIME document.

`getFreeContentsMemory()`

Obtains a maximum size of a module that can be contained in a content memory.

Syntax

`Number getFreeContentsMemory([input Number number_of_resource])`

Arguments

number_of_resource: Number of resources.

Return values

Size of module that can be contained (in 1024-byte units)

NaN: Failure.

Description

This function returns a value (in 1024-byte units) representing a maximum size of a module that can be contained in a content memory, calculated based on the available area of a content memory at the time when the function is invoked.

If lockModuleOnMemory() was invoked to request a module to be locked and the lock has not been completed before the getFreeContentsMemory() function is invoked, the getFreeContentsMemory() returns the same value as that in the case where lockModuleOnMemory() was not invoked.

The maximum available value to number_of_resource is 999. Note that any return value is used only for reference purpose and does not ensure the returned size of module is successfully locked.

It is recommended that when in order to verify whether or not two or more modules are allowed to be locked per content, the concerned content is responsible for invoking getFreeContentsMemory() before a separate module is specified to be locked.

`isSupportedMedia()`

Verifies whether or not a service media type is supported.

Syntax

```
Number isSupportedMedia (input String mediaName)
```

Argument

`mediaName`: String representing a broadcasting media type to be verified.

Return values

1: Supported media type.

0: Not supported media type.

Description

This function verifies whether or not the broadcasting media type that is represented with a string is supported by a receiver. Any string specified with mediaName is case sensitive.

When an unknown string as mediaName , 0 is returned. The available values to mediaName of this function and linkMedia/Array[6] of linkMedia/Array[6] are for further study. For future uses, all the strings not listed below are reserved.

`detectComponent()`

Detects a component.

Syntax

```
Number detectComponent(input String component_ref)
```

Argument

`component_ref`: Component to be detected.

Return values

1: Specified component is described in PMT.

–1: Specified component is not described in PMT.

NaN: Failure.

Description

This function verifies whether or not the component specified with component_ref is described in PMT. The description of component_ref complies to the namespace.

`lockModuleOnMemoryEx()`

Receives a module into cache memory and locks the module.

Syntax

```
Number lockModuleOnMemoryEx(
input String module_ref
[,input Number remaining_space]
)
```

Argument

module_ref: URI identifying a module.

remaining_space: Free space in the content memory into which the specified module has been locked (in bytes). This argument accepts only an integral multiple of 4096. When a value that is not an integral multiple of 4096, the value is rounded up to the least integral multiple of 4096 of integral multiples of 4096 that are greater than the originally specified value to be interpreted as what remaining_space contains.

Return values

1: Success.

–3: No component transmitting the module exists (as far as detected based on PMT).

–4: Extra component is tried to be received.

NaN: Failure by other causes.

Description

This function receives a module specified with module_ref (including information related to a content) from the carousel and lock it in the content memory. The contents module is locked in cache memory until unlockModuleOnMemoryEx() or unlockModuleOnMemory() is called, the end of the tuning of its service, or any update to the currently presented data event is detected. The description of module_ref conforms to the conventions on namespace. This function exits without waiting for the module to be actually obtained. When a component that transmits the specified module does not exist, –3 is returned. When the maximum number of components have already received before this function specifies a component used to transmit the specified module, –4 is returned. The available maximum size is defined in an operational standard regulation. When the module is actually obtained, it is detected that the module does not exists, or it is detected that the available cache is smaller for caching the module, ModuleLocked specified with bevent occurs.

If lockModuleOnMemoryEx() tries to lock a module that has been locked, a ModuleLocked event is generated. This function is applicable to a module that is transmitted in a component that is part of the same service as that to which the currently presented document belongs. When the remaining_space argument is specified, the specified module is locked into the specified content memory as long as the content memory will have a free space at least as large as remaining_space. When a free space will be smaller than remaining_space, the specified module is not locked into the specified content memory. When the remaining_space argument is not specified, "0" is assumed as a value of the remaining_space argument.

`unlockModuleOnMemoryEx()`

Unlocks a locked module.

Syntax

```
Number unlockModuleOnMemoryEx(input String module_ref)
```

Argument

module_ref: URI identifying a module.

Return values

1: Success.

NaN: Failure.

Description

This function unlocks a module specified with module_ref to release it from the content memory. If the module has not been locked in the content memory by lockModuleOnMemoryEx(), Failure is

returned. The description of module_ref conforms to the conventions on namespace defined in clause 9.2.1.2. If a request to lock a module that has not been locked in the content memory is launched while this function tries to unlock the module, the request to lock the module is cancelled.

If unlockModuleOnMemoryEx() is invoked to unlock a module that has not been locked in a content memory and no request to lock is working on the module, Failure is returned. When one attemps to unlock by unlockModuleOnMemory() a module that has been locked by lockModuleOnMemoryEx(), an error is returned. This kind of unlocking is not supported.

`unlockAllModulesOnMemory()`

Unlocks all locked module.

Syntax

`Number unlockAllModulesOnMemory()`

Argument

None.

Return values

1: Success.

NaN: Failure.

Description

This function unlocks all modules locked in a content memory. This function is applicable to any module locked in a content memory despite of the function used to lock, lockModuleOnMemory() or lockModuleOnMemoryEx(). This function is also applicable to any module which has not been locked and on which a request to lock has been made. Any such request for a module is successfully cancelled.

`getLockedModuleInfo()`

Obtains a list of modules locked in a content memory.

Syntax

`Array getLockedModuleInfo()`

Arguments

None.

Return values

null: Failure.

Array containing information about modules: Success.

Array values/contents:

– Array[0]: Module status.

– Array[0][0]: Module name.

– Array[0][1]: Function that has requested module to be locked.

1: lockModuleOnMemory()

2: lockModuleOnMemoryEx()

– Array[0][2]: Locked status of module.

  1: Has been locked in contents memory.

  2: Locking request is working on.

– Array[1]: Module status.

– Array[1][0]: Module name.

– Array[1][1]: Function that has requested module to be locked.

– Array[1][2]: Locked status of module.

– A similar format is applied to Array[2].

Description

This function obtains a list of modules locked on content memory as an array. This list includes any module that has been locked in a content memory and any module that has not been locked but on which a locking request is working. When there are no applicable modules, an array of length 0 is returned. When an array of length 1 or greater is returned, each array element itself is an array object consisting of three elements. The first element contains a module name. The second element contains the specific function that is responsible for the locking, that is, lockModuleOnMemory() or lockModuleOnMemoryEx(). The third element verifies whether the module has been locked in a content memory or the module is in a locking request operation.

`getBrowserStatus()`

Obtains the status of a browser.

Syntax

```
Number getBrowserStatus (
input String sProvider,
input Sring statusname,
input String additionalinfo
)
```

Arguments

sProvider: String identifying a broadcaster or an entity that has configured the browser.

statusname: String describing a status name.

additionalinfo: String adding information about the status.

Return values

1: Indicates that the browser is in the specified status.

0: Indicates that the browser is not in the specified status.

NaN: Indicates that the status of the browser cannot be obtained.

Description

This function returns a value indicating whether or not the browser is in the status specified with a combination of the three Arguments, sProvider, statusname, and addtionalinfo. Strings applicable to the three arguments are defined in an operational standard regulation. Note that the four-character "ARIB" string is reserved as a string applicable to sProvide to identify a function defined in this specification. When one of the arguments contains a string unknown to an implementation, NaN

(return value indicating that the status of the browser cannot be obtained) is returned. The sProvider and statusname are case-sensitive arguments.

`getResidentAppVersion()`

Obtains information on resident application software, including versions.

Syntax

```
Array getResidentAppVersion(input String appName)
```

Arguments

appName: Name of a resident application software.

Return values

Array representing the application software information: Success.

Array[0]: String representing a manufacture ID.

Array[1]: A string arbitrarily defined by the manufacturer (maximum of 20 characters).

Array[2]: String representing a major version number.

Array[3]: String representing a minor version number.

Array[4]: More information for an individual resident application software.

null: Failure.

Description

This function obtains information used for identifying a resident application specified in the argument appName. Values applicable to the argument appName are defined in an operational standard regulation. In Array[0], the function returns a value representing a manufacture ID. The Array[0] contains a string representing a number in the hexadecimal notation. Note that the string requires a leading "0", if necessary, to be a two-digit number, instead of having characters or strings indicating that the string is the hexadecimal notation. This implies that a leading "0x" and an appended "h" must not be used. In Array[1], the function returns a string of 20 or less characters, that is defined arbitrarily by an individual manufacturer. Each character belongs to the CodeSet 0 of EUC-JP.

In Array[2] and Array[3], the function returns a string representing a version number, as defined arbitrarily by an individual manufacturer. The maximum length of each number is four digits in the hexadecimal notation. When the number has three or less digits, leading 0s are required to make it a four-digit number. In Array[4], the function returns more information on the resident application software , as specified for an individual type of the resident application software. How it is specified is defined in an operational standard regulation.

`setISPParams()`

Sets ISP parameters specific to automatic connection.

Syntax

```
Number setISPParams (
input String ispname,
input String tel,
input Boolean bProvider,
```

```
input String uid,
input String passwd,
input String nameServer1,
input String nameServer2,
input Boolean softCompression,
input Boolean headerCompression,
input Number idleTime,
input Number status
[,input Number lineType]
)
```

Arguments

ispname: String representing an ISP name.

tel: Telephone number character string. Note that an empty string is used when a line that requires no dialling is used.

bProvider: Network identification flag.

uid: User ID.

passwd: Password.

nameServer1: IP address of a primary name server.

nameServer2: IP address of a secondary name server.

softCompression: Flag indicating whether or not software compression is required.

encryptedPassword: Flag indicating whether or not encrypted password is used.

headerCompression: Flag indicating whether or not header compression is used.

idleTime: The maximum period of time in which the connection is kept without any data transmission and reception (in milliseconds).

status: Status of configured parameters

lineType: Preferred line type to be used for an ISP connection.

Return values

null: Failure.

Array[7]: Flag indicating whether or not header compression is used.

Array[8]: The maximum period of time in which the connection is kept without any data transmission or reception. (in milliseconds)

Array[9]: Status of configured parameters

Array[10]: String representing service operator identification, which conforms to the identifying information stored by a receiver feature when the etISPParams() function is executed. Detailed usage of strings are defined in an operational standard regulation.

Description

This function is applicable to a terminal that has an IP connection feature. It obtains connection parameters in a non-volatile memory as an Array object. The connection parameters include the Internet service operator and related parameters that are specific to the data broadcasting program currently received. To use this function more securely, guidelines for describing content to which this function is applicable, protecting the retained information, displaying confirming messages on a receiver, and others should be developed. Especially, great care should be put to prevent any unintended, accidental configuration even if the concerned content is a Class A content.

`getISPParams()`

Obtains ISP parameters specific to automatic connection.

Syntax

```
Array getISPParams ( )
```

Argument

None.

Return values

null: Failure.

Array[0]: String representing an ISP name.

Array[1]: Telephone number character string. Note that an empty string is used when a line that requires no dialling is used.

Array[2]: Network identification flag.

Array[3]: User ID.

Array[4]: IP address of a primary name server.

Array[5]: IP address of a secondary name server.

Array[6]: Flag indicating whether or not software compression is required.

Array[7]: Flag indicating whether or not header compression is used.

Array[8]: The maximum period of time in which the connection is kept without any data transmission or reception (in milliseconds).

Array[9]: Status of configured parameters.

Array[10]: String representing service operator identification, which conforms to the identifying information stored by a receiver feature when the setISPParams() function is executed. Detailed usage of strings are defined in an operational standard regulation.

Description

This function is applicable to a terminal that has an IP connection feature. It obtains connection parameters in a non-volatile memory as an Array object. The connection parameters include the Internet service operator and related parameters that are specific to the data broadcasting program currently received. To use this function more securely, guidelines for describing content to which this function is applicable, protecting the retained information, displaying confirming messages on a receiver, and others should be developed.

`connectPPP()`

Establishes a dial-up PPP connection.

Syntax

```
Number connectPPP (
input String tel,
input Boolean bProvider,
input String uid,
input String passwd,
input String nameServer1,
input String nameServer2,
input Boolean softCompression,
input Boolean headerCompression,
input Number idleTime
)
```

Arguments

tel:	Telephone number character string. Note that an empty string is used when a line that requires no dialling is used.
bProvider:	Network identification flag.
uid:	User ID.
passwd:	Password.
nameServer1:	IP address of a primary name server.
nameServer2:	IP address of a secondary name server.
softCompression:	Flag indicating whether or not software compression is required.
headerCompression:	Flag indicating whether or not header compression is used.
idleTime:	The maximum period of time in which the connection is kept without any data transmission or reception (in milliseconds).

Return values

1: Success.

–1: Parameter error.

–3: Time-out occurred.

–4: No dial tone detected.

–5: No carrier detected.

–6: Disconnection enforced.

–8: Line is busy.

–100: PPP connection has been established.

–200: Receiver has been configured not to use PPP for connections.

–301: Outside of the network service range (when use of mobile phone is preferred and line types are detectable).

–302: External communication device was not available (when use of mobile phone is preferred and line types are detectable).

NaN: Failure by other causes.

Description

This function establishes a PPP connection according to the specified arguments. This function is independent of configured parameters for a receiver to automatically connect to ISP (Internet Service Provider). When the bProvider network identification flag is true, a carrier identification code defined in a receiver's configuration may be placed at the beginning of the called telephone number. Any information specified with an argument of this function is only applicable to a PPP connection that is established using this function. When a line type that does not perform an explicit dialling is used as the preferred line type, the tel argument may contain an empty string. An established PPP connection is disconnected in cases; when the disconnectPPP() function is explicitly executed, when the period of time specified in idleTime has passed before a packet is sent/received, or when a disconnecting feature in a receiver is explicitly invoked by an end user. The –100 return value (Failure) is returned and the function exits when the PPP connection has already been established using an automatic connection feature in the receiver or an automatic connection function. The –200 return value (Failure) is returned and the function exits when a receiver supports only Fixed IP/DHCP as connection protocols. The –301 return value (Failure) is returned and the function exits when the preferred line type is mobile phone and the function is used outside of the concerned network service range. The –302 return value (Failure) is returned and the function exits when the concerned external communication device is not available.

`connectPPPWithISPParams()`

Establishes a PPP connection.

Syntax

```
Number connectPPPWithISPParams(
[input Number idleTime]
)
```

Argument

idleTime: The maximum period of time in which the connection is kept without any data transmission and reception (in milliseconds).

Return values

1: Success.

–1: Parameter error.

–3: Time-out occurred.

–4: No dial tone detected.

–5: No carrier detected.

–6: Disconnection enforced.

–7: Modem in use.

–8: Line is busy.

–100: PPP connection has been established.

–200: Receiver has been configured not to use PPP.

–301: Outside of the network service range (when use of mobile phone/PHS is preferred and line types are detectable).

–302: External communication device was not available (when use of mobile phone/PHS is preferred and line types are detectable).

NaN: Failure by other causes.

Description

This function establishes a PPP connection according to the receiver configuration, especially the ISP connection related parameters applicable to automatic connection. An established PPP connection is disconnected in cases: when the disconnectPPP() function is explicitly executed, when the period of time defined in the receiver or specified with idleTime has passed before a packet is sent/received, or a disconnecting feature in a receiver is explicitly invoked by an end user. When no value is set for idleTime, a value configured in a receiver is recognized as a default value. The –100 return value (Failure) is returned and the function exits when the PPP connection has been established using an automatic connection feature in the receiver or an automatic connection function. The –200 return value (Failure) is returned and the function exits when the preferred line type has not been configured to use PPP. The –301 return value (Failure) is returned and the function exits when the preferred line type is mobile phone and the function is used outside of the concerned network service range. The –302 return value (Failure) is returned and the function exits when the concerned external communication device is not available.

`disconnectPPP()`

Disconnects an established PPP connection.

Syntax

```
Number disconnectPPP ( )
```

Argument

None.

Return values

1: Success.

–1: No PPP connection has been established.

–200: Receiver has been configured not to use PPP.

NaN: Failure.

Description

This function disconnects a PPP connection that has been established using the connectPPP() function, the connectPPPWithISPParams() function, or an automatic connection feature in a receiver. An established line connection is also disconnected. The –200 return value (Failure) is returned and the function exits when the receiver supports no PPP connections. The NaN return value (Failure) is returned and the function exits when this function has been executed to fail to disconnect an established PPP connection due to a busy line which is occupied by another application in a receiver or other causes.

`getConnectionType()`

Obtains a preferred line type used to connect to ISP.

Syntax

`Number getConnectionType ( )`

Arguments

None.

Return values

1: PSTN.

100: ISDN.

200: PHS (No specific PHS type was identified).

201: PHS (PIAFS2.0).

202: PHS (PIAFS2.1).

300: Mobile phone (No specific mobile phone type was identified).

301: Mobile phone (PDC).

302: Mobile phone (PDC-P).

303: Mobile phone (DS-CDMA).

304: Mobile phone (MC-CDMA).

305: Mobile phone (CDMA cellular system).

401: Ethernet (PPPoE).

402: Ethernet (Fixed IP).

403: Ethernet (DHCP).

NaN: Failure.

Description

This function is applicable to a terminal that has an IP connection feature. This function returns the preferred line type used by a receiver to automatically connect to ISP either via the receiver's automatic ISP connection feature or an automatic connection function, `connectPPP()` or `connectPPPWithISPParams()`. The return value 200 is retuned, when the preferred line type is PHS and the specific type (PIAFS2.0 or PIAFS2.1) is not identified. The return value 300 is retuned when the preferred line type is Mobile phone and the connection procedure specific to the carrier is not identified.

`isIPConnected()`

Verifies whether or not an IP (Internet Protocol) connection has been established.

Syntax

Number isIPConnected ( )

Arguments

None.

Return values

0: No IP connection has been established.

1: IP connection has been established using automatic connection feature.

2: IP connection has been established using the connectPPP()/connectPPPWithISPParams() function.

NaN: Failure.

Description

This function is applicable to a terminal that has an IP connection feature. This function returns a value indicating whether or not an IP connection has been established by the receiver.

`transmitTextDataOverIP()`

Sends and receives a text mail using TCP/IP.

Syntax

```
Array transmitTextDataOverIP(
input String uri,
input String text,
input String charset
)
```

Arguments

`uri`: URI representing a service that send the specified text data.

`text`: Text data to be sent.

`charset`: Character encoding used to send and receive the text data.

The available values are:

"`EUC-JP`" EUC-JP

"`Shift_JIS`" Shift-JIS

"`UTF-8`" UCS/UTF-8

"`UTF-16`" UCS/UTF-16

Return values

Array[0]: Numeric value representing the result code

1: Success.

–1: Parameter error.

–2: Line was disconnected during transfer.

–3: Time-out occurred.

–300: Failed to establish an automatic connection.

–400: Failed to map names using DNS.

–500: Failed to process TLS-based operation.

NaN: Failure by other causes.

Array[1]: Status-Code string in HTTP1.1.

Array[2]: Received text data.

Description

This function sends text data to the resource on the Internet specified in the `uri` argument. The protocol used to send the data depends on uri. When "`https://`" is described in `uri`, the function requires the receiver to operate TLS-based operation before the function sends or receives the data. The acceptable size of text data and the character encoding (charset) used to send/receive the data are defined in an operational standard regulation.

`playRomSound()`

Plays sound of an event built in the receiver.

Syntax

```
Number playRomSound(input String soundID)
```

Argument

soundID: Identifies sound of an event built in the receiver based on the namespace convention (romsound://<sound_id>).

Return values

1: Success.

NaN: Failure.

Description

This function plays sound of an event built in the receiver that is specified with soundID based on the conventions on the namespace.

`sleep()`

Pauses processing for a period specified in milliseconds.

Syntax

```
Number sleep(input Number interval)
```

Argument

interval Pausing interval (in milliseconds).

Return values

1: Success.

NaN: Failure.

Description

This function pauses processing for a period specified with interval (in milliseconds).

`setInterval()`

Performs a processing command in each specified interval (in milliseconds).

Syntax

```
Number setInterval(
input String func,
```

```
input Number msec,
input Number iteration
)
```

Arguments

func: Command or function name executed by this function.

msec: Interrupt interval (in milliseconds).

iteration: Number of repeats.

Return values

Positive value: Registered timer ID.

NaN: Failure.

Description

This function invokes a function or command specified with func in each interval specified with msec for the number of times specified with iteration. If iteration is 0 (zero), the invocation is repeated until clearInterval is called.

`clearTimer()`

Terminates processing of a registered timer ID which is specified.

Syntax

```
Number clearTimer (input Number timerID)
```

Arguments

timerID: Registered timer ID.

Return values

1: Success.

NaN: Failure.

Description

This function cancels processing of a registered timer ID specified with timerID.

`pauseTimer()`

Pauses the timer with a registered timer ID which is specified.

Syntax

```
Number pauseTimer (input Number timerID)
```

Argument

timerID: Registered timer ID.

Return values

1: Success.

NaN: Failure.

Description

This function gives a pause to the timer that has been registered with timerID. Unlike the sleep function, other functions are not affected. This function is applicable to a timer generated by setInterval().

`resumeTimer()`

Resumes a paused timer with a registered timer ID which is specified.

Syntax

```
Number resumeTimer(input Number timerID)
```

Argument

timerID: Registered timer ID.

Return values

1: Success.

NaN: Failure.

Description

This function resumes the paused timer that has been registered with timerID. This function applicable to a timer generated by setInterval(). Once this function has been executed, any interval consumes the specified milliseconds in Timer functions, instead of the remaining milliseconds when the timer was paused by pauseTimer(). That is, once resumeTimer( ) has been executed, any following function is executed when the specified interval expires. Once this function has been executed to a timer generated by setInterval(), the timer is invoked for the number of times, that is the result of subtracting the number of times for which the timer had been invoked until the timer was paused by pauseTimer() from the number specified with iteration. However, when iteration is 0, the timer is invoked iteratively until clearTimer () is invoked.

`setCurrentDateMode()`

Specifies the type of time to be referenced when performing `Date()` *and other built-in functions.*

Syntax

```
Number setCurrentDateMode(input Number time_mode)
```

Arguments

time_mode: Time mode (0: Absolute playback time; 1: Reception time)

Return values

1: Success.

NaN: Failure.

Description

This function specifies the type of time to be obtained by a time acquisition functionality provided by Date() and other ECMAScript built-in functions. If time_mode is 0 (zero), the absolute time at which the playback starts is specified. When playing a stream-recorded content, the absolute time during playback is also referenced. In this case, for example, it is assumed that when playing the received contents, the time in TOT/TDT or the time of a clock that is based on TOT/TDT is referenced, and when playing a stream-stored contents, a clock that retains the absolute time during

playback is referenced. If time_mode is 1 (one), the absolute time during playback is specified. When playing a received content at a time, the operation is the same as for time_mode 0 (zero). When playing a stream-recorded content, the operation is controlled based on the time standard at the time of receive. In this case, for example, it is assumed that when playing the received contents, the time in TOT/TDT or the time of a clock that is based on TOT/TDT is referenced. It is assumed that when playing a stored-stream content, a clock that is based on PartialTS Time Descriptor of SIT is referenced.

`loadDRCS()`

Configures external character data.

Syntax

```
Number loadDRCS(input String DRCS_ref)
```

Argument

DRCS_ref: URI representing a location containing external character data

Return values

1: Success

NaN: Failure

Description

This function loads external character data from DRCS data in a URI location specified in DRCS_ref. The description of DRCS_ref conforms to the namespace. The loaded external character data is effective until unloadDRCS() is called or the display of a LIME document ends. The content referenced by DRCS_ref conforms to the format conventions described in [ARIB STD-B24] Volume 1, Appendix D.

`setCCDisplayStatus()`

Switches the display state of the specified language.

Syntax

```
Number setCCDisplayStatus(
input Number language,
input Boolean status
)
```

Arguments

language: Language selection

1: First language.

2: Second language.

3: Third language.

4: Fourth language.

5: Fifth language.

6: Sixth language.

7: Seventh language.

8: Eighth language.

status Display control (True: present; False: do not present)

Return values

1: Success.

NaN: Failure.

Description

This function switches the display state of the language specified by the first argument to the state specified by the second argument. If the subtitle does not include the specified language, the return value is NaN for the status set to True (display) and '1' for the status set to False (not display). If the display state of the subtitle is changed after performing this function, the event CCStatusChanged occurs. Further, if the status is set to True for the language which has been displayed, or if the status is set to False for the language which has not been displayed, the display status of subtitle is not changed and the value '1' is returned.

`getCCDisplayStatus()`

Obtains the display state of the subtitle for each language.

Syntax

```
Number getCCDisplayStatus(input Number language)
```

Argument

language: Language selection

1: First language.

2: Second language.

3: Third language.

4: Fourth language.

5: Fifth language.

6: Sixth language.

7: Seventh language.

8: Eighth language.

Return values

0: The specified language in the subtitle is in hidden state.

1: The specified language in the subtitle is in display state.

NaN: Failure.

Description

This function obtains the display state of the language specified by the argument. If the subtitle does not include the language specified by the argument, the return value is 0.

`getCCLanguageStatus()`

Verifies whether or not a specified language exists in the subtitle.

Syntax

String getCCLanguageStatus(input Number language)

Argument

language: Language selection

1: First language.

2: Second language.

3: Third language.

4: Fourth language.

5: Fifth language.

6: Sixth language.

7: Seventh language.

8: Eighth language.

Return values

0: The specified language does not exist in the subtitle.

1: The specified language exists in the subtitle.

NaN: Failure.

Description

This function verifies whether or not the language specified by the argument exists in the subtitle.

9.2.1.1 Identification of events

The following character string is used to reference an event.

`scheme`://<*original_network_id*>.<*transport_stream_id*>.<*service_id*>.<*event_id*>

NOTE – `scheme` is currently fixed to "arib" in Japan.

9.2.1.2 Identification of resources

Module

Any module is uniquely identified in the network with the following name.

`scheme`://<*original_network_id*>.<*transport_stream_id*>.<*service_id*>
[;<*content_id*>] [.<*event_id*>]/<*component_tag*>/<*moduleName*>

<*moduleName*> is a character string in Name descriptor of DII (download info indication). If no Name descriptor is used, moduleId must be assigned to <*moduleName*> and:

– If only one content exists in an event, ";<*content_id*>" may be omitted.

– If the current service is specified without using an event identifier, ".<*event_id*>" may be omitted.

– "*content_id*" must be used to reference a stored content. "*event_id*" is not used.

– IDs other than <*moduleName*> are described in hexadecimal notation.

– A moduleId used for <*moduleName*> is a hexadecimal character string.

NOTE – "`scheme`" is currently fixed to "arib-dc" in Japan.

Resource directly mapped to a module

When a resource is directly mapped to a module, the resource is identified with a name based on the conventions described for "Module" above.

Resource stored in a module in entity format

Any resource packaged in a module in an entity format is uniquely identified with the module name followed by the resource name, as shown below.

`scheme`://<*original_network_id*>.<*transport_stream_id*>.<*service_id*>[;<*content_id*>]
[.<*event_id*>]/<*component_tag*>/<*moduleName*>/<*resourceName*>

<*resourceName*> is equivalent to the character string specified in Content-Location: of the resource entity header. <*resourceName*> is case insensitive.

NOTE – "`scheme`" is currently fixed to "arib-dc" in Japan.

10 Use of DOM in LIME

Table 10-1 gives the DOM interfaces that are used in LIME-DOM and Table 10-2 shows the profile for the DOM core basic interface attributes.

Table 10-1 – DOM core fundamental interfaces

Interface	Operation
Basic interface group	
DOMException	–
DOMImplementation	R1
DocumentFragment	–
Document	R1
Node	R1
NodeList	–
NamedNodeMap	–
CharacterData	R1
Attr	–
Element	R1
Text	R1
Comment	–
Extended interface group	
CDATASection	R1
DocumentType	–
Notation	–
Entity	–
EntityReference	–
ProcessingInstruction	–

Table 10-2 – DOM core basic interface attributes of LIME-DOM

Interface	Attribute/method	Operation	Restriction
DOMImplementation			
	hasFeature()	R1	
Document			
	doctype	–	
	implementation	R1	R
	documentElement	R1	R
	createElement()	–	
	createDocumentFragment()	–	
	createTextNode()	–	
	createComment()	–	
	createCDATASection()	–	
	createProcessingInstruction()	–	
	createAttribute()	–	
	createEntityReference()	–	
	getElementByTadName()	–	
Node		–	
	nodeName	–	
	nodeValue	–	
	nodeType	–	
	parentNode	R1	R
	childNodes	–	
	firstChild	R1	R
	lastChild	R1	R
	previousSibling	R1	R
	nextSibling	R1	R
	attributes	–	
	ownerDocument	–	
	insertBefore	–	
	replaceChild	–	
	removeChild	–	
	appendChild	–	
	hasChildNodes()	–	
	cloneNode()	–	
CharacterData			
	data	R1	RW (Note)
	length	R1	R
	substringData()	–	
	appendData()	–	
	insertData()	–	

Table 10-2 – DOM core basic interface attributes of LIME-DOM

Interface	Attribute/method	Operation	Restriction
	deleteData()	–	
	replaceData()	–	
Element			
	tagName()	R1	R
	getAttribute()	–	
	setAttribute()	–	
	removeAttribute()	–	
	getAttributeNode()	–	
	setAttributeNode()	–	
	removeAttributeNode()	–	
	getElementsByTagName()	–	
	normalize()	–	
Text			
	splitText()	–	
CDATASection			
NOTE – The child nodes of script and style are not accessed in the operation. Only the child nodes of *p*, *span* and *a* can be written in the operation.			

10.1 DOM HTML interface group

Table 10-3 shows the profile of DOM HTML interface used in LIME-DOM and Table 10-4 shows the profile of attributes and methods of the DOM HTML interface group for LIME.

Table 10-3 – Profile of DOM HTML interface group

Interface	Operation
HTMLCollection	–
HTMLDocument	R1
HTMLElement	R1
HTMLBlockquoteElement	–
HTMLPreElement	–
HTMLHeadingElement	–
HTMLHRElement	–
HTMLDivElement	R1
HTMLParagraphElement	R1
HTMLQuoteElement	–
HTMLBRElement	R1
HTMLModElement	–
HTMLAnchorElement	R1
HTMLBaseElement	–
HTMLLinkElement	–

Interface	Operation
HTMLDListElement	–
HTMLOListElement	–
HTMLUListElement	–
HTMLLIElement	–
HTMLButtonElement	–
HTMLFieldSetElement	–
HTMLFormElement	–
HTMLInputElement	R1
HTMLLabelElement	–
HTMLLegendElement	–
HTMLOptGroupElement	–
HTMLOptionElement	–
HTMLSelectElement	–
HTMLTextAreaElement	–
HTMLTableCaptionElement	–

Table 10-3 – Profile of DOM HTML interface group

Interface	Operation
HTMLTableColElement	–
HTMLTableElement	–
HTMLTableSectionElement	–
HTMLTableCellElement	–
HTMLTableRowElement	–
HTMLImageElement	–
HTMLAreaElement	–
HTMLMapElement	–
HTMLObjectElement	R1
HTMLParamElement	–

Interface	Operation
HTMLFrameSetElement	–
HTMLFrameElement	–
HTMLIFrameElement	–
HTMLMetaElement	R1
HTMLTitleElement	R1
HTMLScriptElement	R1
HTMLStyleElement	R1
HTMLBodyElement	R1
HTMLHeadElement	R1
HTMLHtmlElement	R1

Table 10-4 – Profile of attributes and methods of DOM HTML interface group

Interface	Attribute/method	Operation	Restriction
HTMLDocument			
	title	–	
	referrer	–	
	domain	–	
	uRL	–	
	body	–	
	images	–	
	applets	–	
	links	–	
	forms	–	
	anchors	–	
	cookie	–	
	open()	–	
	close()	–	
	write()	–	
	writeln()	–	
	getElementById()	R1	
	getElementsByName()	–	
HTMLElement			
	id	R1	R
	title		
	lang		
	dir		
	className	R1	R

Table 10-4 – Profile of attributes and methods of DOM HTML interface group

Interface	Attribute/method	Operation	Restriction
HTMLDivElement			
HTMLParagraphElement			
HTMLBRElement			
HTMLAnchorElement			
	accesskey	R1	R
	charset	–	
	cords	–	
	href	R1	RW
	hreflang	–	
	name	–	
	rel	–	
	rev	–	
	shape	–	
	tabIndex	–	
	target	–	
	type	–	
	blur()	R1	
	focus()	R1	
HTMLInputElement			
	defaultValue	R1	R
	defaultChecked	–	
	form	–	
	accept	–	
	accesskey	R1	R
	alt	–	
	checked	–	
	disabled	R1	RW
	maxLength	R1	R
	name	–	
	readOnly	R1	RW
	size	–	
	src	–	
	tabIndex	–	
	type	R1	R
	useMap	–	
	value	R1	RW
	blur()	R1	
	focus()	R1	
	select()	–	

Table 10-4 – Profile of attributes and methods of DOM HTML interface group

Interface	Attribute/method	Operation	Restriction
	click()	–	
HTMLObjectElement			
	form	–	
	code	–	
	archive	–	
	codebase	–	
	codeType	–	
	data	R1	RW (Note)
	declare	–	
	height	–	
	name	–	
	standby	–	
	tabIndex	–	
	type	R1	R
	useMap	–	
	width	–	
HTMLMetaElement			
	content	R1	R
	httpEquiv	–	
	name	R1	R
	scheme	–	
HTMLTitleElement			
	text	R1	R
HTMLScriptElement			
	text	–	
	htmlFor	–	
	event	–	
	charset	–	
	defer	–	
	src	–	
	type	–	
HTMLStyleElement			
	disabled	–	
	media	–	
	type	–	
HTMLBodyElement			
HTMLHeadElement			
	profile	–	
HTMLHtmlElement			

Table 10-4 – Profile of attributes and methods of DOM HTML interface group

Interface	Attribute/method	Operation	Restriction
	version	–	
NOTE – If the DOM application programming interface (API) changes the data attribute of an object concerning the monomedia that is transmitted using data carousel, the data attribute value will be read again even when remaining unchanged. If the module containing a resource specified by the data attribute is locked, the locked data will be applied as it is; otherwise, the presentation must be updated after getting the data from a transmission stream again. Note that dynamically changing type attributes and dynamically changing schemas by changing data attributes for sound are not applicable to the object element.			

10.2 DOM interface specific to LIME-DOM

This interface operates LIME element attributes and CSS properties. LIME documents are based on XHTML 1.0. Therefore, the DOM Level1 HTML DOM interfaces can be applied to operate element attributes. However, the HTML DOM interfaces do not define operations of element CSS properties. So that extended HTML DOM interfaces for operating the CSS properties of LIME-HTML elements as well as interfaces for handling elements with attributes added for LIME are required. This clause defines these interfaces.

Table 10-5 summarizes the profile of the DOM interface specific to LIME-DOM and Table 10-6 shows the profile of attributes and methods of the DOM interface for LIME-DOM. Any defined attribute and method that is not listed is assumed that its operation is "–".

Table 10-5 – Profile of interface (DOM interface group)

Interface	Operation
LIMEDocument	R1
LIMEElement	R1
LIMEBlockquoteElement	–
LIMEPreElement	–
LIMEHeadingElement	–
LIMEHRElement	–
LIMEDivElement	R1
LIMESpanElement	R1
LIMEParagraphElement	R1
LIMEQuoteElement	–
LIMEBRElement	R1
LIMEModElement	–
LIMEAnchorElement	R1
LIMELinkElement	–
LIMEDListElement	–
LIMEOListElement	–
LIMEUListElement	–
LIMELIElement	–
LIMEButtonElement	–
LIMEFieldSetElement	–

Table 10-5 – Profile of interface (DOM interface group)

Interface	Operation
LIMEFormElement	–
LIMEInputElement	R1
LIMELabelElement	–
LIMELegendElement	–
LIMEOptGroupElement	–
LIMEOptionElement	–
LIMESelectElement	–
LIMETextAreaElement	–
LIMETableCaptionElement	–
LIMETableColElement	–
LIMETableElement	–
LIMETableSectionElement	–
LIMETableCellElement	–
LIMETableRowElement	–
LIMEImageElement	–
LIMEAreaElement	–
LIMEMapElement	–
LIMEObjectElement	R1
LIMEFrameSetElement	–
LIMEFrameElement	–
LIMEIFrameElement	–
LIMEBodyElement	R1
LIMEBmlElement	R1
LIMEBeventElement	R1
LIMEBeitemElement	R1
LIMEListTableElement	–
LIMEItemElement	–

Table 10-6 summarizes the attributes and methods of the DOM interface for LIME-DOM. Any defined attribute and method that is not listed is assumed that its operation is "–".

Table 10-6 – Profile of attributes and methods (DOM interface group)

Interface	Attribute/method	Operation	Remarks
LIMEDocument			
	currentFocus	R1	R
	currentEvent	R1	R (Note 2)
LIMEDivElement			
	style	–	
	normalStyle	R1	RW (Note 1)

Table 10-6 – Profile of attributes and methods (DOM interface group)

Interface	Attribute/method	Operation	Remarks
	focusStyle	R1	RW (Note 1)
	activeStyle	R1	RW (Note 1)
	accessKey	R1	R
	focus()	R1	
	blur()	R1	
LIMESpanElement			
	style	–	
	normalStyle	R1	RW (Note 1)
	focusStyle	R1	RW (Note 1)
	activeStyle	R1	RW (Note 1)
	accessKey	R1	R
	focus()	R1	
	blur()	R1	
LIMEParagraphElement			
	style	–	
	normalStyle	R1	RW (Note 1)
	focusStyle	R1	RW (Note 1)
	activeStyle	R1	RW (Note 1)
	accessKey	R1	R
	focus()	R1	
	blur()	R1	
LIMEBRElement			
	style	–	
	normalStyle	R1	RW (Note 1)
	focusStyle	–	
	activeStyle	–	
LIMEAnchorElement			
	style	–	
	normalStyle	R1	RW (Note 1)
	focusStyle	R1	RW (Note 1)
	activeStyle	R1	RW (Note 1)
	effect	–	
LIMEInputElement			
	style	–	
	normalStyle	R1	RW (Note 1)
	focusStyle	R1	RW (Note 1)
	activeStyle	R1	RW (Note 1)
LIMEObjectElement			
	style	–	

Table 10-6 – Profile of attributes and methods (DOM interface group)

Interface	Attribute/method	Operation	Remarks
	normalStyle	R1	RW (Note 1)
	focusStyle	R1	RW (Note 1)
	activeStyle	R1	RW (Note 1)
	classId	–	
	accessKey	R1	R
	remain	R1	RW
	streamPosition	R1	RW (Note 3)
	streamStatus	R1	RW
	streamLooping	–	
	streamSpeedNumerator	–	
	streamSpeedDenominator	–	
	streamLevel	–	
	setSpeed()	–	
	movePosition()	–	
	hasAssociatedIndex()	–	
	assignToLocalEvent()	–	
	assignToNodePlayMode()		
	getMainAudioStream()	R1	
	setMainAudioStream()	R1	
	focus()	R1	
	blur()	R1	
LIMEBodyElement			
	invisible	R1	RW
	style	–	
	normalStyle	R1	RW (Note 1)
	focusStyle	–	
	activeStyle	–	
LIMEBmlElement			
	style	–	
	normalStyle	–	
	focusStyle	–	
	activeStyle	–	
LIMEBeventElement			
LIMEBeitemElement			
	type	R1	R
	esRef	R1	RW
	messageGroupId	R1	R
	messageId	R1	RW
	messageVersion	R1	RW

Table 10-6 – Profile of attributes and methods (DOM interface group)

Interface	Attribute/method	Operation	Remarks
	moduleRef	R1	RW
	languageTag	R1	RW
	registered	–	
	serviceId	–	
	eventide	–	
	timeMode	R1	R
	timeValue	R1	RW
	objectId	R1	RW
	subscribe	R1	RW
NOTE 1 – These attributes are accessed through the LIMECSS2PropertyInterface. If the attributes are read directly, an object is returned. Writing the attributes directly is inhibited. NOTE 2 – The values derived from document.currentEvent must not be referenced by other handlers by being substituted to global variables. The result of the substitution is not guaranteed. NOTE 3 – It can be read and written only if the type attribute is "image/X-arib-mng". The "setMainAudioStream()" and "getMainAudioStream()" methods of the BMLObjectElement interface are not used by LIME. The "invisible" attribute of the BMLBodyElement interface is not used by LIME.			

LIMEDocument DOM interface

This interface operates the whole LIME document. It is the HTMLDocument interface defined in Table 10-4 with methods for obtaining context information of the event currently processed and methods for obtaining the LIME object element that has the focus.

Interface definition:

```
interface LIMEDocument : HTMLDocument {
readonly attribute BMLEvent currentEvent;
readonly attribute LIMEElement currentFocus;
};
```

Attributes:

`currentEvent` Context information that indicates the event currently processed.

`currentFocus` LIME object element that has the focus.

Method:

None.

LIME element DOM interface

The following interfaces are HTML DOM interfaces defined to operate the CSS property of a LIME document. Each interface inherits an HTML DOM interface that corresponds to the element with an extension of `normalStyle`, `focusStyle`, and `activeStyle` attributes that are LIMECSS2Properties objects for retaining the CSS properties.

LIMEBlockquoteElement interface

Interface definition:

```
interface LIMEBlockquoteElement : HTMLBlockquoteElement {
```

```
attribute LIMECSS2Properties normalStyle;
attribute LIMECSS2Properties focusStyle;
attribute LIMECSS2Properties activeStyle;
};
```

Attributes:

`normalStyle`	Retains the inherited value of CSS property that is applied for presentation in normal state. The retained value must be a computed value. Therefore, "inherit" must not be specified.
`focusStyle`	Retains the inherited value of CSS property that is applied for presentation in focus state. However, before the value of a CSS property in focusStyle is changed at first, it must not affect the decision on the value applied to the CSS property. And the retained value must l be a computed value. Therefore "inherit" must not be specified for this attribute.
`activeStyle`	Retains the inherited value of CSS property that is applied for presentation in an active state (e.g., when the Enter key on a remote control was pressed.). However, before the value of a CSS property in activeStyle is changed at first, it must l not affect the decision on the value applied to the CSS property. And the retained value must be a computed value Therefore "inherit" must not be specified for this attribute.

LIMEDivElement interface

This interface is used for the *div* element. It corresponds to the additional definition of the attributes the for *div* element.

Interface definition:

```
interface LIMEDivElement : HTMLDivElement {
attribute DOMString accessKey;
attribute LIMECSS2Properties normalStyle;
attribute LIMECSS2Properties focusStyle;
attribute LIMECSS2Properties activeStyle;
void focus();
void blur();
};
```

Attributes:

`accessKey`	Value of the accesskey attribute
`normalStyle`	Inherited value of CSS property that is applied for presentation in normal state.
`focusStyle`	Inherited value of CSS property that is applied for presentation in focus state.
`activeStyle`	Inherited value of CSS property that is applied to presentation in active state.

Methods:

`focus()`	Moves the focus to the item. Parameter: None Return value: None
`blur()`	Moves the focus away from the item.

Parameter: None

Return value: None

LIMESpanElement interface

This interface is used for the span element. It corresponds to the additional definition of the attributes for the span element.

Interface definition:

```
interface LIMESpanElement : HTMLSpanElement {
attribute DOMString accessKey;
attribute LIMECSS2Properties normalStyle;
attribute LIMECSS2Properties focusStyle;
attribute LIMECSS2Properties activeStyle;
void focus();
void blur();
};
```

Attributes:

`accessKey`	Value of the accesskey attribute.
`normalStyle`	Inherited value of CSS property that is applied for presentation in normal state.
`focusStyle`	Inherited value of CSS property that is applied for presentation in focus state.
`activeStyle`	Inherited value of CSS property that is applied to presentation in active state.

Method:

`focus()`	Moves the focus to the item. Parameter: None Return value: None
`blur()`	Moves the focus away from the item. Parameter: None Return value: None

LIMEParagraphElement interface

This interface is used for the *p* element. It corresponds to the additional definition of attributes for the *p* element.

Interface definition:

```
interface LIMEParagraphElement : HTMLParagraphElement {
attribute DOMString accessKey;
attribute LIMECSS2Properties normalStyle;
attribute LIMECSS2Properties focusStyle;
attribute LIMECSS2Properties activeStyle;
void focus();
void blur();
};
```

Attributes:

`accessKey`	Value of the accesskey attribute.

`normalStyle`	Inherited value of CSS property that is applied for presentation in normal state.
`focusStyle`	Inherited value of CSS property that is applied for presentation in focus state.
`activeStyle`	Inherited value of CSS property that is applied to presentation in active state.

Methods:

`focus()`	Moves the focus to the item. Parameter: None Return value: None
`blur()`	Moves the focus away from the item. Parameter: None Return value: None

LIMEAnchorElement interface

This interface is used for the *a* element. It corresponds to the additional definition of attributes for the *a* element above.

Interface definition:

```
interface LIMEAnchorElement : HTMLAnchorElement {
attribute LIMECSS2Properties normalStyle;
attribute LIMECSS2Properties focusStyle;
attribute LIMECSS2Properties activeStyle;
};
```

Attributes:

`normalStyle`	Inherited value of CSS property that is applied for presentation in normal state.
`focusStyle`	Inherited value of CSS property that is applied for presentation in focus state.
`activeStyle`	Inherited value of CSS property that is applied to presentation in active state.

Method:

None.

LIMEObjectElement interface

This interface is used for the *object* element. It corresponds to the classId attribute defined in XHTML 1.0 (undefined in HTML DOM) and additional attribute definitions of the object element.

Interface definition:

```
interface LIMEObjectElement : HTMLObjectElement
attribute LIMECSS2Properties normalStyle;
attribute LIMECSS2Properties focusStyle;
attribute LIMECSS2Properties activeStyle;
attribute boolean remain;
attribute long streamPosition;
attribute DOMString streamStatus;

attribute DOMString mainAudioStreaml;
```

```
boolean selectMainAudioStream(input DOMString audio_ref);
attribute DOMString accessKey;
void focus();
void blur();
};
```

Attributes:

`normalStyle`	Inherited value of CSS property that is applied for presentation in normal state.
`focusStyle`	Inherited value of CSS property that is applied for presentation in focus state.
`activeStyle`	Inherited value of CSS property that is applied to presentation in active state.
`remain`	If true, continues monomedia play while document transition. Value of the remain attribute of the object element
`streamPosition`	Relative position of play to the head of the stream. Value of the streamposition attribute of the object element
`streamStatus`	State of stream. The value shall be "play", "stop" or "pause". Changing this value controls playback of monomedia. Value of the streamstatus attribute of the object element
`accessKey`	Value of the accesskey attribute

Methods:

`setMainAudio Stream()`	Applicable to the object element with the main audio stream specified by setting component_tag=-1 in the data element. This method controls switching of the main audio stream. Parameter: audio_ref URI character string indicating audio ES/channel in the following format: /<component_tag>[;<channel_id>/] Return value: true for success and false for fail.
`getMainAudio Stream()`	Applicable to an audio stream with setting component_tag=-1 in data element. This method obtains URI character string indicating selected audio ES and channel. Otherwise, obtains null. Parameter: None Return value: URI character string indicating audio ES and channel, or null.
`focus()`	Moves the focus to the item. Parameter: None Return value: None
`blur()`	Moves the focus away from the item. Parameter: None Return value: None

LIMEBodyElement interface

This interface is used for the body element. It corresponds to the additional attribute definition of the body element.

Interface definition:

```
interface LIMEBodyElement : HTMLBodyElement {
attribute LIMECSS2Properties normalStyle;
attribute boolean invisible;
}
```

Attributes:

`normalStyle`	Contains the inherited value of a CSS property that is applied for presentation in normal state.
`invisible`	When it is true, no element and no background of the LIME document is displayed.

Methods:

None.

LIMEBeitemElement interface

This interface is used for the beitem element, which is an extended LIME element for specifying events defined.

Interface definition:

```
interface LIMEBeitemElement : HTMLElement {
attribute readonly DOMString type:
attribute DOMString esRef;
attribute unsigned short messageId;
attribute unsigned short messageVersion;
attribute unsigned short messageGroupId;
attribute DOMString moduleRef;
attribute unsigned short languageTag;
attribute DOMString timeMode;
attribute DOMString timeValue;
attribute DOMString objectId;
attribute DOMString segmentId;
attribute boolean subscribe:
};
```

Attributes:

`type`	Type of events. Value of type attribute LIME of beitem element
`esRef`	Value of es_ref attribute LIME of beitem element
`MessageId`	Value of message_id attribute LIME of beitem element
`MessageVersion`	Value of message_version attribute LIME of beitem element
`MessageGroupId`	Value of message_group_id attribute LIME of beitem element
`moduleRef`	Value of module_ref attribute LIME of beitem element
`languageTag`	Value of language_tag attribute LIME of beitem element
`timeMode`	Value of time_mode attribute LIME of beitem element
`timeValue`	Value of time_value attribute LIME of beitem element
`objectId`	Value of object_id attribute LIME of beitem element
`subscribe`	Value of subscribe attribute LIME of beitem element. Specifies whether events are valid or not.

Methods:

None.

10.3 Interface for LIME interrupt event

The event DOM Interface is an extended DOM interface for obtaining the context information (e.g., "Type of Event Occurred" and "Target of Event") of a LIME event. Table 10-7 summarizes the interface corresponding to the LIME events.

Table 10-7 – Profile of interfaces for LIME interrupt event

Interface	Attribute/method	Operation	Remarks
LIMEEvent			
	type	R1	R
	target	R1	R
LIMEIntrinsicEvent			
	keyCode	R1	R
LIMEBeventEvent			
	status	R1	R
	privateData	R1	R
	esRef	R1	R
	messageId	R1	R
	messageVersion	R1	R
	messageGroupId	R1	R
	moduleRef	R1	R
	languageTag	R1	R
	registerId	–	
	serviceId	–	
	eventId	–	
	object	R1	R
NOTE – If the URI string is returned for esRef, moduleRef, etc. The returned value must be in the short format (e.g., "/XX", "/XX/YYYY").			

LIMEEvent interface

LIMEEvent interface retains the context information of a LIME event.

Interface definition:

```
interface LIMEEvent {
readonly attribute DOMString type;
readonly attribute HTMLElement target;
};
```

Attributes:

`type` Name of event.

`target` Target of event. For example, an event for broadcasting service uses LIMEBeitemElement Interface, which is a LIME element DOM interface for beitem element.

Method:

None.

LIMEIntrinsicEvent interface

LIMEIntrinsicEvent Interface retains the Intrinsic Event context information of a LIME event. It is a LIMEEvent with attributes specific to Intrinsic Event.

Interface definition:

```
interface LIMEIntrinsicEvent : LIMEEvent {
readonly attribute unsigned long keyCode;
};
```

Attributes:

`keyCode`	Value of the key for remote control key entry events (onkeydown, onkeypress, and onkeyup). 0 for other events.

Method:

None.

LIMEBeventEvent interface

This interface is a LIMEEvent with attributes specific to broadcasting service events.

Interface Definition:

```
interface LIMEBeventEvent : LIMEEvent {
readonly attribute signed short status;
readonly attribute DOMString privateData;
readonly attribute DOMString esRef;
readonly attribute DOMString messageId;
readonly attribute DOMString messageVersion;
readonly attribute DOMString messageGroupId;
readonly attribute DOMString moduleRef;
readonly attribute unsigned short languageTag;
readonly attribute LIMEObjectElement object;
readonly attribute DOMString segmentId;
};
```

Attributes:

`status`	State after occurrence of event. Negative value: Normal event has not occurred because of an error. Non negative value: Normal event has occurred.
`privateData`	If the event is an event message (EventMessageFired), the character string date written in the privateDataByte field of the received event message is retained. Empty string for other events.

`esRef`	If the event is an event message (EventMessageFired), it must be a URI character string of a component in which the received event message is transmitted. If the event is NPTReferred, it must be a URI character string that identifies the component carrying the NPT reference descriptor. If the event is CCStatusChanged, it must be a URI character string that identifies the referenced subtitle component. If the event is mainAudioStreamChanged, it must be a URI character string that identifies the referenced audio stream and the channel in it. For other events, empty string.
`messageId`	If the event is an event message (EventMessageFired), a value of the upper eight bits of event_msg_id of the received event message. For other events, 0 (zero).
`messageVersion`	If the event is an event message (EventMessageFired), a value of the lower eight bits of event_msg_id of the received event message. For other events, 0 (zero).
`messageGroupId`	If the event is an event message (EventMessageFired), a value of event_msg_group_id of the received event message. For other events, 0 (zero).
`moduleRef`	If the event is a module acquisition event (ModuleUpdated or ModuleLocked), the URI character string of the module. For other events, an empty string.
`languageTag`	If the event is CCStatusChanged, a language identifier value of the subtitle whose presentation status has been changed.
`object`	If the event is a monomedia decoding event (MediaStopped, MediaStarted, or MediaRepeated), object element to that an event is issued. For other events, null.
`segmentID`	When the event is SegmentPlayEnded, segmentID represents the segment that has ended. Otherwise, segmentID is set to null.

Method:

None.

Table 10-8 lists the correspondence between interrupt event and type attribute of LIMEEvent.

Table 10-8 – Correspondence between interrupt event and type attribute of LIMEEvent

Interrupt event	type value
Remote control key was pressed	"keydown"
Remote control key was released	"keyup"
Element was determined by pressing enter key or access key	"click"
Focus was set	"focus"
Focus is out of position	"blur"
Document was loaded	"load"
Document unloading was noticed in advance	"unload"
When the focus on an input element is out, the change of the value attribute of the concerning input element is detected	"change"
Event message was received	"EventMessageFired"
Module update was detected	"ModuleUpdated"
Module was locked	"ModuleLocked"
Timer set by beitem triggered	"TimerFired"
Process such as getNPT() was enabled	"NPTReferred"
Monomedia presentation was stopped	"MediaStopped"
data_event_id update was detected	"DataEventChanged"
Display status of caption is changed	"CCStatusChanged"
Main audio stream is changed	"MainAudioStreamChanged"
Data button was pressed	"DataButtonPressed"
Execution of global codes was started, or the functions specified by executing setTimeout() and setInterval() was started	Undefined (Note)
NOTE – The target attribute is null in this case.	

10.4 LIMECSS2 properties interface for LIME-DOM

The LIMECSS2 properties interfaces are designed with the goal of exposing CSS constructs to object model consumers. Cascading style sheets is a declarative syntax for defining presentation rules, properties and ancillary constructs used to format and render web documents. [b-ITU-T H.740] specifies a mechanism to programmatically access and modify the rich style and presentation control provided by CSS. This augments CSS by providing a mechanism to dynamically control the inclusion and exclusion of individual style sheets, as well as manipulate CSS rules and properties. Table 10-9 shows the profile of the LIMECSS2Properties interface used in LIME-DOM.

The LIME-DOM attribute values below that are to be operated conform to the conventions on operation of the CSS2 properties.

Table 10-9 – Profile of LIMECSS2Properties interface

Property	Operation	Remarks
Box model		
paddingTop	R1	R
paddingRight	R1	R
paddingBottom	R1	R
PaddingLeft	R1	R
borderWidth	R1	R
borderStyle	R1	R
Visual format model		
Left	R1	RW
Top	R1	RW
Width	R1	RW
Height	R1	RW
lineHeight	R1	R
Other visual effects		
visibility	R1	RW
Font		
fontFamily	R1	RW
fontSize	R1	RW
fontWeight	R1	RW
Text		
textAlign	R1	R
letterSpacing	R1	R
LIME extension		
borderTopColorIndex	R1	RW
borderRightColorIndex	R1	RW
borderLeftColorIndex	R1	RW
borderBottomColorIndex	R1	RW
backgroundColorIndex	R1	RW
colorIndex	R1	RW
grayscaleColorIndex	R1	RW
clut	R1	R
Resolution	R1	R
displayAspectRatio	R1	R
navIndex	R1	R
navUp	R1	R

Table 10-9 – Profile of LIMECSS2Properties interface

Property	Operation	Remarks
navDown	R1	R
navLeft	R1	R
navRight	R1	R
usedKeyList	R1	RW

11 Specific functions for IPTV services

11.1 Licensing

The following are functions related to licensing:

– The function to get an IPTV licence: Obtain the licence for the specified content.

– The function to get IPTV licence information: Obtain information concerning the specified licence.

– The function to get DRM ID: Obtain the identifier of the conditional access system (CAS)/digital rights management (DRM) client supporting the specified CAS/DRM.

11.2 Content initialization

The following is the function related to content initialization:

– The function to launch IPTV content: To initialize IPTV content by launching it.

11.3 Service registration

The following are functions related to service registration:

– The function to set IPTV service registration information: To set the basic registration information of linear IPTV and VoD services.

– The function to check IPTV service registration information: To confirm the basic registration information of linear IPTV and VoD services.

11.4 Communication of licence information

The function to set content package information: Set the information for the purchased content package.

The function to update package licence information: Update the information of the licences for all package content.

11.5 Page-transition control

The function to launch an unmanaged document: Changes to a document in the unmanaged state; IPTV unmanaged.

The function to get the document management status: Obtains the information on the management status of the document.

11.6 Control of display

The function to display marquee text: Displays the strings in the "p" element as marquee.

11.7 Parental control function

The function to check the parental control password: Confirms the password for ensuring parental control.

11.8 Use of URI

URI usage has the following operational restrictions:

– Maximum URI size is 1024 bytes.

– URI cannot contain multi-byte characters.

– If the URI refers to a directory, it must contain "/" at the end.

– In case of IPv6 network layer, the URI cannot contain the IP address. In case of IPv4, the IP address can be directly included in the URI.

12 Transport of LIME document and related issues

The LIME document for a portal service is transported using HTTP or HTTPS. The version of HTTP is fixed as HTTP/1.1, and the server is required not to use HTTP/1.0. The action of the receiver when it receives an HTTP/1.0 message is dependent on implementation. The protocol for HTTP/1.1 is required to be compliant with [IETF RFC 2616]. For HTTPS, the receiver and the server are to establish a connection using TLS1.0 and SSL3.0, and then to conduct encrypted communication using HTTP. The versions of transport layer security (TLS) and secure socket layer (SSL) are TLS1.0 and SSL3.0, respectively, and the details of their use are described in [b-IETF RFC 2818].

12.1 Use of HTTP/1.1

– **Communication port**

When the URI is specified as "http:", the receiver and the server are to communicate using HTTP/1.1 at the port specified in the URI. When the URI is specified as "https:", the receiver and the server are to establish a connection using TLS1.0 and SSL3.0, and then to conduct encrypted communication using HTTP/1.1 at the port specified in the URI. If the port number is not specified in the URI, port number 80 is used for "http:" and 443 is used for "https:" as default. However, there are cases where, depending on such factors as firewalls, the port number might be different depending on the connection. The default port can optionally be configured on the receiver, taking into account the connection environment.

– **Format of date and time**

Date and time formats are to use the fixed-length subset defined in [b-IETF RFC 1123]. All date and time stamps are to be in GMT, except where otherwise specified.

- The server is recommended to return to the receiver data/time only in the format as defined in [b-IETF RFC 1123], namely the fixed-length subset.
- The receiver is required to interpret the date and time formatted in the fixed-length subset defined in [b-IETF RFC 1123]. When receiving date and time formatted according to [b-IETF RFC 1036] or [ISO/IEC 9899] asctime() format, the receiver can optionally interpret these formats, or it can also ignore them.

Examples:

Sun, 06 Nov 1994 08:49:37 GMT ([b-IETF RFC 1123]).

Sunday, 06-Nov-94 08:49:37 GMT ([b-IETF RFC 1036]).

Sun Nov 6 08:49:37 1994 ([ISO/IEC 9899]).

– **Content coding**

For content-coding, "identity" is used. "deflate" and "gzip" can optionally be used. If a receiver that does not support "deflate" and "gzip" receives "deflate" and "gzip", or any other value, the expected action is implementation dependent and out of the scope of this Recommendation.

– **Transfer coding**

When receiving a response from the server, the receiver is required to be able to receive "chunked" transfer-coding, specified in [IETF RFC 2616]. When specifying the transfer-coding, "chunked" should be used. The action of the receiver when it receives other values is implementation dependent.

– **Use of request methods**

- "GET": Both the client side and server side use this method.
- "POST": Both the client side and server side use this method.
- "HEAD": The client side can optionally use this method. When receiving the request with "HEAD", the server is required to respond with the format compliant with [IETF RFC 2616].
- "OPTIONS": Both the client side and server side can optionally use this method.

– **Other methods**

The use of "CONNECT", "PUT", "DELETE" and "TRACE" depend on the implementation, and are outside the scope of this Recommendation.

12.2 Supported HTTP request headers

This clause describes HTTP headers for web servers supporting HTTP/1.1 during a request. Table 12-1 lists headers and their respective support level ("S" denotes "supported" and "–" denotes "neither supported nor optional").

Table 12-1 – HTTP headers: Request

	Header name	Header operation		Notes
		Terminal	Server	
General headers	Cache-Control	S	S	Only no-cache is supported
	Connection	S	S	Only close is supported
	Date	–	–	
	Pragma	S	S	Only no-cache is supported, optional for the terminal
	Trailer	–	–	
	Transfer-Encoding	–	–	
	Upgrade	–	–	
	Via	–	–	
	Warning	–	–	

Table 12-1 – HTTP headers: Request

	Header name	Header operation		Notes
		Terminal	Server	
Request header	Accept	S	S	
	Accept-Charset	S	S	
	Accept-Encoding	S	S	Identity, deflate, only gzip supported
	Accept-Language	S	S	Currently fixed to ja
	Authorization	–	–	
	Expect	–	–	
	From	–	–	
	Host	S	S	
	If-Modified-Since	–	S	
	If-Match	–	S	
	If-None-Match	–	S	
	If-Range	–	–	
	If-Unmodified-Since	–	S	
	Max-Forwards	–	–	
	Proxy-Authorization	–	–	
	Range	–	–	
	Referer	–	–	
	TE	–	–	
	User-Agent	S	S	
Entity header	Allow	–	–	
	Content-Encoding	–	–	
	Content-Language	–	–	
	Content-Length	–	–	
	Content-Location	–	–	
	Content-MD5	–	–	
	Content-Range	–	–	
	Content-Type	–	–	
	Expires	–	–	
	Last-Modified	–	–	

12.3 Persistent connections

In case of HTTP/1.1 connections, sessions can be closed by including the "Connection: close" header in the request. If the connection header does not include "close", or in absence of the connection header, the HTTP connection is kept alive. HTTP persistent connections eliminate the need to establish TCP connections for every request, reducing the overall processing time and improving the response.

12.4 User-Agent

User-Agent is a required header. It enables the server to identify the type of IPTV terminal device originating the request. An example of User-Agent information delivered by the IPTV terminal device when establishing a connection with the server(s) is shown in Appendix IV.

12.5 Supported HTTP response headers

This clause describes HTTP headers for web servers supporting HTTP/1.1 during a response. Table 12-2 lists headers and their respective support level ("S" denotes "supported" and "–" denotes "neither supported nor optional").

Table 12-2 – HTTP headers: Response

	Header name	Header operation		Notes
		Terminal	Server	
General headers	Cache-Control	S	S	No-cache, no-store is supported; max-age is optional
	Connection	S	S	Only close is supported
	Date	S	S	
	Pragma	S	S	No-cache is optional
	Trailer	–	–	
	Transfer-Encoding	S	S	Chunked is supported
	Upgrade	–	–	
	Via	–	–	
	Warning	–	–	
Response headers	Accept-Ranges	–	–	
	Age	–	–	
	ETag	–	S	
	Location	S	S	
	Proxy-Authenticate	–	–	
	Retry-After	–	–	
	Server	–	S	
	Vary	–	–	
	WWW-Authenticate	–	–	

Table 12-2 – HTTP headers: Response

	Header name	Header operation		Notes
		Terminal	Server	
Entity headers	Allow	S	S	
	Content-Encoding	S	S	Identity is supported
	Content-Language	S	S	Currently fixed to ja
	Content-Length	S	S	
	Content-Location	S	S	Used within play control meta file
	Content-MD5	–	–	
	Content-Range	–	–	
	Content-Type	S	S	
	Expires	–	S	Use of Cache-Control:max-age as expiration limit is recommended
	Last-Modified	–	S	
Other	Extended headers	–	S	

12.6 Cookies

The use of cookies is based on [IETF RFC 2965]. In order to be interoperable with the existing web servers, the following should be considered.

12.6.1 Use of response header

The receiver is required to be able to interpret the Set-Cookie response header. The interpretation of Set-Cookie2 response header may depend on each implementation.

Table 12-3 shows the parameters that the receiver is required to interpret.

– It is required to interpret ";" as a separator for attributes. The interpretation of "," may depend on implementation.

– The interpretation of attributes other than those listed above depend on implementation.

– One response header contains only one cookie. If more than one cookie needs to be used, more than one response header should be provided. If a response header contains more than one NAME=VALUE attribute, the action of the receiver is up to each implementation and outside the scope of this Recommendation.

– The attribute "expires" is interpreted in the following way:

  1) The date of the reception of Set-Cookie is the current time and/or beyond the valid-through date: The cookie needs to be discarded

  2) The date of the reception of Set-Cookie is within the valid-through date and:

     a) is invalid when the request is received: the cookie in question is discarded, and no cookies are to be sent to the server;

     b) is valid when the request is received: the cookie needs to be kept until the specified date, and a cookie is to be sent to the server. The receiver does not have to guarantee that the cookie is kept until the specified date.

3) No specified value for "expires": how the cookie is discarded is implementation dependent.

Table 12-3 – Parameters receiver is required to interpret

Attributes	Server side requirements	Content
NAME=VALUE	Mandatory	– The main cookie information – The interpretation of double-quote ("), space, tab, LF, CR contained in the VALUE is up to each implementation
domain=DOMAIN	Optional	A valid domain name declared by the cookie
path=PATH	Optional	A valid path declared by the cookie
secure	Optional	When the cookie has "secure" attribute, it is transmitted only if there is a secure connection to the host (e.g., connection to the server using HTTPS).
expires=DATE	Optional	The date until when the cookie is good. The format of DATE follows either of the following: – Wdy,dd Mth yyyy hh:mm:ss GMT – Wdy,dd-Mth-yyyy hh:mm:ss GMT

Annex A

LIME-HTML versions

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

Since new elements and attributes may be added to the specification by extending this specification in the future, a LIME-HTML document must contain a version number that is used to decide whether a LIME-HTML document written with an extended encoding scheme can be viewed by LIME browsers that support only older schemes.

For LIME, the version number consists of a major number and a minor number. The available value range of a major number is 1 to 65535. The available value range of a minor number is 0 to 255. These numbers are represented as a decimal character string with leading zeros suppressed. The version number must be updated as follows.

When a LIME-HTML document in an extended coding scheme can be successfully viewed with older LIME browsers, the minor version number must be updated and the major version number must not be updated. When a LIME-HTML document in an extended coding scheme cannot be successfully viewed without a newer LIME browser, the major version number must be updated. Actual numbering of the version number will be determined in the operation for each media type. The numbering method must be well thought out for the interchange between different types of media.

```
<?bml bml-version="[major number].[minor number]" ?>
```

For LIME-HTML documents following the versioning form "`[major number].[minor number]`" assigned by ARIB, the value is always 100.0. The action of the receiver if other LIME versions are received is up to each implementation and outside the scope of this Recommendation.

Annex B

Multimedia resources

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

B.1 Use of monomedia

B.1.1 Video

It is envisaged that a LIME document would refer to content from IP linear TV and VoD services. Video coding as a monomedia is utilized only as a video elementary stream (ES) within the transport stream (TS) that constitutes the content.

B.1.2 Graphics and bitmap coding

Graphics and bitmap coding are required to comply with guidelines in this clause. The following gives a summary of the requirements.

– **JPEG**: JPEG is required to be in compliance with the baseline method of [ITU-T T.81].

– **PNG**: [ISO/IEC 15948] is required to be used for the portable network graphics (PNG) file format.

NOTE – [ISO/IEC 15948] is the same specification as [b-W3C PNG].

– **MNG**: The specification based on MNG format version 0.96-19990718 (see [b-MNG]) is recommended to be used for file format of animation graphics by multiple-image network graphics (MNG)).

B.1.3 Audio

– It is envisaged that a LIME-HTML document would refer to content from IP linear TV and VoD services. Audio coding of an audio stream as a monomedia is utilized only as an audio ES within the TS that constitutes the content.

– MPEG-1 layer 2 stream.

– MPEG2 AAC-LC audio file.

– "Built-in sound": The encoding method for built-in sound receivers depends on the receiver implementation.

Annex C

Character encoding and font specification

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

C.1 Character specifications

This clause specifies character encoding recommendations specific to a language or group of languages sharing the same character encoding sets.

C.1.1 Character encoding (Japan)

This clause describes the character encoding specific to the Japanese language for LIME documents, closed captions and external files referenced with an object.

C.1.1.1 Character encoding for LIME documents

A LIME document is recommended to use the following character encoding schemes:

– EUC-JP, UTF-8, Shift-JIS.

C.1.1.1.1 EUC-JP

Extended UNIX code (-JP) EUC-JP [b-JIS X 0208] is a Japanese character encoding used predominantly in a UNIX environment. The following character sets can be represented using EUC-JP: JIS X 0201 [b-JIS X 0201] (ASCII, half-width kana), JIS X 0208 [b-JIS X 0208] (two byte) and JIS X 0212 [b-JIS X 0212] (three byte). It is encoded based on ISO/IEC 2022.

C.1.1.2 Character encoding for closed caption subtitles

The character encoding for closed caption subtitles consists of JIS 8-bit encoding characters. Closed caption subtitles are transported within the TS of IP broadcasting or VoD streams in the "subtitle PES" id=0x06. The following character sets are supported:

– Alphanumeric set (1 byte).

– Hiragana (1 byte).

– Katakana (1 byte).

– Chinese character (2 byte code sections 1-94).

– Macro-code (1 byte).

C.1.1.2.1 JIS 8-bit character code

The types of character code set available are Kanji set, alphanumerical set, Hiragana set, Katakana set, mosaic set, supplemental character (Gaiji) set, macro-code set, JIS compatible Kanji plane 1 set, JIS compatible Kanji plane 2 set and additional symbols set.

C.1.2 Character encoding (US, west European)

For further study.

C.1.2.1 Character encoding for LIME documents

For further study.

C.1.2.2 Character encoding for closed caption subtitles

For further study.

C.1.3 Character encoding (east European)

For further study.

C.1.3.1 Character encoding for LIME documents

For further study.

C.1.3.2 Character encoding for closed caption subtitles

For further study.

C.1.4 Character encoding (east Asia, Korea, China)

For further study.

C.1.4.1 Character encoding for LIME documents

For further study.

C.1.4.2 Character encoding for closed caption subtitles

For further study.

C.1.5 Character encoding (Middle East, Arabic, Hebrew, Farsi)

For further study.

C.1.5.1 Character encoding for LIME documents

For further study.

C.1.5.2 Character encoding for closed caption subtitles

For further study.

C.2 Font specifications

C.2.1 LIME font specifications (Japan)

For further study.

C.2.2 LIME font specifications (US, west European)

For further study.

C.2.3 LIME font specifications (east European)

For further study.

C.2.4 LIME font specifications (east Asia, Korea, China)

For further study.

C.2.5 LIME font specifications (Middle East, Arabic, Hebrew, Farsi)

For further study.

Annex D

Data type definition (DTD) for LIME-HTML

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

The name of a data type definition (DTD) file conforms to the following convention that uses major number and minor number in the version information.

```
bml_[major number]_[minor number].dtd
```

For example, the DTD file name for version 1.0 DTD is "bml_1_0.dtd". Note that both major number and minor number are part of a version number that represents DTD; the two numbers are not part of the coding scheme version described below.

```
<!-- ======== Lightweight interactive multimedia environment for IPTV (LIME) x.0
DTD [OPERATABLE] ======== -->
<!ENTITY %    ContentType "CDATA">
<!ENTITY %    Charset "CDATA">
<!ENTITY %    Character "CDATA">
<!ENTITY %    LanguageCode "NMTOKEN">
<!ENTITY %    Number "CDATA">
<!ENTITY %    URI "CDATA">
<!ENTITY %    Script "CDATA">
<!ENTITY %    StyleSheet "CDATA">
<!ENTITY %    Text "CDATA">
<!ENTITY % Events.attrib
"onclick  %Script;   #IMPLIED
onkeydown %Script; #IMPLIED
onkeyup   %Script;  #IMPLIED">
<!ATTLIST a
     onfocus %Script; #IMPLIED
     onblur %Script; #IMPLIED
>
<!ATTLIST input
     onfocus %Script; #IMPLIED
     onblur %Script; #IMPLIED
     onchange %Script; #IMPLIED
>
<!ATTLIST body
     onload %Script; #IMPLIED
     onunload %Script; #IMPLIED
>
<!ATTLIST div
     onfocus %Script; #IMPLIED
     onblur %Script; #IMPLIED
     accesskey %Character; #IMPLIED
>
<!ATTLIST p
     onfocus %Script; #IMPLIED
     onblur %Script; #IMPLIED
     accesskey %Character; #IMPLIED
>
<!ATTLIST object
     onfocus %Script; #IMPLIED
     onblur %Script; #IMPLIED
     accesskey %Character; #IMPLIED
>
<!ATTLIST span
     onfocus %Script; #IMPLIED
     onblur %Script; #IMPLIED
     accesskey %Character; #IMPLIED
```

```
>
<!ENTITY % Core.attrib
"id ID #IMPLIED
class CDATA #IMPLIED
style %StyleSheet; #IMPLIED"
>
<!ENTITY % Common.attrib
"%Core.attrib;
%Events.attrib;"
>
<!ENTITY % Inlstruct.class "br | span">
<!ENTITY % Inline.class "%Inlstruct.class;
| a">
<!ENTITY % Inline-noa.class "%Inlstruct.class;">
<!ENTITY % Blkstruct.class "p | div">
<!ENTITY % Block.class "%Blkstruct.class;">
<!ENTITY % Boxed.mix "%Block.class;
| object
| input">
<!ENTITY % Br.content "EMPTY">
<!ELEMENT br %Br.content;>
<!ATTLIST br
     %Core.attrib;
>
<!ENTITY % Span.content "( #PCDATA | %Inline.class; )*">
<!ELEMENT span %Span.content;>
<!ATTLIST span
     %Common.attrib;
>
<!ENTITY % Div.content "( %Boxed.mix; )*">
<!ELEMENT div %Div.content;>
<!ATTLIST div
     %Common.attrib;
>
<!ENTITY % P.content "( #PCDATA | %Inline.class; )*">
<!ELEMENT p %P.content;>
<!ATTLIST p
     %Common.attrib;
>
<!ENTITY % Script.content "( #PCDATA )">
<!ELEMENT script %Script.content;>
<!ATTLIST script
     src %URI; #IMPLIED
>
<!ENTITY % Style.content "( #PCDATA )">
<!ELEMENT style %Style.content;>
<!ENTITY % A.content "( #PCDATA | %Inline-noa.class; )*">
<!ELEMENT a %A.content;>
<!ATTLIST a
     %Common.attrib;
     href %URI; #IMPLIED
     accesskey %Character; #IMPLIED
>
<!ENTITY % Object.content "EMPTY">
<!ELEMENT object %Object.content;>
<!ATTLIST object
     %Common.attrib;
     data %URI; #IMPLIED
     type %ContentType; #IMPLIED
     remain (remain) #IMPLIED
     streamposition %Number; "0"
     streamstatus (stop | play | pause) #IMPLIED
>
<!ENTITY % InputType.class "( text | password )">
```

```
<!ENTITY % Input.content "EMPTY">
<!ELEMENT input %Input.content;>
<!ATTLIST input
     %Common.attrib;
     type %InputType.class; "text"
     value CDATA #IMPLIED
     disabled (disabled) #IMPLIED
     readonly (readonly) #IMPLIED
     maxlength %Number; "40"
     accesskey %Character; #IMPLIED
     inputmode (direct | indirect | none) "none"
     charactertype (all|number|alphabet|hankaku|zenkaku|katakana|hiragana) "all"
>
<!ENTITY % Title.content "( #PCDATA )">
<!ELEMENT title %Title.content;>
<!ENTITY % Meta.content "EMPTY">
<!ELEMENT meta %Meta.content;>
<!ATTLIST meta
     name NMTOKEN #IMPLIED
     content CDATA #REQUIRED
>
<!ENTITY % Head.content "( title, meta?, style?, link?, script*, bevent? )">
<!ELEMENT head %Head.content;>
<!ENTITY % Body.content "( div | p )+">
<!ELEMENT body %Body.content;>
<!ATTLIST BODY
     %Core.attrib;
     invisible (invisible) #IMPLIED
>
<!ENTITY % Bml.content "( head, body )">
<!ELEMENT bml %Bml.content;>
<!ENTITY % bevent.content "( beitem )+">
<!ELEMENT bevent %bevent.content;>
<!ATTLIST bevent
     id ID #IMPLIED
>
<!ENTITY % BMLEventType
"(TimerFired|CCStatusChanged|MediaStopped|DataButtonPressed)">
<!ENTITY % BMLTimeMode "(absolute)">
<!ENTITY % beitem.content "EMPTY">
<!ELEMENT beitem %beitem.content;>
<!ATTLIST beitem
     id ID #REQUIRED
     type %BMLEventType; #REQUIRED
     onoccur %Script; #REQUIRED
     es_ref %URI; #IMPLIED
     language_tag %Number; #IMPLIED
     time_mode %BMLTimeMode; #IMPLIED
     time_value CDATA #IMPLIED
     object_id CDATA #IMPLIED
     subscribe (subscribe) #IMPLIED
>
<!ENTITY % link.content "EMPTY">
<!ELEMENT link %link.content;>
<!ATTLIST link
     href %URI; #IMPLIED
>
<!-- End of Lightweight interactive multimedia environment for IPTV (LIME) x.0
DTD -->
```

A DTD declaration is as follows:

```
<?xml version="1.0" encoding="EUC-JP" ?>
<!DOCTYPE bml PUBLIC
```

```
     "-//IPTVF CDN:2008//DTD BML Document for IPTV//JA"
     "http://www.itu.int/ITU-T/formal-language/itu-
t/h/h762/2011/bml_100_0_iptv.dtd">
<?bml bml-version="100.0" ?>
```

Appendix I

Browser functions for LIME

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

I.1 Video and graphics display

An example of the presentation functionality of the receiver can be found in [ARIB STD-B24] Volume 1, Section 1.

To reproduce the multimedia service sent from the service provider on screen just as the producer intended through the receiver, display and playback functions on the receiver should be specified. Therefore, a specification related to the presentation function is necessary as a basic requirement of the receiver. The presentation function is designed based on the logic structure of the display screen composed of video plane, still picture plane, text and graphic plane, subtitle plane, and control plane switching and controlling video and still picture (see [ARIB STD-B-24] Volume 1, Chapter 6.2).

Table I.1 describes the conditions on the monomedia coding for each presentation plane. It is presupposed that monomedia data other than those specified here will not be sent or used by the source content provider.

Table I.1 – Overview of the conditions on the monomedia coding for each presentation plane

Coding		Conditions	
Video coding	ITU-T H.264	Transport method	Video packetized elementary stream (PES) in TS for linear IPTV or VoD streaming Stream format Id = 0x1B
		Size	1920x1080 (16:9), 1440x1080 (16:9), 1280x720 (16:9), 720x480 (16:9), 720x480 (4:3)
		Scaling	256/128, 192/128, 160/128, 128/128, 112/128, 96/128, 80/128, 64/128, 48/128, 32/128 (Note)
	MPEG-2	Transport method	Video PES in TS for linear IPTV or VoD streaming Stream format Id = 0x02
		Size	720x480 (16:9), 720x480 (4:3)
		Scaling	256/128, 192/128, 160/128, 128/128, 112/128, 96/128, 80/128, 64/128 (Note)
Graphics coding	JPEG	Transport method	JPEG file via HTTP
		Size	Any from horizontal/vertical 16 pixels to full size
		Scaling	128/128
		Other	Resolution of 4:2:0 scheme is assumed

Table I.1 – Overview of the conditions on the monomedia coding for each presentation plane

Coding		Conditions	
Character/geometrics coding	PNG	Transport method	PNG file via HTTP
		Size	Any from horizontal/vertical 2 pixels to full size
		Scaling	128/128
	MNG	Transport method	MNG file via HTTP
		Size	Any from horizontal/vertical 2 pixels to full size
		Scaling	128/128
	8-unit character coding, including EUC-JP	Transport method	For use in captioning: Captioning PES in TS for linear IPTV or VoD streaming (stream format Id = 0x06) For use in portal: LIME document file via HTTP
NOTE – The scaling factor should be compliant with the definitions in [b-ARIB TR-B14], Volume 3, Section 2, A4.			

I.2 Audio playback

Table I.2 describes the specification for audio playback. It is presupposed that monomedia data other than those specified here will not be sent or used by the source content provider.

Table I.2 – Audio playback specification

Coding method	Content	
MPEG-2 AAC-LC	Transmission methods	Audio PES; stream format identifier = 0x0F Audio file; HTTP
	Sampling rate	48 kHz
	Maximum file size of continuous playback	512 kilobytes
MPEG-1 audio layer 2	Transmission methods	Audio PES stream format identifier = 0x03
	Sampling rate	48 kHz, 32 kHz
Built-in sound encoding	Transmission methods	Audio file; HTTP
	Sampling rate	1/4 of the 12-kHz main sound track
	Maximum file size of continuous playback	96 kilobytes
Caption alert	Transmission methods	Built-in sound
	Sampling rate	12 kHz
	Maximum file size of continuous playback	48 kilobytes

I.3 Remote controller

LIME assumes that, compliant with [b-ARIB TR-B14] Volume 3, Section 2, chapter 1.3, the remote controller is provided so that the keys in Table I.3 are accessible to the LIME browser. In order to avoid user confusion, multiple meanings should not be assigned to one button. When assigning multiple meanings to one button, operation content should be explicitly explained to the user within the contents.

Table I.3 – Remote control keys used

Key type	Guidelines
Up, down, left, right arrow keys	To move up, down, left, right
0-9 (number keys)	To input numbers
Enter	Separator of operation (enter)
Return	Cancel operation
	Back space of user input character (or bulk erase)
	Disconnection of a call to a communication server. During connection (Note), receiver units will take the instruction; after connection, instruction is carried out in the contents (a display to the effect that the connection will be terminated is desirable when the back key is pressed). NOTE – It is acceptable to use LIME documents for the purpose of going back. However, whether or not there is something available after returning should be considered.
D	Data button: switches display/non-display of multimedia data broadcasting
Blue, red, green, yellow (colour keys)	Selection of operation (execution). NOTE – Location of buttons on the remote control should be in order of blue, red, green, yellow from the left and each button should have the corresponding words "blue", "red", "green" and "yellow" displayed.
Bookmark (optional)	Recording of bookmark

I.4 Key masks

If multimedia content is in compliance with [ARIB STD-B24], then key masks can be performed. However, keys related to selecting stations (one-touch select button, channel up/dedicated button, screen image key) should not be masked by contents except during on-line communication. Masks on number keys (one-touch select button) should not be performed unless number input is necessary. Masks should be released once the input is over.

I.5 Character entry function

The character entry function, assuming there is a software keyboard, etc., for the purpose of supporting character entry to LIME contents by viewer operation, is defined as a resident application. The details are to be compliant with [b-ARIB TR-B14] Volume 3, Section 2, 1.6.

Appendix II

An example of a LIME document

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

```
<?xml version="1.0" encoding="EUC-JP" ?>
<!DOCTYPE bml PUBLIC
     "-//IPTVF CDN:2008//DTD BML Document for IPTV//JA"
     "http://www.itu.int/ITU-T/formal-language/itu-
t/h/h762/2011/bml_100_0_iptv.dtd">
<?bml bml-version="100.0" ?>
<bml>
     <head>
     <title>An example of an LIME document</title>
     <style>
     <![CDATA[
     p {
     left:0px; width:640px; height:25px;
     background-color-index:5;
     }
     p:focus {background-color-index:0;}
     ]]>
     </style>
     <script>
     // example of a script
     <![CDATA[
     var img = document.getElementById("id_1");
     img.data = "photo2.jpg";
     ]]>
     </script>
     </head>
<!-- Comment: Beginning of the body -->
     <body style="background-color-index:7;">
     <div id="d" style="width:320px;height:480px">
     <object id="id_1" type="image/jpeg" data="photo1.jpg"
style="width:260px;height:180px;"/>
     <p style="width:760px;height:20px;">Hello IPTV World!!!</p>
     </div>
     </body>
</bml>
```

Appendix III

Implementation example of LIME-Script

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

III.1 Implementation example of LIME-Script

In a LIME document that contains more than one script element, when all the scripts (i.e., the script described in the resource designated by the `src` attribute of the `script` element and the internal script written within a `script` element without an `src` attribute) are loaded, the following restrictions apply.

As in [ARIB STD-B24] volume 2, A3-5.4.1, "Operation of script working environment", the following restriction may be put on script work memory.

Table III.1 shows an operation example of script work memory.

Table III.1 – Restriction example on the script work memory

Item	Maximum value	Remarks
Length of a symbol name character string	255 bytes	–
Function arguments	255	–
Local variables	255	–
Total length of all character strings	131 072 bytes	The total length of strings (including evaluated values of string equations, string constants, string variables) and symbol names.
Instances of objects	516	Total number of instances of Object, Number, String, Boolean, Array, Date, Binary Table and Function.
Properties of one instance of one object	256	Maximum number of properties for each instance of Object, Number, String, Boolean, Array, Date, Binary Table or Function. For Array, multiple properties that have corresponding subscripts are not counted.
Elements of one array	1024	–
Nest levels with function for invoking	32	Including functions invoked through event handlers.
Total number of properties of all objects	8192	The total number of properties to which the "properties of one instance of one object" restriction is applied. Including number of properties of activation objects and argument objects. Excluding built-in properties of global objects (built-in functions, built-in objects, extended functions for broadcasting, extended objects for broadcasting) and properties of host objects.
Global variables	256	–
Function definitions	256	Any function is defined globally. Excluding event handlers.
Work memory for LIME script	1648 steps	Based on the computing method defined in [ARIB STD-B24] Vol.2 Annex C "Counting rule for the restriction on the memory size of ECMAScript."

It is assumed that any data type has the following restrictions:

– Number must be of single precision (32 bits). Note that any Number object must also be of single precision (32 bits).
– Float must not be supported.
– Math built-in objects must not be supported.
– Dynamic type conversion must be restricted.
– The run-time interruption of a script character string must not be supported.
– EUC-JP must be used as the character coding scheme of character string data.
– Use of Unicode value must be restricted.
– Functions for compatibility with old codes must be restricted.

III.1.1 Effects on basic objects caused by data type restrictions

As in [ARIB STD-B24] volume 2, A2-5.4.3, "Effects on basic objects caused by data type restrictions", the following effects are expected.

III.1.2 Effects caused by number object of signed 32-bit integer

The details are referred to [ARIB STD-B24] Volume 2, A2-5.4.3.1.

III.1.3 Behaviour in the case of not using Float

The details are referred to [ARIB STD-B24] Volume 2, A2-5.4.3.2.

III.1.4 Effects by restrictions on run-time interpretation of script character string

The details are referred to [ARIB STD-B24] Volume 2, A2-5.4.3.3.

III.1.5 Behaviour in case of using EUC-JP character code

The details are referred to [ARIB STD-B24] Volume 2, A2-5.4.3.4.

III.1.6 Effects by operation of a specific character set

The details are referred to [ARIB STD-B24] Volume 2, A2-5.4.3.5.

III.1.7 Restrictions of functions left for compatibility with older codes

The details are referred to [ARIB STD-B24] Volume 2, A2-5.4.3.6.

III.2 Operational general rule of implementation-dependent behaviour

Compliant with [ARIB STD-B24] Volume 2, A2-5.4.4, "Operational general rule of implementation-dependent behaviour", the following conditions are set on the implementation of LIME.

III.3 Main syntax

– Sequence of the properties taken out by "for (variable in Expression) statement" (Page 58 of [b-ISO/IEC 16262]).

  It must remain implementation dependent. The "for (variable in Array object)" sequence is also implementation dependent.

III.4 Host object

– Host object range.
– DOM object.
– Browser pseudo object.

– Result of typeof for host object (Page 43 of [b-ISO/IEC 16262]).

  Returns: "hostobject".

– Result of `new Object(hostobject)` for host object (Page 66 of [b-ISO/IEC 16262]) returns reference to hostobject in the same manner as when using a normal object as an argument.

– Results of `Array.prototype.join()`, `Array.prototype.reverse()` and `Array.prototype.sort()` for host object (Pages 71, 72 and 73 of [b-ISO/IEC 16262], clause 15.4.4.3) inhibits addition of any property to the host object.

  NOTE – There is no need to consider this behaviour because it becomes impossible to insert `Array.prototype.join`, etc., as a new property of the host object.

– [[Class]] of host object.

  "hostobject".

III.5 Built-in object

– [[prototype]] (page 62 of [b-ISO/IEC 16262]) of Global object null.

– Result of `Function.prototype.toString()` (Page 69 of [b-ISO/IEC 16262]).

  "`function FUNCTIONNAME() {}`", where `FUNCTIONNAME` is the name of a specified function.

– Result of `Array.prototype.sort()` (Page 72 of [b-ISO/IEC 16262]).

  An array element that is not influenced on sorting may not be automatically generated as undefined. The internal comparison sequence of `sort()` may depend on implementation.

– Result of `Number.prototype.toString(radix)` without radix = 10 (Page 81 of [b-ISO/IEC 16262]).

  Only radix = 8, 10 or 16 is applicable. Otherwise, operation depends on implementation.

  radix = 8: ["0"-"7"]*.

  radix = 16: ["0"-"9","a"-"f"]*.

– Result of `Date.prototype.toString()` (Page 95 of [b-ISO/IEC 16262]).

  This must be in the format of "DateThoursminutesseconds".

  Date must be YYYY-MM-DD (e.g., 1999-01-01).

  Hours, minutes and seconds must be hh:mm:ss. (e.g., 23:01:34).

  'T' (character code 0x54) must be used as a delimiter between the date and the hours, minutes and seconds. (e.g., 1999-01-01T23:01:34).

  If the result is a negative value, the low-order four digits are used and the sign (d.c. or a.c.) is ignored.

– Results of `Date.prototype.toLocaleString()` and `Date.prototype.toUTCString()` (pages 100 and 101 of [b-ISO/IEC 16262]) must be of the same output format as `Date.prototype.toString()`.

– `TimeClip()` clip range (see clause 15.9.1.14 of [b-ISO/IEC 16262]) must be within the range of signed 64-bit integers.

III.6 Implementation of event handler

The code type of the event handler must be an implementation-supplied code. Also, neither `ImplicitThis` nor `ImplicitParent` must be set. Even if an event handler return value is `false`, the next processing is performed continuously regardless of its value.

Appendix IV

Example of user-agent information

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

The following is an example of User-Agent information delivered by the IPTV terminal device when establishing a connection with the server(s).

```
User-Agent Operationial Guideline
User-Agent: *[Other Description] IptvServiceProduct IptvServiceComment *[Other
Description]
IptvServiceProduct ::= IptvServiceAppName "/" IptvServiceSpecVersion
IptvServiceAppName ::= "IptvSvcClient"
IptvServiceSpecVersion ::= <Version of the Specification>
IptvServiceComment ::= "(" MakerId ";" ModelId ";" MajorVer ";" MinorVer
*[";" Optional Other Description] ")"
MakerId ::= <string identifying Manufacturer>
ModelId ::= <string identifying Model>
MajorVer ::= <Major version number >
MinorVer ::= <Minor version number >
Other Description ::= <any description, can not start with "IptvSvcClient/" >
Optional Other Description for future use
Refer to [IETF RFC2616] for allowed strings.
For example:
IptvSvcClient/1.0 (008045;D40;001;000)
Mozilla/4.0 (compatible;ABCD;EFG;HIJ) IptvSvcBrowser/1.0 (008045;D40;001;000)
```

Bibliography

[b-ITU-T H.740] Recommendation ITU-T H.740 (2010), *Application event handling for IPTV services*.

[b-ITU-T Y.101] Recommendation ITU-T Y.101 (2000), *Global Information Infrastructure terminology: Terms and definitions*.

[b-ITU-T Y.1901] Recommendation ITU-T Y.1901 (2009), *Requirements for the support of IPTV services*.

[b-ISO/IEC 16262] ISO/IEC 16262:2002, *Information technology – ECMAScript language specification*.

[b-IETF RFC 1036] IETF RFC 1036 (1987), *Standard for interchange of USENET messages*.

[b-IETF RFC 1123] IETF RFC 1123 (1989), *Requirements for Internet Hosts – Application and Support*.

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

[b-ARIB TR-B14] ARIB TR-B14 V.2.8 (2006), *Operational Guidelines for Digital Terrestrial Television Broadcasting.*

[b-IPTVFJ-0006] IPTV Forum Japan IPTVFJ STD-0006 V.1.1 (2008), *IPTV Specification – CDN Scope Service Approach Specification* (Japanese).

[b-JIS X 0208] JIS X 0208 (1997), *7-bit and 8-bit double byte coded KANJI sets for information interchange*.

[b-JIS X 0201] JIS X 0201 (1997), *7-bit and 8-bit coded character sets for information interchange*.

[b-JIS X 0212] JIS X 0212 (1990), *Code of the supplementary Japanese graphic character set for information interchange*.

[b-MNG] MNG V.0.96 (1999), *Multiple-image Network Graphics (MNG) format*. <http://www.libpng.org/pub/mng>.

[b-W3C CSS1] W3C Recommendation CSS1 (2008), *Cascading Style Sheets, level 1*.

[b-W3C CSS2] W3C Recommendation CSS2 (2007), *Cascading Style Sheets, level 2*.

[b-W3C DOM1] W3C Recommendation DOM1 (1998), *Document Object Model (DOM) Level 1 Specification*.

[b-W3C PNG] W3C Recommendation PNG (2003), *Portable Network Graphics (PNG) Specification*.

[b-W3C XHTML] W3C Recommendation XHTML (2002), *XHTML 1.0, The Extensible HyperText Markup Language (Second Edition).*

[b-W3C XML 1.0] W3C Recommendation XML 1.0 (2008), *Extensible Markup Language (XML) 1.0 (Fifth Edition).*

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, 2012